

Studies in Computational Intelligence 812

Luca Maria Aiello
Chantal Cherifi
Hocine Cherifi
Renaud Lambiotte
Pietro Lió
Luis M. Rocha *Editors*

Complex Networks and Their Applications VII

Volume 1 Proceedings The 7th
International Conference on Complex
Networks and Their Applications
COMPLEX NETWORKS 2018

Studies in Computational Intelligence

Volume 812

Series editor

Janusz Kacprzyk, Polish Academy of Sciences, Warsaw, Poland
e-mail: kacprzyk@ibspan.waw.pl

The series “Studies in Computational Intelligence” (SCI) publishes new developments and advances in the various areas of computational intelligence—quickly and with a high quality. The intent is to cover the theory, applications, and design methods of computational intelligence, as embedded in the fields of engineering, computer science, physics and life sciences, as well as the methodologies behind them. The series contains monographs, lecture notes and edited volumes in computational intelligence spanning the areas of neural networks, connectionist systems, genetic algorithms, evolutionary computation, artificial intelligence, cellular automata, self-organizing systems, soft computing, fuzzy systems, and hybrid intelligent systems. Of particular value to both the contributors and the readership are the short publication timeframe and the world-wide distribution, which enable both wide and rapid dissemination of research output.

More information about this series at <http://www.springer.com/series/7092>

Luca Maria Aiello · Chantal Cherifi
Hocine Cherifi · Renaud Lambiotte
Pietro Lió · Luis M. Rocha
Editors

Complex Networks and Their Applications VII

Volume 1 Proceedings The 7th International Conference on Complex Networks and Their Applications COMPLEX NETWORKS 2018



Springer

Editors

Luca Maria Aiello
Nokia Bell Labs
Cambridge, UK

Chantal Cherifi
IUT Lumière
University of Lyon
Bron Cedex, France

Hocine Cherifi
LE2I UMR CNRS 6306 9
University of Burgundy
Dijon Cedex, France

Renaud Lambiotte
Mathematical Institute
University of Oxford
Oxford, UK

Pietro Lió
Department of Computer Science
and Technology, The Computer Laboratory
University of Cambridge
Cambridge, UK

Luis M. Rocha
Center for Complex Networks and Systems
Research, School of Informatics,
Computing, and Engineering
Indiana University
Bloomington, IN, USA

ISSN 1860-949X

ISSN 1860-9503 (electronic)

Studies in Computational Intelligence

ISBN 978-3-030-05410-6

ISBN 978-3-030-05411-3 (eBook)

<https://doi.org/10.1007/978-3-030-05411-3>

Library of Congress Control Number: 2018963067

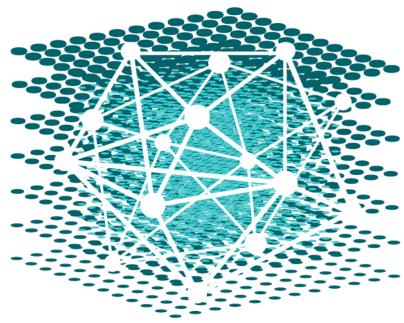
© Springer Nature Switzerland AG 2019

This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed.

The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

The publisher, the authors and the editors are safe to assume that the advice and information in this book are believed to be true and accurate at the date of publication. Neither the publisher nor the authors or the editors give a warranty, express or implied, with respect to the material contained herein or for any errors or omissions that may have been made. The publisher remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

This Springer imprint is published by the registered company Springer Nature Switzerland AG
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland



**COMPLEX
NETWORKS**

Preface

The International Conference on Complex Networks and their Applications has been initiated in 2011 and has grown to become one of the major international events in network science.

This year it is hosted at the University of Cambridge which has a long, successful tradition for cultivating interdisciplinarity and multidisciplinarity.

The variety of scientific topics ranges from network and graph theory, statistical mechanics, models of networks, community structure, geometry, visualization, financial and economic networks, social and urban networks, human behavior, epidemic models, political networks, earth sciences applications, control and synchronization, resilience and robustness, machine learning and other fields of computer science, network medicine and neuroscience, and many others. Therefore, only a large conference such as The International Conference on Complex Networks and their Applications could exploit the “speciation” of the network fields to generate inspiration, enchantment, and cross-fertilization between fundamental issues and innovative applications.

This is clearly reflected in the volume of proceedings that contain a high-quality selection of papers presented during the seventh edition, hosted at the Department of Computer Science and Technology of the University of Cambridge (UK)—a truly place of innovators—from December 11 to December 13, 2018.

Numbers talk by themselves: We received 421 submissions originating from more than 50 countries around the world; in particular, we received submissions from countries usually not represented in other similar conferences. Each paper has been peer-reviewed by three independent reviewers from a large international program committee. After the review process, 123 papers were selected to be included in the proceedings.

Certainly, the success of the conference depends on the authors who have produced high-quality papers. It goes also to the impressive list of keynote speakers. Our speakers are:

- Vittoria Colizza (INSERM, France): “Vulnerability of Networked Host Populations to Epidemics”

- Aristides Gionis (Aalto University, Finland): “Maximizing Diversity in Social Networks”
- Heather Harrington (Oxford University): “Topological Data Analysis for Investigation of Dynamics and Biological Networks”
- Sune Lehmann (Technical University of Denmark, Denmark): “Measuring Social Networks with High Resolution: What Have We Learned?”
- Hernan Makse (City College of New York, USA): “Essential Nodes and Keystone Species in the Brain, Ecosystems and Social Systems”
- Romualdo Pastor-Satorras (Universitat Politècnica de Catalunya, Spain): “Effects of Social Influence on Collective Motion”
- Markus Strohmaier (RWTH Aachen University, Germany): “Modeling Minorities in Social Networks”
- Donald Towsley (UMass Amherst, USA): “Motifs in Social Networks.”

These keynote talks provide a remarkable tessellation of the entire field of complex networks. In some sense, the ensemble of the topics addressed by the keynote speakers provides itself an example of data integration.

Traditionally, prior to the conference, two tutorials are delivered. This year, Jesús Gómez-Gardeñes (University of Zaragoza, Spain) gave a lecture entitled “Network Epidemiology: From Simple to Data-Driven Models” and Silvio Lattanzi (Google Zurich, Switzerland) delivered the lecture “From Micro to Macro: Ego-Network Analysis and Its Applications.”

Each edition of the conference represents a challenge that cannot be successfully achieved without the deep involvement of plenty of people, institutions, and sponsors.

First of all, our sincere gratitude goes to the members of the advisory board: Jon Crowcroft (University of Cambridge, UK), Raissa D’Souza (University of California, Davis, USA), Eugene Stanley (Boston University, USA), and Ben Y. Zhao (University of Chicago, USA). They inspire the essence of the conference.

We record our thanks to our fellow members of the Organizing Committee.

Elsa Arcaute (UCL, UK), Márton Karsai (ENS/INRIA Lyon, France), and Jat Singh (University of Cambridge, UK), the poster chairs for arranging the poster sessions program.

Sebastian Anhert (University of Cambridge, UK), Jean-Charles Delvenne (UC Louvain, Belgium), and Huijuan Wang (TU Delft, The Netherlands), for arranging and chairing the lightning sessions program.

Jinhu Lü (Chinese Ac. Science, Beijing, China), Sarah Morgan (University of Cambridge, UK), and Eiko Yoneki (University of Cambridge, UK), the tutorial chairs for managing the tutorials.

Carlos Gershenson (UNA de Mexico, Mexico), Bruno Gonçalves (New York University, USA), Leto Peel (UC Louvain, Belgium), and Feng Xia (Dalian University of Technology, China), the publicity chairs for encouraging participation from, respectively, America, Europe, and Asia.

We extend our thanks to the Milan team (University of Milan, Italy): Sabrina Gaito, the publication chair, Matteo Zignani, the web chair, and Christian Quadri, the submission chair for the incredible work they have done, in maintaining the Web site and editing the proceedings and the book of abstracts.

We would also like to record our appreciation for the work of the Local Committee chairs, Jat Singh and Petra Vertes and members, William Grant, Giovanna Maria Dimitri, Sandra Servia Rodriguez, and Helena Andres for their tremendous work to make this edition a success.

We are also indebted to our partners, Alessandro Fellegara and Alessandro Egro (Tribe Communication), for their passion and patience in designing such an amazing visual identity of the Conference.

We would like to express our gratitude to the editors involved in the sponsoring of the conference: Cambridge University Press, Elsevier, MDPI, Plos, World Scientific, and Springer Nature.

We would also like to express our deepest appreciation to all those who make the growing success of this meeting year after year. Sincere thanks to the contributors, the richness of the technical program would not be possible without their creativity.

Finally, we would like to express our most sincere thanks to the Program Committee members for their huge effort to produce more than 1200 high-quality reviews within a very short time.

These volumes make the most advanced contribution of the international community to the research issues surrounding the fascinating world of complex networks.

We hope that you enjoy the papers as much as we enjoyed organizing the conference and putting this collection together.

December 2018

Luca Maria Aiello
Chantal Cherifi
Hocine Cherifi
Renaud Lambiotte
Pietro Lió
Luis M. Rocha

Organization and Committees

General Chairs

Hocine Cherifi	University of Burgundy, France
Renaud Lambiotte	University of Oxford, UK
Pietro Lió	University of Cambridge, UK

Advisory Board

Jon Crowcroft	University of Cambridge, UK
Raissa D’Souza	University of California, Davis, USA
Eugene Stanley	Boston University, USA
Ben Y. Zhao	University of Chicago, USA

Program Co-chairs

Luca Maria Aiello	Nokia Bell Labs, UK
Chantal Cherifi	University of Lyon 2, France
Luis Mateus Rocha	Indiana University, USA

Poster Chairs

Elsa Arcaute	UCL, UK
Márton Karsai	ENS/INRIA Lyon, France
Jat Singh	University of Cambridge, UK

Lightning Chairs

Sebastian Anhert

University of Cambridge, UK

Jean-Charles Delvenne

UC Louvain, Belgium

Huijuan Wang

TU Delft, The Netherlands

Media and Publicity Chairs

Bruno Gonçalves

New York University, USA

Feng Xia

Dalian University of Technology, China

Carlos Gershenson

UNA de Mexico, Mexico

Leto Peel

UC Louvain, Belgium

Tutorial Chairs

Jinhu Lü

Chinese Academy of Sciences, China

Sarah Morgan

University of Cambridge, UK

Eiko Yoneki

University of Cambridge, UK

Local Chairs

Jat Singh

University of Cambridge, UK

Petra Vertes

University of Cambridge, UK

Local Committee

William Grant

University of Cambridge, UK

Giovanna Maria Dimitri

University of Cambridge, UK

Sandra Servia Rodriguez

University of Cambridge, UK

Helena Andres

University of Cambridge, UK

Publication Chair

Sabrina Gaito

University of Milan, Italy

Submission Chair

Christian Quadri

University of Milan, Italy

Web Chair

Matteo Zignani

University of Milan, Italy

Program Committee

Filip Agneessens	University of Surrey, UK
Nesreen Ahmed	Intel Labs, USA
Sebastian Ahnert	University of Cambridge, UK
Masaki Aida	Tokyo Metropolitan University, Japan
Marco Aiello	University of Stuttgart, Germany
Luca Maria Aiello	Nokia Bell Labs, UK
Deepak Ajwani	Nokia Bell Labs, UK
Tatsuya Akutsu	Kyoto University, Japan
Reka Albert	The Pennsylvania State University, USA
Antoine Allard	Laval University, Canada
Claudio Altafini	Linköping University, Sweden
Lucila Alvarez-Zuzek	IFIMAR-UNMdP, Argentina
Fred Amblard	IRIT-University Toulouse 1 Capitole, France
Marco Tulio Angulo	National Autonomous University of Mexico, Mexico
Alberto Antonioni	Carlos III University of Madrid, Spain
Nino Antulov-Fantulin	ETH Zurich, Switzerland
Nuno Araujo	Universidade de Lisboa, Portugal
Elsa Arcaute	University College London, UK
Alex Arenas	URV, Spain
Panos Argyraakis	Aristotle University of Thessaloniki, Greece
Tomaso Aste	University College London, UK
Martin Atzmueller	Tilburg University, The Netherlands
Rodolfo Baggio	Bocconi University, Italy
James Bagrow	University of Vermont, USA
Sven Banisch	Max Planck Institute, Germany
Yaneer Bar-Yam	New England Complex Systems Institute, USA
Paolo Barucca	University of Zurich, Switzerland
Anastasia Baryshnikova	Calico Life Sciences, USA
Nikita Basov	St. Petersburg State University, Russia
Gareth Baxter	University of Aveiro, Portugal
Mariano Beguerisse Diaz	Spotify Limited, Sweden
Rosa M. Benito	Universidad Politécnica de Madrid, Spain
Jacob Biamonte	University of Malta, Malta
Ginestra Bianconi	Queen Mary University of London, UK
Jeremy Blackburn	University of Alabama at Birmingham, USA
Anthony Bonato	Ryerson University, Canada
Javier Borge-Holthoefer	Internet Interdisciplinary Institute, Spain

Pierre Borgnat	ENS de Lyon, France
Stefan Bornholdt	Universität Bremen, Germany
Dan Braha	NECSI, USA
Ulrik Brandes	ETH Zurich, Switzerland
Lidia A. Braunstein	UNMdP, Argentina
Markus Brede	University of Southampton, UK
Marco Bressan	Sapienza University of Rome, Italy
Dirk Brockmann	Humboldt University, Germany
Piotr Bródka	Wrocław University of Science and Technology, Poland
Raffaella Burioni	Università di Parma, Italy
Kanat Camlibel	University of Groningen, Germany
Paolo Campana	University of Cambridge, UK
Carlo V. Cannistraci	TU Dresden, Germany
Vincenza Carchiolo	Università di Catania, Italy
Alessio Cardillo	University of Bristol, UK
Rui Carvalho	Durham University, UK
Giona Casiraghi	ETH Zurich, Switzerland
Ciro Cattuto	ISI Foundation, Italy
Remy Cazabet	University of Lyon, France
L. Elisa Celis	EPFL, Switzerland
Tanmoy Chakraborty	IIIT Delhi, India
David Chavalarias	CNRS, France
Kwang-Cheng Chen	University of South Florida, USA
Hocine Cherifi	University of Burgundy, France
Chantal Cherifi	University of Lyon, France
Peter Chin	Boston University, USA
Fu Lai Chung	Hong Kong Polytechnic University, Hong Kong
Matteo Cinelli	University of Rome Tor Vergata, Italy
Richard Clegg	Queen Mary University of London, UK
Giacomo Como	Politecnico di Torino, Italy
Luciano Costa	Universidade de SaPaulo, Brazil
Emanuele Cozzo	BiFi, Spain
Regino Criado	Universidad Rey Juan Carlos, Spain
Matthieu Cristelli	CNR, Italy
Mihai Cucuringu	University of California, Los Angeles, USA
Bhaskar Dasgupta	University of Illinois at Chicago, USA
Joern Davidsen	University of Calgary, Canada
Fabrizio De Vico Fallani	Inria-ICM, France
Matthias Dehmer	UMIT, Austria
Charo I. Del Genio	The University of Warwick, UK
Michela Del Vicario	IMT Lucca, Italy
Pietro Delellis	University of Naples Federico II, Italy
Jean-Charles Delvenne	University of Louvain, Belgium
José Devezas	INESC TEC and DEI-FEUP, Portugal

Matías A. Di Muro	IFIMAR-CONICET, Argentina
Wenbo Du	Beihang University, China
Louis J Dubé	Laval University, Canada
Jordi Duch	Universitat Rovira i Virgili, Spain
Marten During	C2DH, Luxembourg
Mohammed El Hassouni	Mohammed V University in Rabat, Morocco
Michael T. M. Emmerich	Leiden University, The Netherlands
Frank Emmert-Streib	Tampere University of Technology, Finland
Gunes Ercal	SIUE, USA
Pau Erola	University of Bristol, UK
Ernesto Estrada	University of Strathclyde, UK
Tim Evans	Imperial College London, UK
Mauro Faccin	Université Catholique de Louvain, Belgium
Giorgio Fagiolo	Sant'Anna School of Advanced Studies, Italy
Shobeir Fakhraei	University of Southern California, USA
Kate Farrahi	University of Southampton, UK
Alessandro Flammmini	Indiana University Bloomington, USA
Manuel Foerster	University of Hamburg, Germany
Mattia Frasca	University of Catania, Italy
Xinchu Fu	Shanghai University, China
Angelo Furno	University of Lyon, France
Sabrina Gaito	University of Milan, Italy
Lazaros Gallos	Rutgers University, USA
José Manuel Galán	Universidad de Burgos, Spain
Yerali Gandica	Université Catholique de Louvain, Belgium
Jianxi Gao	Rensselaer Polytechnic Institute, USA
David Garcia	Medical University of Vienna, Austria
Álvaro García-Recuero	Atos Research & Innovation, France
Silvia Giordano	SUPSI, Switzerland
James Gleeson	University of Limerick, UK
Kwang-II Goh	Korea University, Korea
Jesús Gomez-Gardeñes	Universidad de Zaragoza, Spain
Bruno Gonçalves	New York University, USA
Przemyslaw Grabowicz	Max Planck Institute, Germany
Jelena Grujic	Vrije Universiteit Brussel, Belgium
Jean-Loup Guillaume	L3i-Université de la Rochelle, France
Mehmet Gunes	University of Nevada, USA
Emre Guney	Pompeu Fabra University, Spain
Sergio Gómez	Universitat Rovira i Virgili, Spain
Aric Hagberg	Los Alamos National Laboratory, USA
Edwin Hancock	University of York, UK
Chris Hankin	Imperial College London, UK
Jin-Kao Hao	University of Angers, France
Yukio Hayashi	JAIST, Japan
Denis Helic	Graz University of Technology, Austria

Shaun Hendy	The University of Auckland, Australia
Laura Hernandez	CNRS-Université de Cergy-Pontoise, France
Henry Hexmoor	Southern Illinois University, USA
Babak Heydari	Stevens Institute of Technology, USA
Philipp Hoevel	University College Cork, Australia
Seok-Hee Hong	University of Sydney, Australia
Ulrich Hoppe	University Duisburg-Essen, Germany
Agnes Horvat	Northwestern University, USA
Mark Humphries	The University of Manchester, UK
Laurent Hébert Dufresne	University of Vermont, USA
Yuichi Ikeda	Kyoto University, Japan
Roberto Interdonato	CIRAD-UMR TETIS, France
Giulia Iori	City, University of London, UK
Francesco Iorio	Wellcome Sanger Institute, UK
Gerardo Íñiguez	Next Games and Aalto University, Finland
Sarika Jalan	IIT Indore, India
Mahdi Jalili	RMIT University, Australia
Jaroslaw Jankowski	West Pomeranian University of Technology, Poland
Marco Alberto Javarone	Coventry University, UK
Hawoong Jeong	KAIST, South Korea
Tao Jia	Southwest University, Chongqing, China
Di Jin	Tianjin University, China
Hang-Hyun Jo	APCTP, South Korea
Nick Jones	Imperial College, UK
Bertrand Jouve	CNRS, France
Byungnam Kahng	Seoul National University, South Korea
Andreas Kaltenbrunner	NTENT, USA
Rushed Kanawati	Université Paris 13, France
Márton Karsai	ENS de Lyon, France
Mehmet Kaya	Firat University, Italy
Przemysław Kazienko	Wrocław University of Science and Technology, Poland
Dror Kenett	Johns Hopkins University, USA
Yoed Kenett	University of Pennsylvania, USA
Khaldoun Khashanah	Stevens Institute of Technology, USA
Hamamache Kheddouci	University of Lyon 1, France
Hyounghick Kim	Sungkyunkwan University, South Korea
Maksim Kitsak	Northeastern University, USA
Mikko Kivela	Aalto University, Finland
Konstantin Klemm	IFISC (CSIC-UIB), Spain
Peter Klimek	Medical University of Vienna, Austria
Xiangjie Kong	Dalian University of Technology, China
Johan Koskinen	The University of Manchester, UK
Nicolas Kourtellis	Telefonica Research, Spain

Danai Koutra	University of Michigan, USA
Jérme Kunegis	University of Namur, Belgium
Valentina Kuskova	National Research University, Moscow, Russia
Ryszard Kutner	Faculty of Physics, University of Warsaw, Poland
Haewoon Kwak	Qatar Computing Research Institute, Qatar
Lucas Lacasa	Queen Mary University of London, UK
Renaud Lambiotte	University of Oxford, UK
Christine Lärgeron	Université de Lyon, France
Vito Latora	Queen Mary University of London, UK
Stephen Law	University College London, UK
Anna T. Lawniczak	University of Guelph, Canada
Benedicte Le Grand	Université Paris 1 Panthéon-Sorbonne, France
Eric Leclercq	University of Burgundy, France
Deok-Sun Lee	Inha University, South Korea
Sune Lehmann	Technical University of Denmark, Denmark
Elizaveta Levina	University of Michigan, USA
Fabrizio Lillo	University of Bologna, Italy
Gustav Lindmark	Linköping University, Sweden
Yang-Yu Liu	Harvard University, USA
Jianguo Liu	Michigan State University, USA
Giacomo Livan	University College London, UK
Alessandro Lomi	University of Lugano, Switzerland
Alessandro Longheu	University of Catania, Italy
Jinhu Lü	Chinese Academy of Sciences, China
Claudio Lucchese	Ca' Foscari University of Venice, Italy
John C. S. Lui	Chinese University of Hong Kong, Hong Kong
Matteo Magnani	Uppsala University, Sweden
Clemence Magnien	LIP6 (CNRS-Sorbonne Université), France
Fragkiskos Malliaros	University of Paris-Saclay, France
Eric Malmi	Google, USA
Giuseppe Mangioni	University of Catania, Italy
Madhav Marathe	Virginia Tech, USA
Radek Marik	Czech Technical University in Prague, Czech Republic
Andrea Marino	University of Pisa, Italy
Antonio Marques	King Juan Carlos University, Spain
Christoph Martin	Leuphana University of Lüneburg, Germany
Cristina Masoller	Universitat Politècnica de Catalunya, Spain
Rossana Mastrandrea	IMT Institute of Advanced Studies, Italy
Naoki Masuda	University of Bristol, UK
Michael Mathioudakis	University of Helsinki, Finland
Matúš Medo	UESTC, China
Natarajan Meghanathan	Jackson State University, USA
Jörg Menche	CeMM, Austria
Jose Fernando Mendes	University of Aveiro, Portugal

Ronaldo Menezes	University of Exeter, UK
Anke Meyer-Baese	FSU, USA
Radosław Michalski	Wrocław University of Science and Technology, Poland
Bivas Mitra	Indian Institute of Technology Kharagpur, India
Marija Mitrovic Dankulov	Institute of Physics Belgrade, Serbia
Raul Mondragon	Queen Mary University of London, UK
Misael Mongiovì	Universitá di Catania, Italy
Yamir Moreno	University of Zaragoza, Spain
Esteban Moro	Universidad Carlos III de Madrid, Spain
Sotiris Moschoyiannis	University of Surrey, UK
Igor Mozetic	Jozef Stefan Institute, Slovenia
Animesh Mukherjee	Indian Institute of Technology, Kharagpur, India
Tsuyoshi Murata	Tokyo Institute of Technology, Japan
Katarzyna Musial	Bournemouth University, UK
Alessandro Muscoloni	Technische Universität Dresden, Germany
Mirco Musolesi	University College London, UK
Michael Mäs	ETH Zurich, Switzerland
Muaz Niazi	COMSATS Institute of IT, Pakistan
Andrea Omicini	Università di Bologna, Italy
Gergely Palla	MTA-ELTE, Hungary
Pietro Panzarasa	Queen Mary University of London, UCL
Fragkiskos Papadopoulos	Cyprus University of Technology, Cyprus
Symeon Papadopoulos	Information Technologies Institute, Greece
Evangelos Papalexakis	University of California, Riverside, UK
Michela Papandrea	SUPSI, Switzerland
Noseong Park	UNC Charlotte, USA
Juyong Park	KAIST, South Korea
Han Woo Park	YeungNam University, South Korea
Andrea Passarella	IIT-CNR, Italy
Leto Peel	Université Catholique de Louvain, Belgium
Tiago Peixoto	University of Bath, UK
Matjaz Perc	University of Maribor, Australia
Giovanni Petri	ISI Foundation, Italy
Juergen Pfeffer	Technical University of Munich, Germany
Carlo Piccardi	Politecnico di Milano, Italy
Clara Pizzuti	National Research Council of Italy (CNR), Italy
Chiara Poletto	INSERM, Sorbonne Université, France
Nataša Pržulj	UCL, UK
Christian Quadri	University of Milan, Italy
Marco Quaggiotto	ISI Foundation, Italy
Filippo Radicchi	Northwestern University, USA
Jose J. Ramasco	IFISC (CSIC-UIB), Spain
Asha Rao	RMIT University, Australia
Gesine Reinert	University of Oxford, UK

Benjamin Renoust	Osaka University, Japan
Pedro Ribeiro	University of Porto, Portugal
Massimo Riccaboni	IMT Institute for Advanced Studies, Lucca, Italy
Laura Ricci	Universita di Pisa, Italy
Luis M. Rocha	Indiana University Bloomington, USA
Luis E. C. Rocha	University of Greenwich, UK
Francisco Rodrigues	University of São Paulo, Brazil
Luca Rossi	IT University of Copenhagen, Denmark
Camille Roth	Sciences Po, France
Tarik Roukny	Massachusetts Institute of Technology, USA
Amir Rubin	Ben-Gurion University of the Negev, Israel
Meead Saberi	Monash University, Australia
Ali Safari	Universität Erlangen-Nürnberg, Germany
Alessandra Sala	Bell Labs, Ireland
Marc Santolini	Northeastern University, USA
Francisco C. Santos	Universidade de Lisboa, Portugal
Jari Saramäki	Aalto University, Finland
Antonio Scala	Italian National Research Council, Italy
Michael Schaub	Massachusetts Institute of Technology, USA
Maximilian Schich	The University of Texas at Dallas, USA
Rossano Schifanella	University of Turin, Italy
Ingo Scholtes	ETH Zurich, Switzerland
Frank Schweitzer	ETH Zurich, Switzerland
Caterina Scoglio	Kansas State University, USA
Simone Severini	University College London, UK
Rajesh Sharma	University of Tartu, Estonia
Amitabh Sharma	Harvard University, USA
Cynthia Siew	University of Warwick, UK
Tiago Simas	Telefonica Innovation Alpha, Spain
Anurag Singh	NIT Delhi, India
Per Sebastian Skardal	Trinity College, UK
Michael Small	The University of Western Australia, Australia
Tom Snijders	University of Groningen, Germany
Chaoming Song	University of Miami, USA
Massimo Stella	Institute for Complex Systems Simulation, UK
Markus Strohmaier	RWTH Aachen University and GESIS, Germany
Blair D. Sullivan	North Carolina State University, USA
Jie Sun	Clarkson University, USA
Pål Sundsøy	NBIM, Norway
Michael Szell	Central European University, Hungary
Boleslaw Szymanski	Rensselaer Polytechnic Institute, USA
Bosiljka Tadic	Jozef Stefan Institute, Slovenia
Andrea Tagarelli	University of Calabria, Italy
Lucia Tajoli	Politecnico di Milano, Italy
Kazuhiro Takemoto	Kyushu Institute of Technology, Japan

Frank Takes	Leiden University, The Netherlands
Fabien Tarissan	CNRS-ENS Paris-Saclay (ISP), France
Dane Taylor	University at Buffalo, SUNY, USA
Claudio Juan Tessone	Universität Zürich, Switzerland
Claudia Test	GESIS, Germany
My Thai	University of Florida, USA
I-Hsien Ting	National University of Kaohsiung, Taiwan
Olivier Togni	University of Burgundy, France
Vincent Antonio Traag	CWTS, Leiden University, The Netherlands
Ljiljana Trajkovic	Simon Fraser University, Canada
Jan Treur	Vrije Universiteit Amsterdam, The Netherlands
Chi K. Tse	Hong Kong Polytechnic University, Hong Kong
Liubov Tupikina	Ecole Polytechnique, France
János Török	Budapest University of Technology and Economics, Hungary
Stephen Uzzo	New York Hall of Science, USA
Lucas D. Valdez	Boston University, USA
Sergi Valverde	University Pompeu Fabra, Spain
Piet Van Mieghem	Delft University of Technology, The Netherlands
Balazs Vedres	CEU, Hungary
Huijuan Wang	Delft University of Technology, The Netherlands
Xiaofan Wang	Shanghai Jiao Tong University, China
Guanghui Wen	Southeast University, China
Richard Wilson	University of York, UK
Ernst Wit	University of Groningen, Germany
Jinshan Wu	Beijing Normal University, China
Bin Wu	Beijing University, China
Feng Xia	Dalian University of Technology, China
Bo Xu	Dalian University of Technology, China
Gang Yan	Tongji University, Shanghai, China
Taha Yasseri	University of Oxford, UK
Wenwu Yu	Southeast University, China
Zi-Ke Zhang	Hangzhou Normal University, China
Yi-Cheng Zhang	University of Fribourg, Germany
Junfei Zhao	Columbia University, USA
Matteo Zignani	University of Milan, Italy
Antonio G. Zippo	Consiglio Nazionale delle Ricerche, Italy
Vinko Zlatic	Institute Rudjer Boskovic, Croatia
Arkaitz Zubiaga	The University of Warwick, UK
Katharina Anna Zweig	University of Kaiserslautern, Germany
Yves-Alexandre de Montjoye	Imperial College London, UK
Marco van der Leij	University of Amsterdam, The Netherlands

Contents

Link Analysis and Ranking

A New Group Centrality Measure for Maximizing the Connectedness of Network Under Uncertain Connectivity	3
Takayasu Fushimi, Kazumi Saito, Tetsuo Ikeda, and Kazuhiro Kazama	
Walk Prediction in Directed Networks	15
Chuankai An, A. James O’Malley, and Daniel N. Rockmore	
Average-Case Behavior of k-Shortest Path Algorithms	28
Alexander Schickedanz, Deepak Ajwani, Ulrich Meyer, and Paweł Gawrychowski	
Scaling of Random Walk Betweenness in Networks	41
Onuttom Narayan and Iraj Saniee	
Fast Approximated Betweenness Centrality of Directed and Weighted Graphs	52
Angelo Furno, Nour-Eddin El Faouzi, Rajesh Sharma, and Eugenio Zimeo	
Node Ordering for Scalable Network Summarization (or, the Apparent Magic of Word Frequency and Age of Acquisition in the Lexicon)	66
Violet Brown, Xi Chen, Maryam Hedayati, Camden Sikes, Julia Strand, Tegan Wilson, and David Liben-Nowell	
Systematic Biases in Link Prediction: Comparing Heuristic and Graph Embedding Based Methods	81
Aakash Sinha, Rémy Cazabet, and Rémi Vaudaine	
Stability and Similarity in Networks Based on Topology and Nodes Importance	94
Fuad Aleskerov and Sergey Shvydun	

Delusive PageRank in Incomplete Graphs	104
Helge Holzmann, Avishek Anand, and Megha Khosla	
Centrality Maps for Moving Nodes	118
Clément Bertier, Farid Benbadis, Marcelo Dias de Amorim, and Vania Conan	
Core Stratification of Two-Mode Networks	130
Henry Soldano, Sophie Bary, Guillaume Santini, and Dominique Bouthinon	
OTARIOS: OpTimizing Author Ranking with Insiders/Outsiders Subnetworks	143
Jorge Silva, David Aparício, and Fernando Silva	
Cascading Effects of Targeted Attacks on the Power Grid	155
Rounak Meyur, Anil Vullikanti, Madhav V. Marathe, Anamitra Pal, Mina Youssef, and Virgilio Centeno	
Community Structure	
A Memory-Based Label Propagation Algorithm for Community Detection	171
Antonio Maria Fiscarelli, Matthias R. Brust, Grégoire Danoy, and Pascal Bouvry	
Estimating the Similarity of Community Detection Methods Based on Cluster Size Distribution	183
Vinh-Loc Dao, Cécile Bothorel, and Philippe Lenca	
Links in Context: Detecting and Describing the Nested Structure of Communities in Node-Attributed Networks	195
Tobias Hecking and H. Ulrich Hoppe	
Overlapping Communities in Bipartite Graphs	207
Radek Marik and Tomas Zikmund	
Communities as Well Separated Subgraphs with Cohesive Cores: Identification of Core-Periphery Structures in Link Communities	219
Frank Havemann, Jochen Gläser, and Michael Heinz	
Ensemble Clustering for Graphs	231
Valérie Poulin and François Théberge	
A Community-Aware Approach for Identifying Node Anomalies in Complex Networks	244
Thomas J. Helling, Johannes C. Scholtes, and Frank W. Takes	

Is Community Detection Fully Unsupervised? The Case of Weighted Graphs	256
Victor Connes, Nicolas Dugué, and Adrien Guille	
Is it Correct to Project and Detect? Assessing Performance of Community Detection on Unipartite Projections of Bipartite Networks	267
Tristan J. B. Cann, Iain S. Weaver, and Hywel T. P. Williams	
Bayesian Complex Network Community Detection Using Nonparametric Topic Model	280
Ruimin Zhu and Wenxin Jiang	
Detecting Latent Terrorist Communities Testing a Gower's Similarity-Based Clustering Algorithm for Multi-partite Networks	292
Gian Maria Campedelli, Iain Cruickshank, and Kathleen M. Carley	
GLaSS: Semi-supervised Graph Labelling with Markov Random Walks to Absorption	304
Max Glonek, Jonathan Tuke, Lewis Mitchell, and Nigel Bean	
Semi-supervised Overlapping Community Finding Based on Label Propagation with Pairwise Constraints	316
Elham Alghamdi and Derek Greene	
Entropy in Network Community as an Indicator of Language Structure in Emoji Usage: A Twitter Study Across Various Thematic Datasets	328
Ryan Hartman, S. M. Mahdi Seyednezhad, Diego Pinheiro, Josemar Faustino, and Ronaldo Menezes	
TimeRank: A Random Walk Approach for Community Discovery in Dynamic Networks	338
Ilias Sarantopoulos, Dimitrios Papatheodorou, Dimitrios Vogiatzis, Grigorios Tzortzis, and Georgios Paliouras	
Diffusion and Epidemics	
Modeling Topical Information Diffusion over Microblog Networks	353
Kuntal Dey, Hemank Lamba, Seema Nagar, Shubham Gupta, and Saroj Kaushik	
Fast Variables Determine the Epidemic Threshold in the Pairwise Model with an Improved Closure	365
István Z. Kiss, Joel C. Miller, and Péter L. Simon	
Consistent Approximation of Epidemic Dynamics on Degree-Heterogeneous Clustered Networks	376
A. Bishop, I. Z. Kiss, and T. House	

DiffuGreedy: An Influence Maximization Algorithm Based on Diffusion Cascades	392
George Panagopoulos, Fragkiskos D. Malliaros, and Michalis Vazirgiannis	
Predicting Information Diffusion in Online Social Platforms: A Twitter Case Study	405
Kateryna Lytvyniuk, Rajesh Sharma, and Anna Jurek-Loughrey	
Modelling and Analysis of Delayed SIR Model on Complex Network	418
Md Arquam, Anurag Singh, and Rajesh Sharma	
Dynamics On/Of Networks	
A Markov Model for Inferring Flows in Directed Contact Networks	433
Steve Huntsman	
A General Model of Dynamics on Networks with Graph Automorphism Lumping	445
Jonathan A. Ward and John Evans	
EvoNRL: Evolving Network Representation Learning Based on Random Walks	457
Farzaneh Heidari and Manos Papagelis	
Distributed PI Control for Multi-agent Consensus Tracking of Heterogeneous Networks with Heterogeneous Uncertainties	470
Yuting Feng, Zhisheng Duan, and Guanrong Chen	
Time Granularity in System-of-Systems Simulation of Infrastructure Networks	482
Mateusz Iwo Dubaniowski and Hans R. Heinemann	
TTPROF: A Weighted Threshold Model for Studying Opinion Dynamics in Directed Temporal Network	491
Eeti Jain, Anurag Singh, and Rajesh Sharma	
Full-Commanding a Network: The Dictator	505
Clara Grácio, Sara Fernandes, and Luís Mário Lopes	
Biased Dynamic Sampling for Temporal Network Streams	512
Shazia Tabassum and João Gama	
Using Network Reliability to Understand International Food Trade Dynamics	524
Madhurima Nath, Srinivasan Venkatramanan, Bryan Kaperick, Stephen Eubank, Madhav V. Marathe, Achla Marathe, and Abhijin Adiga	

An Approach to Structural Analysis Using Moore-Shannon Network Reliability	537
Madhurima Nath, Yihui Ren, and Stephen Eubank	
Motif Discovery	
New Deterministic Model of Evolving Trinomial Networks	553
Alexander Goryashko, Leonid Samokhine, and Pavel Bocharov	
Counting Multilayer Temporal Motifs in Complex Networks	565
Hanjo D. Boekhout, Walter A. Kosters, and Frank W. Takes	
Quasi-Cliques Analysis for IRC Channel Thread Detection	578
Jocelyn Bernard, Sicong Shao, Cihan Tunc, Hamamache Kheddouci, and Salim Hariri	
Triad-Based Comparison and Signatures of Directed Networks	590
Xiaochuan Xu and Gesine Reinert	
Mining Patterns with Durations from E-Commerce Dataset	603
Mohamad Kanaan and Hamamache Kheddouci	
Network Models	
Relating Emerging Network Behaviour to Network Structure	619
Jan Treur	
Multilevel Network Reification: Representing Higher Order Adaptivity in a Network	635
Jan Treur	
An Exploration of the Network Installation and Recovery Problem with Blackstart Nodes	652
Kayla S. Cummings, Janie L. Neal, Andi Chen, and Tzu-Yi Chen	
Mathematical Analysis of a Network's Asymptotic Behaviour Based on Its Strongly Connected Components	663
Jan Treur	
Random Graph Generators for Hyperbolic Community Structures	680
Saskia Metzler and Pauli Miettinen	
Estimating Personal Network Size with Non-random Mixing via Latent Kernels	694
Swupnil Sahai, Timothy Jones, Sarah K. Cowan, and Tian Zheng	
Forman's Ricci Curvature - From Networks to Hypernetworks	706
Emil Saucan and Melanie Weber	

Mapping Structural Diversity in Networks Sharing a Given Degree Distribution and Global Clustering: Adaptive Resolution Grid Search Evolution with Diophantine Equation-Based Mutations	718
Peter Overbury, István Z. Kiss, and Luc Berthouze	
Specialist Cops Catching Robbers on Complex Networks	731
Shiraj Arora, Abhishek Jain, Yenda Ramesh, and M. V. Panduranga Rao	
Evaluating the Natural Variability in Generative Models for Complex Networks	743
Viplove Arora and Mario Ventresca	
Multilayer Networks	
Py3plex: A Library for Scalable Multilayer Network Analysis and Visualization	757
Blaž Škrlj, Jan Kralj, and Nada Lavrač	
Morphogenesis of Complex Networks: A Reaction Diffusion Framework for Spatial Graphs	769
Michele Tirico, Stefan Balev, Antoine Dutot, and Damien Olivier	
Multilayer Network Model of Movie Script	782
Youssef Mourchid, Benjamin Renoust, Hocine Cherifi, and Mohammed El Hassouni	
Effects of Interaction and Learning Distance on Cooperation in Evolutionary Games on a Multiplex Network	797
Yasuyuki Nakamura, Koichi Yasutake, Keiya Ando, and Takahiro Tagawa	
Resilience and Control	
A Genetic Algorithm for Enhancing the Robustness of Complex Networks Through Link Protection	807
Clara Pizzuti and Annalisa Socievole	
Numerical Assessment of the Percolation Threshold Using Complement Networks	820
Giacomo Rapisardi, Guido Caldarelli, and Giulio Cimini	
Optimal Control Rules for Random Boolean Networks	828
Matthew R. Karlsen and Sotiris K. Moschoyiannis	
Robustness Through Regime Flips in Collapsing Ecological Networks	841
Suresh Babu and Gitanjali Yadav	
Enhancing Synchronization Stability in Complex Networks with Probabilistic Natural Frequencies	854
K. Y. Henry Tsang, Bo Li, and K. Y. Michael Wong	

Identifying Vulnerable Nodes to Cascading Failures: Centrality to the Rescue	866
Richard J. La	
Computational Aspects of Fault Location and Resilience Problems for Interdependent Infrastructure Networks	879
Madhav V. Marathe, S. S. Ravi, Daniel J. Rosenkrantz, and Richard E. Stearns	
Author Index	891

Link Analysis and Ranking



A New Group Centrality Measure for Maximizing the Connectedness of Network Under Uncertain Connectivity

Takayasu Fushimi^{1(✉)}, Kazumi Saito^{2,3}, Tetsuo Ikeda⁴, and Kazuhiro Kazama⁵

¹ School of Computer Science, Tokyo University of Technology, 1404-1 Katakuramachi, Hachioji-shi, Tokyo 192-0982, Japan
takayasu.fushimi@gmail.com

² Faculty of Science, Kanagawa University, 2946 Tsuchiya, Hiratsuka-shi, Kanagawa 259-1293, Japan
k-saito@kanagawa-u.ac.jp, kazumi.saito@riken.jp

³ Center for Advanced Intelligence Project, RIKEN, 1-4-1 Nihonbashi, Chuo-ku, Tokyo 103-0027, Japan

⁴ School of Management and Information, University of Shizuoka, 52-1 Yada, Suruga-ku, Shizuoka 422-8526, Japan
t-ikeda@u-shizuoka-ken.ac.jp

⁵ Faculty of Systems Engineering, Wakayama University, 930 Sakaedani, Wakayama-shi, Wakayama 640-8510, Japan
kazama@ingrid.org

Abstract. In this paper, we propose a new centrality measure for the purpose of estimating and recommending installation sites of evacuation facilities that many residents can reach even in the situation where roads are blocked by natural disasters. In the proposed centrality, we model the probabilistically occurring road blockage by the link cut of the graph, and quantify the degree of connectivity of each node by the expectation value of the number of reachable nodes under uncertain connectivity. In a large-scale network, since the number of combinations of disconnecting links is enormous, it is difficult to strictly calculate the expected value. Therefore, approximate connectivity is calculated by an efficient algorithm based on simulation. Furthermore, in order to estimate multiple installation sites, we propose a method of defining and maximizing the degree of connectivity for a node group rather than a single node. From this, it can be expected that duplication of nodes covered by the extracted nodes can be eliminated, so that a practical candidate site of evacuation facility can be estimated. We evaluate the effectiveness and efficiency of the proposed method compared with the method based on distance between nodes and the method based on link density, using real road networks.

Keywords: Road network · Community extraction · Group centrality measure · Connectivity

1 Introduction

In Japan and all over the world, many disasters are occurring like typhoons or hurricanes, river flooding, cliff collapse, earthquakes, tsunami, volcanic eruptions and so forth. In the event of such a disaster, by quick evacuation of residents to the neighborhood evacuation facility, the damage can be reduced. The installation site of evacuation facility needs to be accessible from neighboring residents. Furthermore, in the event of a disaster, due to flooding of the river, landslides, collapse of houses and telegraph pillars, there is a high possibility that the road is blocked. Even under such circumstances, it is desirable that evacuation centers themselves are not isolated, and many neighboring residents can reach them.

This paper focuses on the graph structure of the road network, and proposes a method for estimating and recommending appropriate evacuation facility location. In recent years, studies taking a complex network analysis approach on road networks have been conducted [4, 9, 11]. As a facility location problem on the graph, k -median or k -center problems have been widely studied [7], and some efficient approximate algorithms have been proposed [6, 13]. Also, due to road blockage caused by disasters, the area is divided, and some areas may be isolated. This corresponds to region division by cutting links in the graph, that is, community extraction. The GN method which cuts off links with high edge betweenness and extracts communities is famous [5]. Other community extraction methods including CNM method, spectral clustering, Louvain method, deep community detection, intend to extract densely connected node groups [1–3, 8]. As for road networks, however, since the degree distribution is almost uniform and the highest degree is not so high compared with general networks, it is difficult to apply these methods to the road networks.

In this paper, in a situation where the graph is broken into several connected components by link disconnection, we attempt to extract nodes with a large number of reachable nodes, that is, nodes with high connectivity to neighboring nodes. Therefore, we propose connectedness centrality which defines the degree of connectivity of each node by expectation value of the number of reachable nodes under stochastic occurrence of link disconnection. Closeness centrality, which is an existing measure, quantifies accessibility from the viewpoint of distance, however does not consider situations such as when the collapse of the bridge caused by the rebellion of the river prevents you from going to the area beyond the river. Unlike the method using link density and distance to other nodes, nodes with many paths to other nodes are extracted by connectedness centrality. This is a property different from the methods that focus on closeness of the nodes and the denseness of the link. Therefore, although closeness centrality may extract nodes which likely to be inaccessible from other nodes or be easily isolated, the proposed centrality would not extract such nodes because of considering not only the distance but also the number of routes.

The rest of this paper is organized as follows. In Sect. 2, we summarize the existing research related to the proposed method, In Sect. 3, the detail of the proposed centrality and will be explained. In Sects. 4 and 5, experimental eval-

ations using actual road data and the results are discussed. Finally, we summarize this paper and discuss future issues in Sect. 6.

2 Related Work

Studies taking a complex network analysis approach on road networks have been conducted [9, 11], and centrality measure is one of the most widely-used techniques not only for social networks but also for spatial networks [4]. By using closeness centrality, which is based on the distance to other nodes, we can extract nodes that are easy to reach from other nodes. However closeness centrality does not consider situations such as when the collapse of the bridge caused by the rebellion of the river prevents you from going to the area beyond the river. As a facility location problem, a closeness-centrality-based method has been proposed [13]. Although the method can be solve the location of a single facility very fast, cannot deal with the problem of multiple facilities.

As another line of work on complex networks, methods for community detection or core extraction have been widely studied [1–3, 5, 8, 10, 12]. As mentioned earlier, since almost all of these methods focus on link density and node degree, it is difficult to straightforwardly adopt these methods to spatial networks like urban streets, whose degrees are restricted to relatively small numbers. Moreover, although spectral clustering [8] and deep community detection [2] are based on link cuttings similar to our method, it is often impossible to calculate the Fiedler vector of a spatial network because of abnormal smallness of eigen-gaps.

3 Proposed Measure

In order to estimate adequate installation sites of evacuation facilities, we propose a node ranking measure called connectedness centrality, and its efficient approximate algorithm based on simulations. Furthermore, to select multiple installation sites, we propose group-connectedness centrality by extending the target of connectedness centrality from each node to node groups.

3.1 Connectedness Centrality

First, we propose a new centrality measure called connectedness centrality. Let $G = (\mathcal{V}, \mathcal{E})$ be the graph structure of a given spatial network, and for each link $e \in \mathcal{E}$, we consider a link connection probability $p(e; s)$, which is determined according to some model such as a road blockage model based on geographical properties, where s is a parameter, just like an inverse of magnitude of earthquake, to control the probability $p(e; s)$, and we set s in the range of $0 \leq s \leq 1$ for our convenience. For each link $e \in \mathcal{E}$, let $x(e)$ be a random variable expressing the link connectivity, *i.e.*, $x(e) = 1$ if the link e is connected; otherwise $x(e) = 0$, where note that $p(x(e) = 1; s) = p(e; s)$. Then, by suitably arranging these random variables, we can construct an indicator vector expressed as

$\mathbf{x} = (\dots, x(e), \dots) \in \{0, 1\}^{|\mathcal{E}|}$, whose total number of possible instantiations amounts to $|\{0, 1\}|^{|\mathcal{E}|} = 2^{|\mathcal{E}|}$. For each instance of the indicator vector \mathbf{x} , we can obtain the corresponding graph $G_{\mathbf{x}} = (\mathcal{V}, \mathcal{E}_{\mathbf{x}})$, where $\mathcal{E}_{\mathbf{x}} = \{e \mid e \in \mathcal{E}, x(e) = 1\}$. In this paper, assuming a basic model based on independent Bernoulli trials for all the links, with respect to each graph $G_{\mathbf{x}}$ obtained from \mathbf{x} , we can compute its occurrence probability as follows:

$$q(\mathbf{x}; s) = q(G_{\mathbf{x}}; s) = \prod_{e \in \mathcal{E}} p(e; s)^{x(e)} (1 - p(e; s))^{1-x(e)}. \quad (1)$$

However, we should emphasize that our approach is not necessarily restricted to the model defined in Eq. (1).

After decomposing $G_{\mathbf{x}}$ into connected components, we compute the size of each connected component, as the number of nodes belonging to the component, and let $c(v; G_{\mathbf{x}})$ be the set of the nodes belonging to the connected component in which the node $v \in \mathcal{V}$ is included, where $c(u; G_{\mathbf{x}}) = c(v; G_{\mathbf{x}})$ if the nodes u and v belong to the same connected components. In this study, under a given stochastic model of link connection, we propose to define our connectedness centrality of node $v \in \mathcal{V}$ by the expected size of the connected component in which v is included. More specifically, for each node $v \in \mathcal{V}$, we quantify our connectedness centrality by the following expectation:

$$cnc_1(v) = \int_0^1 \sum_{\mathbf{x} \in \{0, 1\}^{|\mathcal{E}|}} |c(v; G_{\mathbf{x}})| q(\mathbf{x}; s) r_1(s) ds, \quad (2)$$

where $r_1(s)$ stands for a probability distribution with respect to the parameter s . For instance, it can be used to express a fact that small earthquakes occur frequently, but huge ones are quite rare.

3.2 Solution Algorithm

In this paper, we assume that each link connection probability is the same, *i.e.*, $p(e; s) = p(s) = s$, and propose to compute the integration of s by the summation of $H + 1$ equal interval points. Here note that for the h -th point ($0 \leq h \leq H$), the link connection probability is set to $p(h) = h/H$. Moreover, we propose to compute the summation of $2^{|\mathcal{E}|}$ times by Monte Carlo simulation of J times. Let $G_{(h,j)} = (\mathcal{V}, \mathcal{E}_{(h,j)})$ be a graph obtained by the j -th simulation ($1 \leq j \leq J$) at the h -th point, then we can transform our connectedness centrality $cnc_1(v)$ defined in Eq. (2) into the following one:

$$cnc_2(v) = \frac{1}{HJ} \sum_{h=1}^H \sum_{j=1}^J |c(v; G_{(h,j)})| r_2(h), \quad (3)$$

where $r_2(h) = r_1(h/H) / \sum_{h'=1}^H r_1(h'/H)$. Evidently, by setting both H and J to sufficiently large values, it is naturally expected that $cnc_2(v)$ defined in Eq. (3) can be a reasonably accurate estimate to $cnc_1(v)$ defined in Eq. (2). However,

when straightforwardly computing $cnc_2(v)$ for every $v \in \mathcal{V}$ for large H and J , we need a large computational load because its computational complexity becomes $O(HJ(N + L))$, where $N = |\mathcal{V}|$ and $L = |\mathcal{E}|$ respectively stand for the numbers of nodes and links for a given network. Note that the computational complexity of decomposing a graph into its connected components is $O(N + L)$, and during this process, we can simultaneously compute their sizes.

Below we propose another reasonably accurate estimate referred to as $cnc_3(v)$, instead of $cnc_2(v)$, together with an effective algorithm whose computational complexity becomes $O(J(L + N \log N))$, rather than $O(HJ(N + L))$. In our proposed algorithm, from the initial state that all links are disconnected and thus all nodes are isolated in the setting $p(0) = 0$, we repeatedly add a randomly selected link one by one until the final state that all the original links are connected in the setting $p(H) = 1$, and during this process, we attempt to efficiently compute the expected size of connected component for each node $v \in \mathcal{V}$, by focusing on the difference between the graphs caused by adding only one link. Here note that this procedure corresponds to employing the setting of $H = L$. More specifically, for the j -th simulation, we assign a random order to each link $e \in \mathcal{E}$, denoted by $e^{(h,j)}$, where we also use $h \in \{1, \dots, H\}$ to express the order that the link becomes connected one. Then, by considering a graph defined by $G^{(h,j)} = (\mathcal{V}, \mathcal{E}^{(h,j)})$, where $\mathcal{E}^{(h,j)} = \{e^{(h',j)} \in \mathcal{E} \mid h' \leq h\}$, we can derive the following connectedness centrality $cnc_3(v)$:

$$cnc_3(v) = \frac{1}{JH} \sum_{j=1}^J \sum_{h=1}^H |c(v; G^{(h,j)})| r_2(h). \quad (4)$$

Here we should emphasize that with respect to graph $G^{(h,j)}$, the maximum likelihood estimate of the common connection probability becomes $\hat{p} = h/H$. Thus, for a reasonably large J , we can expect to obtain reasonably comparable estimates from both of the two formulae, $J^{-1} \sum_{j=1}^J c(v; G^{(h,j)})$ and $J^{-1} \sum_{j=1}^J c(v; G^{(h,j)})$. This means that our two measures, $cnc_2(v)$ and $cnc_3(v)$, can also be comparably precise estimates.

Below we show some details of our proposed algorithm together with its computational complexity. In the initial state with no link, we set that every node belongs to an individually different component by assigning unique component number $n(v) \in \{1, \dots, N\}$ to each node $v \in \mathcal{V}$. When a new link denoted by $e^{(h,j)} = (x, y)^{(h,j)}$ is added, we can proceed to the next link if the nodes x and y belong to the same connected component; otherwise, we need to change the component number of nodes belonging to one component. More specifically, by assuming $|c(x; G^{(h,j)})| \leq |c(y; G^{(h,j)})|$ without loss of generality, we propose that the component number with smaller size is changed to that with larger one by setting $n(z) \leftarrow n(x)$ for each $z \in c(y; G^{(h,j)})$. Evidently, for each link addition, the number of nodes whose component number is changed never exceeds $N/2$. Thus, during the whole link additions, the computational complexity of these renumbering processes becomes $O(N \log N)$.

Let $cnc_3^{(j,h)}(v)$ be the partial summation of $|c(v; G^{(h',j)})|$ until $h' = h$ for the j -th simulation defined by

$$cnc_3^{(j,h)}(v) = \sum_{h'=1}^h |c(v; G^{(h',j)})| r_2(h'). \quad (5)$$

Now, suppose that when a new link $e^{(h,j)} = (x, y)^{(h,j)}$ was added at the h -th step, the nodes x and y turn to belong to the same connected component for the first time. Then, for arbitrary $h' \geq h$, since $c(x; G^{(h',j)}) = c(y; G^{(h',j)})$, we can obtain the following relation:

$$cnc_3^{(j,h')}(x) - cnc_3^{(j,h')}(y) = cnc_3^{(j,h-1)}(x) - cnc_3^{(j,h-1)}(y). \quad (6)$$

Thus, by maintaining the partial summation $cnc_3^{(j,h')}(x)$ for a head node x of each connected component, and keeping the difference values such as $cnc_3^{(j,h-1)}(x) - cnc_3^{(j,h-1)}(y)$ for the other nodes in the component, we can obtain the final summation values such as $cnc_3^{(j,H)}(y)$ by using Eq. (6). Here note that the computational complexity of obtaining $cnc_3^{(j,H)}(v)$ for every $v \in \mathcal{V}$ is $O(N)$, that of updating these difference values is $O(N \log N)$ because these updates can be done together with the above node renumbering processes. Thus, since we need to shuffle and examine all of the links at the j -th simulation, the total computational complexity of our proposed algorithm becomes $O(J(L + N \log N))$.

3.3 Group-Connectedness Centrality

For extracting multiple installation sites of evacuation facilities, it is not enough just to select some nodes with high connectivity scores, because those nodes are likely to be located near each other and reachable nodes overlap. To overcome this shortcoming, we enhance the notion of our connectedness centrality, called group-connectedness centrality.

In group-connectedness centrality, connectivity of the node set \mathcal{R} is defined as follows:

$$cnc_1(\mathcal{R}) = \int_0^1 \sum_{\mathbf{x} \in \{0,1\}^{|\mathcal{E}|}} |c(\mathcal{R}; G_{\mathbf{x}})| q(\mathbf{x}; s) r_1(s) ds, \quad (7)$$

where $c(\mathcal{R}; G_{\mathbf{x}}) = \bigcup_{r \in \mathcal{R}} c(r; G_{\mathbf{x}})$ stands for the number of reachable nodes from whichever of \mathcal{R} . For evacuation facility location problem, it is desirable that a node group with the largest connectivity is extracted. Hereafter, we refer to the selected node as representative node.

Similarly to the connected centrality, we compute the integration of s by the summation of $H + 1$ equal interval points and set $r(s)$ to be a uniform distribution.

$$cnc_3(\mathcal{R}) = \frac{1}{JH} \sum_{j=1}^J \sum_{h=1}^H |c(\mathcal{R}; G^{(h,j)})| r_2(h). \quad (8)$$

In this paper, we employ a greedy algorithm to extract a node set \mathcal{R} that maximize the objective function Eq. 8. In our greedy algorithm, we select representative node v one by one which maximizes the following marginal gain by fixing already selected nodes \mathcal{R} :

$$MG(v; \mathcal{R}) = cnc_3(\mathcal{R} \cup \{v\}) - cnc_3(\mathcal{R}) = \frac{1}{JH} \sum_{j=1}^J \sum_{h=1}^H mg(v; \mathcal{R})^{(h,j)} r_2(h), \quad (9)$$

where $mg(v; \mathcal{R})^{(h,j)} = |c(\mathcal{R} \cup \{v\}; G^{(h,j)}) \setminus c(\mathcal{R}; G^{(h,j)})|$ stands for the increment of the number of reachable nodes when node v , which is a candidate for the representative node, is added to \mathcal{R} . The greedy algorithm can be naturally applied to the facility location problem which is difficult to change the already built facilities. In the process of extracting K representative nodes, the k -th representative is selected according to $\hat{r}_k \leftarrow \arg \max_{v \in \mathcal{V} \setminus \mathcal{R}_{k-1}} MG(v; \mathcal{R}_{k-1})$, and added to

the representative node set $\mathcal{R}_k \leftarrow \mathcal{R}_{k-1} \cup \{\hat{r}_k\}$. Thus, the total computational complexity of group-connected centrality becomes $O(KJ(L + N \log N))$.

After selecting K representative nodes, each of which is installation sites of evacuation facility, each of the remaining nodes is assigned to the cluster where a representative node with the highest connectivity exists. Suppose that when a new link is added at h -th step of the j -th simulation then a node v turn to belong to the same connected component with a representative node r , degree of connectivity of the node v and r is defined as $f(v, r)^{(j)} = 1 - h/H$. Therefore, the degree of connectivity in all the J times simulations is $F(v, r) = J^{-1} \sum_{j=1}^J f(v, r)^{(j)}$. For each of the remaining nodes, we assign one cluster of a representative node with the highest connectivity as follows:

$$\mathcal{V}^{(k)} = \{v \in \mathcal{V}; r_k = \arg \max_{r \in \mathcal{R}} F(v, r)\}.$$

In the final stage of a simulation, when most of all representative nodes would belong to the same connected component and the degree of connectivity between a remaining node v and each representative node is equal, the node v is assigned to the cluster of the closest representative node in terms of graph distance. As a result, our clustering algorithm is summarized as follows:

1. Extraction of representative nodes;
2. Assignment of clusters to the general nodes;

Hereafter, we refer to this method as CNC.

4 Experimental Settings

To evaluate our CNC method from the viewpoints of effectiveness and efficiency, we conducted several experiments using actual datasets. In our experiments, the number of simulations J is set to 10,000.

We used the following four prefectures extracted from Digital Road Map (DRM) data: Tokyo, Kanagawa, Shizuoka, and Ibaraki. We extracted all the intersections and roads from the DRM data of each prefecture, constructed a spatial network with the intersection as the node and the road between the intersections as the link. In order to simplify our analyses, we deleted nodes used for representing curve-segments of highways by directly connecting both sides of the deleted ones. As a result, each network has 340919, 295151, 110925, and 172892 nodes, and 485858, 402576, 162322, and 263075 links, respectively.

In our experiments, we employed the following two existing methods for comparison.

- Distance based clustering

Closeness centrality is one of the widely-used existing centrality measures and based on the shortest path length between each pair of nodes. Similarly to our group-connectedness centrality, closeness centrality can be extended to group-closeness centrality by generalizing the target from a node to a group of nodes as follows:

$$clc(\mathcal{R}) = \sum_{v \in V} \min_{r \in \mathcal{R}} d(v, r), \quad (10)$$

where $d(v, r)$ stands for the shortest path length between a node v and a representative r . To minimize the objective function Eq. 10, we employ a greedy algorithm as well as our group-closeness centrality. Extracting a node set \mathcal{R} consisting of K representatives and assigning each of the remaining nodes to a cluster based on the shortest path length to the representative node is equivalent to conducting K -medoids clustering based on graph distance. Hereafter, we refer to this method as CLC.

- Density based clustering

Girvan and Newman proposed a community detection method, GN method [5], which removes some links according to the edge betweenness centrality, and treat connected components of the remaining graph as communities. However, the GN method takes much computation time for large graphs, then an accelerated method, CNM method, was proposed by directly optimizing the modularity measure [3]. Hereafter, we refer to this method as CNM. The CNM method divides all nodes into K clusters without extracting representative nodes.

5 Experimental Results

5.1 Visualization of Representatives and Their Clusters

First, we qualitatively evaluate our method, CNC, compared with two existing method, CLC and CNM, by visualizing clustering results shown in Fig. 1(b), (c), and (d). Due to the limitation of the number of pages, only the results of Shizuoka network are shown, but similar results were obtained in other networks. In Fig. 1(b) and (c), the representative nodes are described as star nodes, and the

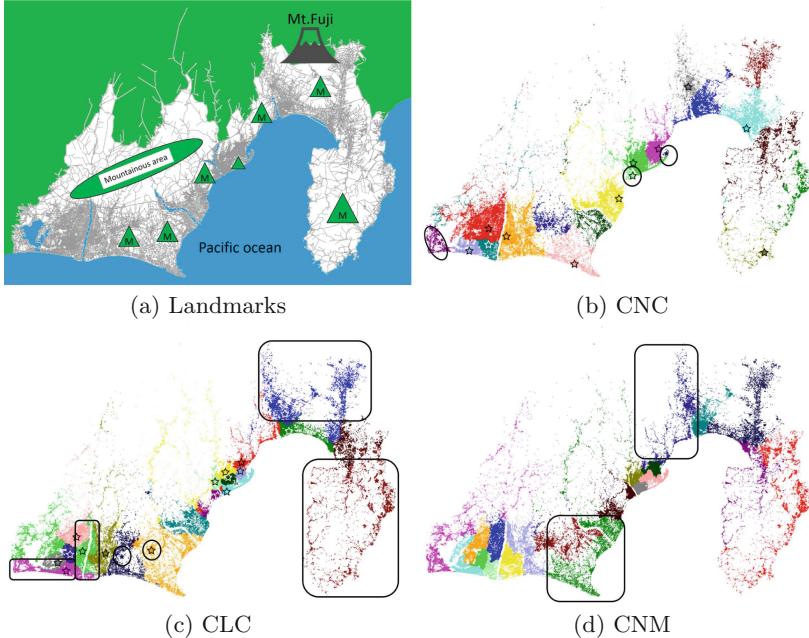


Fig. 1. Visualization of Shizuoka network ($K = 20$)

colors of other nodes stand for their assigned clusters. Figure 1(a) depicts natural environment in and around the Shizuoka network like mountains, and rivers, and these can be constraints in evacuations. From Fig. 1(b), we can observe that our CNC method extracts representatives (star nodes) avoiding mountainous areas and divides nodes into clusters roughly according to the natural environment. Especially for some representatives (surrounded by circles in Fig. 1(b)), although they are located in lakeside, streamside, and peninsula areas, and easy to be isolated, many nodes exist around them. Therefore, evacuation facilities would be needed in these areas. On the other hand, in the result of the CLC and the CNM methods, several clusters (surrounded by squares in Fig. 1(c) and (d)) range across rivers and mountains. In Fig. 1(c), two circled representative nodes are located in the mountainous areas, thus residents in these clusters may be difficult to get to them in case of disaster.

5.2 Stability with Respect to the Number of Simulations

Next we evaluate the stability of our CNC method with respect to the number of simulations J . In this experiment, CNC calculation was performed 10 times by changing the number of simulations $J = 10, 100, 1000, 10,000$. We treat a result with $J = 100,000$ as a converged result and compare the results in terms of similarities of representative nodes and clusters. Figure 2(a) depicts the Minimum Matching Distance (MMD) between representatives extracted at each CNC

calculation, which is calculated as follows:

$$MMD(J) = \frac{1}{K} \sum_{k=1}^K \min_{1 \leq h \leq K} e(r(k), r(h; J)) + \frac{1}{K} \sum_{k=1}^K \min_{1 \leq h \leq K} e(r(k; J), r(h)),$$

where $r(k)$ and $r(k, J)$ respectively stand for the k -th representative node extracted by a CNC calculation with 100,000 and J simulations, and $e(a, b)$ is the Euclidean distance between locations of two representatives a and b . In Fig. 2(a), red errorbar consists of mean, max, and min values of $MMD(J)$ of 10 times with respect to the number of simulations J taken as the horizontal axis, and black solid and dotted line respectively stands for the average Euclidean distance between all node pairs $(u, v) \in \mathcal{V} \times \mathcal{V}$ which satisfy the graph distance $d(u, v) = 1$ or $d(u, v) = 5$. From Fig. 2(a), for all the networks, the average Minimum Matching Distance at $J = 10,000 \ll N, L$ takes a small value, which means that almost same nodes are extracted as representatives.

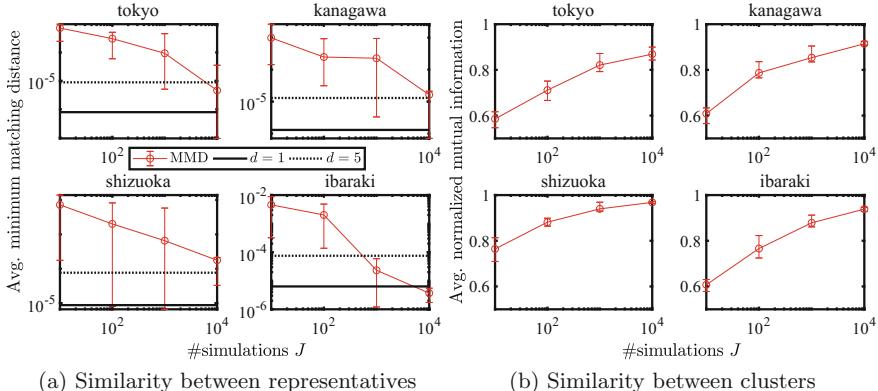


Fig. 2. Stability w.r.t. the number of simulations

Figure 2(b) depicts the Normalized Mutual Information (NMI) between clusters assigned to the nodes at each CNC calculation. From Fig. 2(b), for all the networks, the average Normalized Mutual Information at $J = 10,000$ takes a large value, i.e. $NMI(J) \simeq 0.9$, which means that almost same clustering results can be obtained. From these results, we conclude that we can obtain stable results with the substantially small number of simulations compared with the numbers of nodes N and links L .

5.3 Reachability Under Link Cutting

Next, we quantitatively evaluate the extracted representatives in terms of reachability to the representative nodes under the situation of link disconnection modeling road blockage. In this experiment, we cut off a certain ratio of links

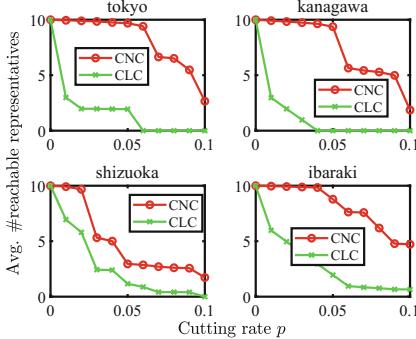


Fig. 3. Reachability to the representatives

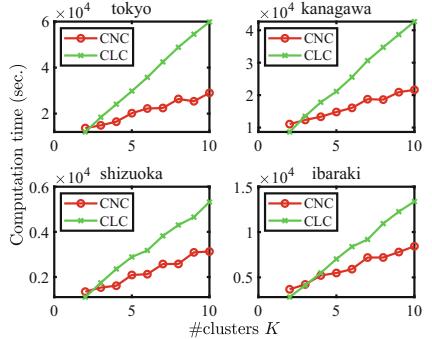


Fig. 4. Computation time

with high edge betweenness, then we count the reachable representatives along links from each node. Figure 3 shows the average number of reachable representatives with respect to the cutting rate p taken as the horizontal axis. From Fig. 3, for all the networks, the average number of reachable representatives extracted by our CNC method is larger than those by the CLC method. Especially for the Tokyo, Kanagawa, and Ibaraki networks, even though 5% of all links are damaged, residents can reach almost all evacuation facilities on average.

5.4 Computation Time

Finally, we evaluate our method from the viewpoint of the computation time. Here, we recall that the number of simulations J is set to 10,000. In Fig. 4, the horizontal and vertical axes stand for the number of clusters K and computation time, respectively. From Fig. 4, we can observe that the difference between the computation time of the CNC and the CLC methods increase as the number of clusters K becomes large. Thus our CNC method can efficiently estimate the installation sites of evacuation facilities.

6 Conclusion

In this paper, for estimating installation sites of evacuation facilities based on road network structure, we proposed a novel centrality measure, connectedness centrality, and an efficient algorithm to calculate the degree of connectivity of each node. Furthermore, to extract multiple installation sites, we extended it to group-connectedness centrality with a greedy algorithm. From experiments using actual road networks, we confirmed the effectiveness and efficiency of our method. In the future, we evaluate our method in the situation that actual evacuation facilities are already given. Furthermore, to comply with more practical problems, we take a population around each node into consideration, and to

reveal the nature of our proposed centrality, we conduct further comparisons with existing centrality measures and community extraction methods.

Acknowledgement. This work was supported by JSPS Grant-in-Aid for Scientific Research (No.17H01826).

References

1. Blondel, V.D., Guillaume, J.L., Lambiotte, R., Lefebvre, E.: Fast unfolding of communities in large networks. *J. Stat. Mech. Theory Exp.* **2008**(10), P10,008 (2008)
2. Chen, P.Y., Hero, A.O.: Deep community detection. *IEEE Trans. Signal Process.* **63**(21), 5706–5719 (2015)
3. Clauset, A., Newman, M.E.J., Moore, C.: Finding community structure in very large networks. *Phys. Rev. E* **70**(6), 066,111+ (2004). <https://doi.org/10.1103/PhysRevE.70.066111>
4. Crucitti, P., Latora, V., Porta, S.: Centrality measures in spatial networks of urban streets. *Phys. Rev. E* **73**(3), 036,125+ (2006)
5. Girvan, M., Newman, M.E.J.: Community structure in social and biological networks. *Proc. Natl. Acad. Sci.* **99**(12), 7821–7826 (2002). <https://doi.org/10.1073/pnas.122653799>
6. Jain, K., Mahdian, M., Saberi, A.: A new greedy approach for facility location problems. In: Proceedings of the Thiry-Fourth Annual ACM Symposium on Theory of Computing, STOC 2002, pp. 731–740. ACM, New York (2002). <https://doi.org/10.1145/509907.510012>
7. Kariv, O., Hakimi, S.L.: An algorithmic approach to network location problems. ii: the p-medians. *SIAM J. Appl. Math.* **37**(3), 539–560 (1979)
8. von Luxburg, U.: A tutorial on spectral clustering. *Stat. Comput.* **17**(4), 395–416 (2007)
9. Montis, D.A., Barthelemy, M., Chessa, A., Vespignani, A.: The structure of interurban traffic: a weighted network analysis. *Environ. Plan. B Plan. Des.* **34**(5), 905–924 (2007)
10. Palla, G., Derényi, I., Farkas, I., Vicsek, T.: Uncovering the overlapping community structure of complex networks in nature and society. *Nature* **435**, 814–818 (2005)
11. Park, K., Yilmaz, A.: A social network analysis approach to analyze road networks. In: Proceedings of the ASPRS Annual Conference 2010 (2010)
12. Seidman, S.B.: Network structure and minimum degree. *Soc. Netw.* **5**(3), 269–287 (1983)
13. Tabata, K., Nakamura, A., Kudo, M.: An efficient approximate algorithm for the 1-median problem on a graph. *IEICE Trans. Inf. Syst.* **E100.D**(5), 994–1002 (2017). <https://doi.org/10.1587/transinf.2016EDP7398>



Walk Prediction in Directed Networks

Chuankai An¹(✉), A. James O’Malley², and Daniel N. Rockmore^{3,4}

¹ Department of Computer Science, Dartmouth College, Hanover, NH 03755, USA
chuankai@cs.dartmouth.edu

² Department of Biomedical Data Science and the Dartmouth Institute of Health Policy and Clinical Practice in the Geisel School of Medicine, Dartmouth College, Lebanon, NH 03756, USA

³ Departments of Computer Science and Mathematics, Dartmouth College, Hanover, NH 03755, USA

⁴ External Faculty of the Santa Fe Institute, Santa Fe, NM 87501, USA

Abstract. In this paper we consider the problem of directed and walk-specific spread of information in complex social networks. Traditional models tend to explain “explosive” information spreading on social media (e.g., Twitter) – a broadcast or epidemiological kind of model with a focus on the sequence of newly “infected” nodes generated from a source node to multiple targets. However, the process of (*single-track*) information flow, wherein there is a node-by-node (and not necessarily a newly visited node) trajectory of information transfer is also a common phenomenon. A key example of interest is the sequence of physician visits of a given patient (*a referral sequence*) in a physician network, wherein the patient is a carrier of information about treatment or disease. With this motivation in mind, we present a Bayesian Personalized Ranking (BPR) model to predict the next node on a walk of a given network navigator using features derived from network analysis. This problem is related to but different from the well-studied problem of link prediction. We apply our model to data from several years of U.S. patient referrals. We present experiments showing that the adoption of network-based features in the BPR framework improves hit-rate and mean percentile rank for next-node prediction.

Keywords: Information walk prediction · Network measures
Bayesian personalized ranking · Patient referral network

1 Introduction

A social network provides a natural framework for the modeling of the interactions between people that might provide a substrate for information flow. In this model, the person-to-person transmission of information becomes a *path* in the network, or more properly a *walk*, as it is possible that the information could loop back to persons (nodes) that have already been visited. This person-to-person or, more generally, node-by-node model of information transmission –

a “single-track model” – is more akin to various kinds of disease transmission models and different from the usual “broadcast” or “diffusive” models that are generally found in the digital realm. The single-track property limits the kind of explosive and exponential spreading from a source node to affected nodes produced after rounds of diffusion.

The single-track transmission model gives rise to the subject of this paper: the problem of *next visit prediction* for a walker in a network. Our initial motivation comes from the study of patient referral networks [1], in which physicians are linked (possibly in a directed and weighted fashion) if a patient has visited both physicians within some (short) time interval. Patterns of “referral sequences” are of interest for understanding information sharing among physicians. Furthermore, accurate prediction of a patient’s next visit could be useful for the efficient deployment of supplies or resources for treatment. As per above, the sequence of contacts may very well have multiple visits to a single node and the possibility of a multiplicity of visits is important and meaningful. Other examples in different contexts include tourist visits to locations of interest, shopper visits to items or stores, or the career histories of workers in an economy. Note that in all of these instances the networks of nodes that can be visited are in general built using a history of transitions from one node to another, the existence and number of which will have implications for the possibility of a future visit from one node to the next. Indeed at any point a walker might be the first to ever transition between two nodes – meaning that in the network, these nodes are not linked. Thus, a more precise framing is the problem of *visit prediction* for a walker in a *state space*. As we assume a context of information sharing and application-related metadata, we call these *information walks* and the related problem one of *information walk prediction*. The several cases mentioned above suggest that not only the last node on the observed walk, but the entire walk up to that point will affect the choice of the next node.

Our work here builds on our previous work analyzing the U.S. Patient Referral Network (see [1] and [2]). Herein, using both traditional network measures as well as others developed ([1] and [2]) we define a preference/attractiveness score between the current node of an observed information walk and any candidate next node. This score considers features of both the observed information walk and related nodes. Using this preference score, we adopt the Bayesian Personalized Ranking (BPR) framework to derive an objective function that enables more accurate estimation of the choice of the next node. Our main example is the (large) U.S. Patient Referral Network. The positional features in the national network of a node enable the prediction between two nodes which are not directly connected in history. The results suggest that information walk-based features enhance the BPR framework to the extent that it outperforms LSTM (long short-term memory) and other baseline methods of link prediction. The incorporation of network measures with other features from metadata are key to our progress on the walk prediction problem.

2 Related Works

Related, although different from the focus of this paper is the problem of *link prediction* in networks, wherein given an evolving network at some fixed time, the link structure at future time is predicted (see [3] and [4]). This is in turn related to the problem of estimating *missing links* [5]. Traditional link prediction methods (e.g., [6] and [7]) conducted by Lu [3] only exploit node similarity based on the connections, without taking into account the observed information walk features. A supervised link prediction framework [8] parses the sequence of global adjacency matrices at several time-points to derive some input features.

The problem/phenomenon of information spreading on social networks is also related but different. Key points of distinction include the assumption of a broadcast or multitrack spreading model framed as information spreading on *directed acyclic graphs* (DAGs). Important work in this regard includes the Linear Threshold (LT) model [9], Independent Cascade (IC) model ([10] and [11]) and other probabilistic models ([12] and [13]) for DAGs. Moreover, the above works ignore the feature of observed walks, while we model it with measures derived from network analysis. Several papers [14] modeled the speed of information spreading with a type of intensity function. Also of interest are diffusion models (see the discussion in [15]) and work in the context of epidemics (see e.g., the practical case study [16]).

Several works aim to predict the next “item” (visited) for specific applications. James [17] predicted the next institute residency of researchers with a “gravity law”. Li [18] proposed a LSTM model to predict the next company on career paths. Choi ([19] and [20]) applied neural networks to electronic health records data with a sequence of visits over time to predict the next medication code in treatment, where the neural network successfully merges observed medical codes and visits to learn an overall feature representation. The BPR framework [21] previously has been applied to the problem of next-basket prediction for online shopping (see [22] and [23]).

The consideration of a mapping of state transition in a space (e.g., [24]) has been examined in the context of recommender systems. Researchers recently proposed a Transition-based Factorization Machine (TFM) model (see [25] and [26]) to predict next state of a user in a space of items. Our work differs from theirs in that we consider the problem from the perspective of network science and show the power of incorporating network measures. We show that our approach outperforms a TFM of the patient referral network and suggest reasons for this result (cf. Sect. 4).

3 Proposed Model

Given an initial walk on a directed network we aim to predict the next node. Specifically, we model with a numerical score the preference/atraction between the last node on an information walk and a candidate node. In this way, when predicting which candidate node is more likely to be the next node, we just need

to compare the preference/atraction scores over those candidates and return a few candidate nodes with relatively large scores, so users could scan quickly the returned list for options. We describe how to generate the set of candidate nodes in Fig. 1 and evaluate the quality of the returned list in Sect. 4.3. The key is the construction of the score.

To formalize the problem, let P represent the set of chronological sequences of nodes occurring in (information) walks on the network. For any $i \in P$, p_i denotes the feature vector of the observed sequence of nodes before a timepoint T , c_i is the last node on walk i before T , f_i is the first node covered by walk i after T (i.e., the ground truth of next node on the walk). Let J denote the set of all other possible candidates as the next node on the walk. J could be as large as the entire network except c_i , or a subset to facilitate model evaluation if the network is huge. The preference/atraction score between the last observed node c_i and a candidate for the next node $j \in J$ is denoted $X(p_i, c_i, j)$. We would like to build and train a model of the score to make $X(p_i, c_i, f_i) > X(p_i, c_i, j)$ for as many candidates $j \in J$ as possible, which means the model estimates the future direction of an information walk better.

Various network-related measures, either endogenous (topological) or exogenous (metadata) attached to the observed walk, help improve the prediction of the next node. Our early works ([1] and [2]) provide a source of such data and the basis for the construction of our model. The relevant feature list is in Table 3.

3.1 Preference/Attraction Score

We propose a new model in the BPR framework, *BPR-IW*, using information walk features p_i , where the preference/atraction score between the current node c_i and a candidate node $j \in J$ is:

$$X(p_i, c_i, j) = p_i^T V \beta_{c_i} + p_i^T S \gamma_j + \beta_{c_i}^T U \gamma_j + wd(c_i, j) \quad (1)$$

where in the above the superscript T is the matrix transpose operator. In Eq. 1, p_i is the feature of an observed information walk i , β_{c_i} is the feature vector of the last node c_i on the walk, γ_j represents the feature vector of a candidate node j , $d(c_i, j)$ refers to the similarity between the profiles of c_i and j . The parameters in the model include three matrices V, S, U of the node-walk interactions (not only the possible interactions in reality, but also the feature interactions like that in a Polynomial Regression [27] or Factorization Machine [28] model) and a weight vector w . Table 1 shows the size of the feature vectors and model parameters in Eq. 1 expressed in the form of matrix operation.

Table 1. Size of features and model parameters in Eq. 1.

Feature	Size	Definition	Parameters	Size	Definition
p	$M \times 1$	Information walk	V	$M \times N$	Walk-node interaction
β	$N \times 1$	Last node on a walk	S	$M \times H$	Walk-node interaction
f, γ	$H \times 1$	Ground truth/candidate node	U	$N \times H$	Node interaction
d	$L \times 1$	Node profile similarity	w	$1 \times L$	Weight of similarity

Equation 1 considers multiple factors in the determination of the next node on an information walk. V represents the last node's interaction with or the response (if applicable) to the initial part of the walk, S, U describes the degree of matching between the current walk (up to the last node) and a future candidate node, respectively. Network analysis offers the general and versatile features p, β, f, γ that could be derived from the structure of connections between any group of nodes while d depends on the network metadata, so we separate it from the node position-based features β, f, γ . Weight w adjusts the importance of node similarity features in the last group of Table 3.

3.2 Learning the Model

The above model defines a preference/atraction score $X(p_i, c_i, j)$ for the last node c_i on information walk i towards a candidate node j , with the goal of generating a personalized ranking of the candidate nodes in J . When predicting the next node for new information walks, it is easy to compute the scores for each candidate, and then sort to get top-K possible candidates as the result. Therefore, the comparative order of the score matters more than the numerical value. Here we adopt the Bayesian Personalized Ranking (BPR) framework [21]. The framework maximizes a MAP-estimator with regularization, as well as the product of the posterior probability of the model parameters conditional on the latent ranking of the choices (in our case, the choice of next physician by the patient) in Eq. 2,

$$\Theta = \underset{\Theta}{\operatorname{argmax}} \sum_{i \in P_{train}} \sum_{j \in J \setminus \{c_i, f_i\}} \log \sigma(\hat{X}(p_i, c_i, f_i) - \hat{X}(p_i, c_i, j)) - \frac{\lambda_\Theta}{2} \|\Theta\|^2 \quad (2)$$

where σ denotes the sigmoid function given by $\sigma(x) = (1 + \exp(-x))^{-1}$. The sigmoid function is the special case of the logistic regression function in which the gap of preference scores is mapped into the interval $(0, 1)$ to compute the loss function, even if the gap diverges to infinity. The argument of σ , $\hat{X}(p_i, c_i, f_i) - \hat{X}(p_i, c_i, j)$, quantifies the difference in preference score between the walk under the actual next node, f_i , versus a potential other next node, j . The set of candidate nodes J should exclude the last node c_i and the ground truth of the next node f_i . In (2), Θ represents the set of all unknown latent parameters in the model, including V, S, U, w . They may initially be assigned random matrices/vectors (e.g., from a multivariate normal distribution), and they are iteratively optimized in the training process. P_{train} refers to the set of information walks for training. λ_Θ regularizes the parameters in the objective function.

Given the huge number ($O(|P_{train}| |J|)$), of more than 1 billion pairs of information walks and candidate nodes (Table 2) in the dataset, stochastic gradient descent (SGD) is an efficient way to optimize Eq. 2 in order to update the set of parameters Θ with Eq. 3. In each round of training, given a pair of walk i and

candidate node j , the gradients of Eq. 2 with respect to a parameter $\theta \in \Theta$ are:

$$\begin{aligned} & \frac{\partial}{\partial \theta} (\log \sigma(\hat{X}(p_i, c_i, f_i) - \hat{X}(p_i, c_i, j)) - \frac{\lambda_\theta}{2} \|\theta\|^2) \\ &= (1 - \sigma(\hat{X}(p_i, c_i, f_i) - \hat{X}(p_i, c_i, j))) \frac{\partial}{\partial \theta} (\hat{X}(p_i, c_i, f_i) - \hat{X}(p_i, c_i, j)) - \lambda_\theta \theta \end{aligned} \quad (3)$$

The partial derivative of $\hat{X}(p_i, c_i, f_i) - \hat{X}(p_i, c_i, j)$ with respect to a model parameter is evaluated using Eq. 1. Equation 4 shows two examples. We do not need to update V because of the offset.

$$\begin{aligned} \frac{\partial}{\partial S} (\hat{X}(p_i, c_i, f_i) - \hat{X}(p_i, c_i, j)) &= p_i \gamma_{f_i}^T - p_i \gamma_j^T \\ \frac{\partial}{\partial U} (\hat{X}(p_i, c_i, f_i) - \hat{X}(p_i, c_i, j)) &= \beta_{c_i} \gamma_{f_i}^T - \beta_{c_i} \gamma_j^T \end{aligned} \quad (4)$$

4 Evaluation

4.1 Dataset

We used U.S. Medicare beneficiary claims data (courtesy of The Dartmouth Institute for Health Policy and Clinical Practice – TDI) for all patients diagnosed with cardiovascular disease during 2007–2011 to build a sequence of referrals of each patient. A *referral* represents a pair of physician visits within a short time interval which combine to a *sequence of referrals* for the same patient (see [1] for details). A sequence of referrals could be treated as a kind of information walk in the derived physician collaboration network (where physicians are linked if they have ever been a source and target of a referral). Since referral sequences can have loops they give rise to walks on the physician network.

For the set of all referral sequences P in a year, given a timepoint T we extract the training set P_{train} to store the sequences ending before T . The test set P_{test} includes the sequences which are ongoing at T . Figure 1 shows two information walks A and B . Since A ends before time point T , it belongs to P_{train} . At the time point T , the information is still spreading between two nodes on the walk B , so it is in P_{test} . For both of the two walks, the red nodes belong to the observed part to generate information walk feature p . For a walk in P_{train} , all nodes except the last one are treated as the observed part and the last node plays the role of ground truth f . As a random sampling of all nodes in the network, the set of candidate nodes J includes the node f of all walks in P_{test} .

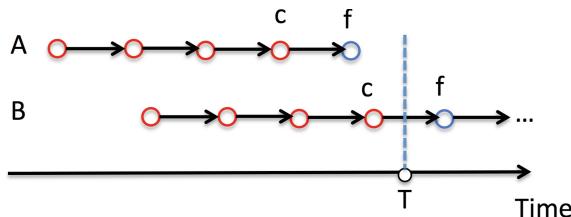


Fig. 1. Example of an information walk A for training and another walk B for testing with the time point T of observation.

Table 2. Size of the TDI patient referral dataset in 2011.

Time point T	#walks in P_{train}	#walks in P_{test}	#candidates in J
Mar. 1st	17569	18652	16628
May 1st	51797	19455	17272
July 1st	83732	16585	14891
Sept. 1st	113073	15751	14300
Nov. 1st	142402	16793	15109

In 2011, the U.S. national physician network supported 4.66M information (referral) walks (as derived from the TDI dataset). After filtering out all (short) walks with fewer than five nodes, Table 2 shows the statistics of the dataset at several time-points T . As P_{train} includes all referral walks ending before T , its size increased from March to November.

Table 3 shows the measures of an information walk p , the feature vector β of the last node c , the feature vector γ of the candidate node j and the ground truth f , and the profile features of physicians for $d(c_i, j)$ are similarly derived. RVU¹, HRR² and other terms are explained in our works ([1] and [2]).

Table 3. The features of information walk and nodes. They include traditional network measures (e.g., PageRank), our proposed measures of an information walk [1], and a few features related to the medical dataset.

Vector	Measures
Information walk p	Length, average time gap, time range, whether a node occurs multiple times, Hospital (PHN) group entropy, #bidirected pairs, total RVU, #covered hospitals, #change points in pagerank sequence, average PageRank for all nodes, average #involved walk for all nodes, average times of being the first node, average times of being the last node, average index of the first occurrence on the walk, average times of multiple occurrences on the walk, average RVUs on the walk
Next node and candidates (j and f)	Clustering coefficient, PageRank, Hindex, #walks in it, times of being the first node, times of being the last node, average first occurrence index, times of multiple occurrences in a walk, #initiated cross-HRR referrals, #initiated cross-PHN referrals
Last node c	In addition to the measures of next node and candidates: time gap with last occurrence, time gap of the previous node, #visiting records, RVU, whether occurs multiple times on the walk, average common neighbors with the previous nodes, indicator of bidirected partner, sign of the gap with the PageRank of the previous node, indicator of the same PHN with the previous node, indicator of the same HRR with the previous node
Node profile for $d(c, j)$	Indicators of same physician specialty/PHN/HRR, edge weights of the directed edge between two nodes

¹ “RVU” stands for “relative value units”, a unit of value of medical service. Different procedures are worth different numbers of RVUs.

² “HRR” stands for “hospital referral region” a geographic area satisfying various medical care criteria [31].

We select the above measures because they show statistical significance for other predictive applications (e.g., the outcome of treatment [1]), and most of them would be widely applicable to other contexts. To avoid the possibility of inconsistencies in the timing and measurement of features when we predict the walks in a given year (e.g., 2011), we would compute node positional measures from the referral network in the previous year (e.g., 2010). Using the prior year’s data to form predictors also helps ameliorate concerns over reverse-causality.

4.2 Baseline Methods

We apply the following methods to get a similarity score X between the last node c and a candidate node j so as to pick up a top-K list of candidate nodes.

Most popular (MP). $X(c, j) = e(c, j)$ It depends on the edge weight between c and its neighbors in the history, such as the number of referred patients.

Two-gram Markov Chain (MC) [23]. $X(c, j) = \text{Prob}(c.\text{next} = j | c, c.\text{prev})$ It considers the second-to-last node $c.\text{prev}$ on a walk to count the frequency.

We also implement several other link prediction methods, such as **Common neighbors (CN)** [29], **Jaccard index** [30], **Adamic-Adar index** [6] and **Preferential attachment index (PA)** [7]. The above similarity-based methods do not consider the initial information walk, and are only applicable for the nodes which have interactions with the last node c in the historical data.

Long short-term memory (LSTM). With a sequence of nodes on an information walk, we put the corresponding features (in Table 3) of all observed nodes into a LSTM model [32]. We hope the LSTM model might articulate the patterns of observed nodes and the output vector would be close to that of the ground truth f . However, the HR (hit-rate) is less than 0.01 for all experimental settings. A related work [19] reported the failure of LSTM model to predict the next medical visit as well.

BPR-no-IW. $X(c, j) = wd(c, j)$. This metric only keeps the node profile similarity in Eq. 1 to explore the effects of information walk features and node features.

Transition-based Factorization Machines (TFM) [26]. The TFM model represents user, current item and next item in a n -dimensional feature vector \vec{y} , and defines a preference score as in Eq. 5. Therein \vec{w} is a linear weight vector, d^2 is the Euclidean distance function, and \vec{v} and \vec{v}' are the latent embedding and translation parameters for each feature unit in \vec{y} , respectively:

$$X(\vec{y}) = w_0 + \sum_{a=1}^n w_a y_a + \sum_{a=1}^n \sum_{b=a+1}^n d^2(\vec{v}_a + \vec{v}'_a, \vec{v}_b) y_a y_b. \quad (5)$$

The hit-rate of TFM on our data is lower than 0.01 for all experiment settings, including a plain \vec{y} with IDs of walker (patient), current node, next node only, and another \vec{y} with the derived network measures. Several possible reasons come

to mind. More than half of the nodes in the physician network have a degree less than four, so the “cold-start” problem limits the TFM model, which runs on filtered datasets [26] composed of the remaining frequent walkers and nodes. Also, when the network measures are not categorical the TFM model might not show its advantage of processing one-hot encoding features. It is possible that not all pairs of feature interactions help with the estimation, and the transition parameter \vec{v}' should depend on the observed walk.

As the main point of this paper is to point out the importance of network measures for the walk prediction problem, further exploration of Factorization Machines [28] related preference scores is left for future work.

4.3 Results

For an information walk i and the ground truth of the next node f_i , given a sorted list of K candidate nodes R_i predicted by BPR-IW or other baseline method, we define two evaluation metrics: hit-rate (HR) and mean percentile rank (MPR) in Eq. 6. HR shows the possibility that a model could predict and present the ground truth f to users, and MPR reflects the quality of the prediction. A larger HR yet smaller MPR implies better prediction. Here we sort the top- K returned list in decreasing order of the preference/attraction score.

$$\begin{aligned} HR &= \frac{1}{|P_{test}|} \sum_{i \in P_{test}} \mathbb{1}(f_i \in R_i) \\ MPR &= \frac{1}{HR \times |P_{test}|} \sum_{i \in P_{test}, f_i \in R_i} \frac{\text{rank}(f_i)}{K} \end{aligned} \quad (6)$$

The following figures display the result of prediction of all successful models on the TDI patient referral dataset over 2008–2011. The initial step size of SGD is 0.05 and the λ_Θ in Eq. 2 is 0.001. The results of LSTM and TFM models are too low (hit-rate less than 0.01) to display.

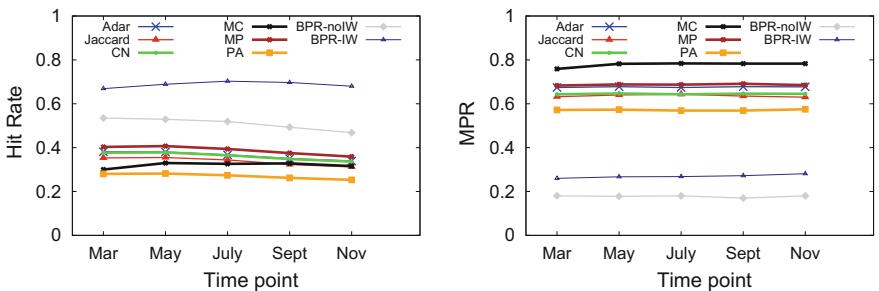


Fig. 2. HR and MPR in 2011. The size of returned list K is 20.

Figure 2 shows the HR and MPR of several models in 2011, when each model returns $K = 20$ candidate nodes. In terms of HR, BPR-IW is the best model.

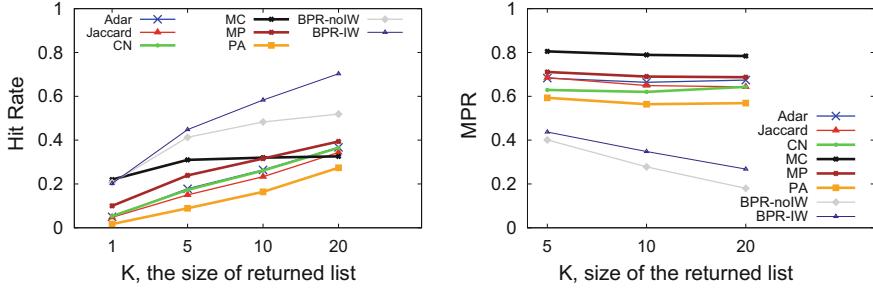


Fig. 3. HR and MPR under several K on July 1st, 2011. When $K = 1$, MPR = 1.0

Besides, BPR-IW and BPR-noIW have the smallest MPR (around 0.2), which means the ground truth f has a better rank among the returned K candidates.

Figure 3 shows the effects of K on HR and MPR for several models on July 1st, 2011. With the increase of K , most of the models tend to predict better with a higher HR. For the BPR-IW model, when $K = 20$, the HR is larger than 0.7 for the large P_{test} consisting of more than 10,000 information walks. Traditional link prediction measures are almost invariant to K in terms of MPR, but BPR-IW and BPR-noIW present a lower MPR with a larger K . Therefore, it would be better to increase K a bit for BPR based models to include the ground truth f with a top position/rank in the returned candidate list. In practice, the choice of K depends on the design of the walk prediction system, like the number of webpages returned by a search engine.

Figure 4 shows the performance on the same day of July 1st in 2008–2011 ($K = 20$). All models achieve stable performance.

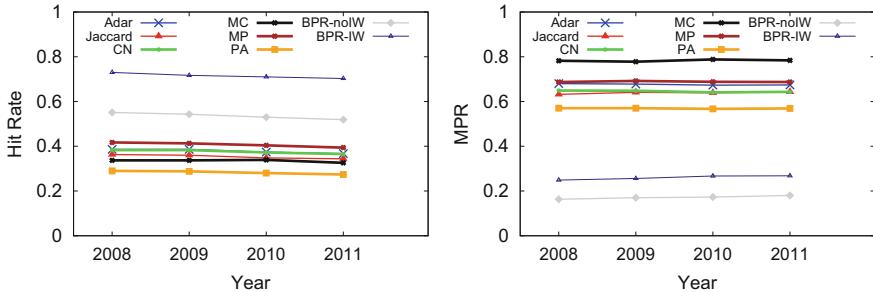


Fig. 4. HR and MPR on the same day (July 1st) in 2008–2011. K is 20.

Table 4 displays the recall of the BPR-IW model for groups of P_{test} on July 1st, 2011. For the number of nodes (length) and time range of an information walk, we classify it into one of the five groups according to the percentile after min-max normalization. We count the recall for the information walks whose f

Table 4. The recall of BPR-IW model on groups divided by length and time range of a walk under different K , on the day of July 1st, 2011.

(a) Length						(b) Time range					
K	0–20%	20–40%	40–60%	60–80%	80–100%	K	0–20%	20–40%	40–60%	60–80%	80–100%
1	0.208	0.202	0.201	0.196	0.174	1	0.191	0.207	0.194	0.216	0.215
5	0.462	0.448	0.459	0.436	0.430	5	0.445	0.449	0.454	0.459	0.474
10	0.605	0.588	0.601	0.565	0.564	10	0.595	0.590	0.584	0.598	0.613
20	0.720	0.710	0.727	0.692	0.690	20	0.727	0.718	0.698	0.714	0.711

are successfully hit by BPR-IW. The stable recall values among all groups for different sizes of the returned list K tend to support that BPR-IW does not have a preference to predict one group (e.g., short walk) better than another.

The above experiments show that features derived from information walks and network analysis improve HR at the cost of a slightly larger MPR. We think it is more important to include the ground truth f in the returned list R , so over all of the experimental settings, BPR-IW gets the best performance. The traditional simple similarity based link prediction methods do not perform as well as the multi-factor BPR based models, because they ignore the information walk feature p and only focus on the connected or related nodes according to the historical data. The BPR framework does not predict the transition probability, but the comparative ranking is sufficient if users are not concerned about a quantitative description of future walks. From the network science perspective we strongly suggest the adoption of network measures and the derived walk (referral path) features in future research. The application of data-specific features to this problem appears likely to reduce misleading node suggestions, such as the profile similarity between two physicians.

5 Conclusions

Herein, we use the context of patient referral sequences in the U.S. physician network to investigate the problem of next node prediction on single-track information walks from a network science perspective. We exploit both features of the information walk and (intrinsic) node position measures to build a BPR-IW model of preference/atraction between nodes. Without the requirement of direct connections in history, the network-based measures enable the deployment of a flexible BPR-IW model on a wider range of candidate nodes, and BPR-IW performs well on the patient referral dataset in terms of a sensitivity analysis that examines the performance across a range of settings for the size of the returned list and the timepoint of observation. The model is quite general and may be easily adapted to other contexts where network measures make sense for a group of nodes with various metadata. Anticipated future work includes more experiments on more diverse datasets and modeling the preference score from the perspective of collaborative filtering.

References

1. An, C., O'Malley, A.J., Rockmore, D.N.: Referral paths in the U.S. physician network. *Appl. Netw. Sci.* **3**(1), 20 (2018)
2. An, C., O'Malley, A.J., Rockmore, D.N., Stock, C.D.: Analysis of the US patient referral network. *Stat. Med.* **37**(5), 847–866 (2018)
3. Lu, L., Zhou, T.: Link prediction in complex networks: a survey. *Phys. A Stat. Mech. Appl.* **390**(6), 1150–1170 (2011)
4. LibenNowell, D., Kleinberg, J.: The linkprediction problem for social networks. *J. Am. Soc. Inf. Sci. Technol.* **58**(7), 1019–1031 (2007)
5. Clauset, A., Moore, C., Newman, M.E.: Hierarchical structure and the prediction of missing links in networks. *Nature* **453**(7191), 98 (2008)
6. Adamic, L.A., Adar, E.: Friends and neighbors on the web. *Soc. Netw.* **25**(3), 211–230 (2003)
7. Barabási, A.L., Albert, R.: Emergence of scaling in random networks. *Science* **286**(5439), 509–512 (1999)
8. Lu, Z., Savas, B., Tang, W., Dhillon, I.S.: Supervised link prediction using multiple sources. In: IEEE 10th International Conference on Data Mining. IEEE (2010)
9. Chen, W., Yuan, Y., Zhang, L.: Scalable influence maximization in social networks under the linear threshold model. In: IEEE 10th International Conference on Data Mining (ICDM). IEEE (2010)
10. Saito, K., Nakano, R., Kimura, M.: Prediction of information diffusion probabilities for independent cascade model. In: International Conference on Knowledge-Based and Intelligent Information and Engineering Systems. Springer, Berlin (2008)
11. Bourigault, S., Lamprier, S., Gallinari, P.: Representation learning for information diffusion through social networks: an embedded cascade model. In: Proceedings of the Ninth ACM WSDM Conference. ACM (2016)
12. Gomez Rodriguez, M., Leskovec, J., Schölkopf, B.: Structure and dynamics of information pathways in online media. In: Proceedings of the Sixth ACM International Conference on Web Search and Data Mining. ACM (2013)
13. Rodriguez, M.G., Balduzzi, D., Schölkopf, B.: Uncovering the temporal dynamics of diffusion networks. arXiv preprint [arXiv:1105.0697](https://arxiv.org/abs/1105.0697) (2011)
14. Guille, A., Hacid, H., Favre, C., Zighed, D.A.: Information diffusion in online social networks: a survey. *ACM Sigmod Rec.* **42**(2), 17–28 (2013)
15. Milli, L., Rossetti, G., Pedreschi, D., Giannotti, F.: Information diffusion in complex networks: the active/passive conundrum. In: International Workshop on Complex Networks and Their Applications. Springer, Cham (2017)
16. Joneydi, S., Khansari, M., Kaveh, A.: An opportunistic network approach towards disease spreading. In: International Workshop on Complex Networks and Their Applications. Springer, Cham (2017)
17. James, C., Pappalardo, L., Sirbu, A., Simini, F.: Prediction of next career moves from scientific profiles. arXiv preprint [arXiv:1802.04830](https://arxiv.org/abs/1802.04830) (2018)
18. Li, L., Jing, H., Tong, H., Yang, J., He, Q., Chen, B.C.: Nemo: next career move prediction with contextual embedding. In: Proceedings of the 26th International Conference on World Wide Web Companion (2017)
19. Choi, E., et al.: Multi-layer representation learning for medical concepts. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM (2016)
20. Choi, E., Bahadori, M.T., Schuetz, A., Stewart, W.F., Sun, J.: Doctor AI: predicting clinical events via recurrent neural networks. In: Machine Learning for Healthcare Conference (2016)

21. Rendle, S., Freudenthaler, C., Gantner, Z., Schmidt-Thieme, L.: BPR: Bayesian personalized ranking from implicit feedback. In: Proceedings of UAI. AUAI Press, pp. 452–461 (2009)
22. Wang, P., Guo, J., Lan, Y., Xu, J., Wan, S., Cheng, X.: Learning hierarchical representation model for next basket recommendation. In: Proceedings of the 38th International ACM SIGIR Conference. ACM (2015)
23. Rendle, S., Freudenthaler, C., Schmidt-Thieme, L.: Factorizing personalized Markov chains for next-basket recommendation. In: Proceedings of the 19th International Conference on World Wide Web. ACM (2010)
24. Leibon, G., Rockmore, D.N.: Orienteering in knowledge spaces: the hyperbolic geometry of wikipedia mathematics. *PLoS One* **8**(7), e67508 (2013)
25. He, R., Kang, W.C., McAuley, J.: Translation-based recommendation. In: Proceedings of the Eleventh ACM Conference on Recommender Systems. ACM (2017)
26. Pasricha, R., McAuley, J.: Translation-based factorization machines for sequential recommendation. To appear in Proceedings of the Eleventh ACM Conference on Recommender Systems. ACM (2018)
27. Peixoto, J.L.: A property of well-formulated polynomial regression models. *Am. Stat.* **44**(1), 26–30 (1990)
28. Rendle, S.: Factorization machines. In: IEEE 10th International Conference on Data Mining. IEEE (2010)
29. Lorrain, F., White, H.C.: Structural equivalence of individuals in social networks. *J. Math. Sociol.* **1**(1), 49–80 (1971)
30. Jaccard, P.: Étude comparative de la distribution florale dans une portion des Alpes et des Jura. *Bull. Soc. Vaud. Sci. Nat.* **37**, 547 (1901)
31. The Dartmouth Atlas of Health Care. <http://www.dartmouthatlas.org/tools/faq/researchmethods.aspx>. Accessed May 2018
32. Hochreiter, S., Schmidhuber, J.: Long short-term memory. *Neural Comput.* **9**(8), 1735–1780 (1997)



Average-Case Behavior of k -Shortest Path Algorithms

Alexander Schickedanz¹, Deepak Ajwani^{2(✉)}, Ulrich Meyer¹, and Paweł Gawrychowski³

¹ Goethe University, Frankfurt am Main, Germany

{alex,umeyer}ae.cs.uni-frankfurt.de

² Nokia Bell Labs, Dublin, Ireland

deepak.ajwani@nokia-bell-labs.com

³ Instytut Informatyki, Uniwersytet Wrocławski, Wrocław, Poland
gawry@cs.uni.wroc.pl

Abstract. The k -shortest path problem is a generalization of the fundamental shortest path problem, where the goal is to compute k simple paths from a given source to a target node, in non-decreasing order of their weight. With numerous applications modeling various optimization problems and as a feature in some learning systems, there is a need for efficient algorithms for this problem. Unfortunately, despite many decades of research, the best directed graph algorithm still has a worst-case asymptotic complexity of $\tilde{O}(k n(n+m))$. In contrast to the worst-case complexity, many algorithms have been shown to perform well on small diameter directed graphs in practice. In this paper, we prove that the average-case complexity of the popular Yen's algorithm on directed random graphs with edge probability $p = \Omega(\log n)/n$ in the unweighted and uniformly distributed weight setting is $O(k m \log n)$, thus explaining the gap between the worst-case complexity and observed empirical performance. While we also provide a weaker bound of $O(k m \log^4 n)$ for sparser graphs with $p \geq 4/n$, we show empirical evidence that the stronger bound should also hold in the sparser setting. We then prove that Feng's directed k -shortest path algorithm computes the second shortest path in expected $O(m)$ time on random graphs with edge probability $p = \Omega(\log n)/n$. Empirical evidence suggests that the average-case result for the Feng's algorithm holds even for $k > 2$ and sparser graphs.

Keywords: k -Shortest path algorithms · Average case analysis
Yen's algorithm · Feng's algorithm

1 Introduction

An important problem in the analysis of complex networks is to find structural relationships between the nodes. In many applications, one is interested in finding many short *simple* paths between two nodes to reveal deeper semantic insights into the relationship between the nodes. For instance, in areas such as financial

fraud analysis, law-enforcement and journalistic investigations, a query such as “What are the different ways in which the person A and person B have indirectly communicated/dealt with/transacted with?” can reveal important connections between individuals that can then be further investigated. For the problem of recommending people to connect in a social network (a problem crucial for the growth of a social network), statistics from short simple paths can be used as a feature in prediction systems as well as in presenting a visual evidence for convincing a user that he/she should connect to the recommended person.

The problem of finding many short paths has applications well beyond finding structural relationship between nodes. For instance, Shih and Parthasarathy [31] used an approach for finding many short simple paths to identify the potential regulatory pathways in a Yeast gene network.

A natural way to find many short paths is to model it as the k -shortest simple path problem [9]. This is a generalization of the fundamental shortest path problem, where the goal is to compute not just one, but k loop-less paths between a given source and target node, in non-decreasing order of their weight. In this variant of k -shortest path, all shortest paths (and not just the first) are constrained to be cycle-free (simple). This is because cycles rarely add any insight when studying structural relationships between nodes, particularly in applications such as semantic search and recommending people to connect to. Note that in this variant, the resultant paths need not be node or edge-disjoint.

The best asymptotic complexity of algorithms for the loop-less variant of the k -shortest path on the weighted directed graphs is $O(k n(m + n \log \log n))$ [17]. There is considerable evidence [2, 32] that the asymptotic worst-case bound can't be improved significantly. This poor asymptotic complexity in the directed graph setting is in stark contrast with the significantly better asymptotic complexity of $O(k(m + n \log n))$ [21] in the undirected setting. This difference in asymptotics discourages the use of exact k -shortest path algorithms in applications where structural relationship between nodes in directed graphs needs to be computed efficiently.

A recent empirical study [3] has found that despite the difference in asymptotics between algorithms for directed and undirected graphs, they are fairly competitive in average on random graphs and that some of the directed graph k -shortest path algorithms scale well on random graphs. We hypothesize that two of the popular directed graph k -shortest simple path algorithms with poor worst-case asymptotic complexity, by Yen [33] and Feng [13], have near-linear average-case asymptotic complexity (explaining their scalability in practice). In this paper, we provide considerable theoretical and empirical evidence to support this hypothesis. Thus, our work bridges this important gap between the worst-case complexity and the empirical performance of these algorithms and motivates the usage of these algorithms in many network analytics applications that have strong efficiency requirements.

We first show that the average-case asymptotic complexity of Yen's algorithm is $O(k m \log n)$ on random directed graphs with uniform random weights, in the setting where $k = O(n)$ and $m = \Omega(n \cdot \log n)$. This setting is motivated by the facts that (i) choosing $k = \omega(n)$ would result in running times of $\omega(n^2)$ which

are infeasible for huge values of n , (ii) the average-case computational savings on random graphs over general graphs are more significant when $m \gg n$, and (iii) we can rely on certain structural properties (lacking in very sparse random graphs) which significantly simplify our proofs. While we also provide a weaker bound of $O(k m \log^4 n)$ for sparse random graphs, we show empirical evidence indicating that the stronger bound for the denser case even holds for very sparse random graphs.

Next, we show that the average-case complexity of computing the second shortest path with Feng's algorithm [13] is even better, namely $O(n + m)$ on unweighted random undirected graphs. Our experimental results suggest that similar bounds hold for $k > 2$ as well, even with uniform and exponential edge weights.

Remark: Note that in this paper, the average-case complexity refers to the expected complexity of computing k -shortest paths from a randomly chosen source node to a randomly chosen target node on a directed random graph from the class $\mathcal{D}(n, p, [0, 1])$. The class $\mathcal{D}(n, p, [0, 1])$ comprises directed graphs on n nodes, where each edge exists with probability p independent of the presence of all other edges, and the edge weights are uniformly distributed between 0 and 1.

2 Related Work

In this paper, our focus is on the average-case complexity of *exact* algorithms for the *loop-less* k -shortest path problem. From a worst-case asymptotic complexity perspective, the best algorithm for this problem in the weighted case is by Gotthilf and Lewenstein [17], that achieves a bound of $O(kn(n \log \log n + m))$. For directed *unweighted* graphs, Roditty and Zwick [29] have improved the bound to $O(km\sqrt{n} \log^2 n)$.

Due to fine-grained conditional complexity results by Williams and Williams [32] and Agarwal and Ramachandran [2], it is unlikely that the asymptotic worst-case complexity of k -shortest path on directed weighted graphs can be improved to $O(n^{3-\delta})$ for some $\delta > 0$ or $o(nm)$, respectively.

In this paper, we focus on a very popular k -shortest simple path algorithm, originally proposed by Yen [33]. Yen's algorithm considers all possible $O(n)$ deviations from each of the previously computed paths. This takes $O(kn(n \log n + m))$ time when using Dijkstra's SSSP algorithm [11] with Fibonacci heaps [14] as a kernel.

There has also been considerable work on practical improvements of the directed k -shortest simple path algorithm (e.g., [13, 19, 22–25, 27, 30, 31]). In particular, Feng's algorithm [13] claims to be the empirically fastest approach on many graph classes, while still matching the worst-case time complexity of Yen's algorithm. In this paper, we present theoretical and empirical evidence to explain the performance of this algorithm in an average-case setting.

The best k -shortest path algorithm for *undirected* graphs [21] has a significantly better asymptotic worst-case complexity of $O(k(m + n \log n))$, compared to directed graph algorithms. Despite the big gap between the worst-case

asymptotic complexity of directed and undirected graph algorithms, a recent experimental paper [3] showed comparable performance of these algorithms on the random graphs, motivating this theoretical study.

Our current study on average-case complexity of exact k -shortest simple path algorithms excludes the work done on approximating k -shortest simple paths [5, 15], all-pair k -shortest simple paths [1], loopy k -shortest path algorithms [4, 12] as well as distance oracles for computing replacement paths [6, 10].

3 Algorithms

3.1 Yen's Algorithm

Given a graph $G(V, E)$, a source node s and a target node t , Yen's algorithm [33] starts by computing the shortest path from s to t , namely $P_0 = (s, \dots, t)$. The algorithm then maintains a priority queue C of candidates for the next shortest paths. Initially, the queue C consists of only P_0 with the priority $\text{weight}(P_0)$.

We then iteratively compute the shortest paths P_i for $i = 1, \dots, k - 1$ and update C with candidate paths for the next iteration. The path P_i is obtained by simply extracting the minimum weight path from C . Let $P_i = (v_1^i = s, v_2^i, \dots, v_d^i, \dots, v_l^i = t)$ and let P_j be the path with maximum common prefix $(v_1^j = v_1^i, v_2^j = v_2^i, \dots, v_d^j = v_d^i)$ among all paths $\{P_1, \dots, P_{i-1}\}$. Further, let $\text{dev}(P_i) = v_d^i$ be the node at which P_i deviated from P_j (we define $v_d^1 = s$ for P_1). Then, the new candidate paths to be inserted into C are the shortest paths deviating from P_i at nodes $\{v_d^i, \dots, v_{l-1}^i\}$. To compute the deviation path from a node v_f^i , we first compute the shortest path P^f from v_f^i to t in the graph G' obtained by removing nodes $\{v_1^i, \dots, v_{f-1}^i\}$, incident edges and all edges in E^f from G , where $E^f = \{(v_h^i, v_{h+1}^i) : 1 \leq h \leq i \text{ and } v_1^h = v_1^i, \dots, v_f^h = v_f^i\}$. The shortest deviation path from P_i at a node v_f^i is simply the path $\{v_1^i, v_2^i, \dots, v_{f-1}^i\}$ appended by P^f . We insert it into C with its weight as priority. See Algorithm 1 for a pseudo-code of the algorithm.

3.2 Feng's Algorithm

Feng's algorithm [13] optimizes the computation of deviations in Yen's algorithm by significantly pruning the subgraph for deviation computation. We refer to the pruned subgraph, involved in the computation of the deviation from v_f^i from path P_i , as the yellow graph $Y_f^i(V_y^{i,f}, E_y^{i,f})$. We next describe the set of nodes and edges in this graph and then modify the description for efficient computation of yellow graphs.

Let T be the reverse shortest path tree rooted in t . When computing the deviation from path P_i at node v_f^i , we conceptually colour the nodes into three colours: Nodes v_1^i, \dots, v_f^i are coloured red, all nodes whose shortest path to t in T goes through a red node are yellow and the remaining nodes are green. The node set $V_y^{i,f}$ of the yellow graph Y_f^i consists of all yellow nodes, only one red node v_f^i and only one green node t . The set E_y consists of all edges between the

yellow nodes and only outgoing edges of v_f^i . All edges $(u, v) \in E$ where u is a yellow node and v is a green node are replaced by edges (u, t) in E_y . The weight of the edge (u, t) in E_y equals the sum of the weight $w(u, v)$ of edge (u, v) in E and the weight of the path $d(v)$ from v to t in T .

We modify the original description of Feng's algorithm [13] to compute the yellow graphs efficiently. The reverse shortest path tree T from t is computed only once, before the deviation computation commences. As part of the shortest path tree computation, we also store the distance $d(v)$ of a node v from t in T in addition to all the in-edges to v . To compute $V_y^{i,f}$, we traverse T from each of the red nodes r to identify the subtree of yellow nodes rooted at r in T . A simple BFS traversal from r in T that skips the other red nodes (and the subtrees rooted at them) suffices for this purpose. The union of nodes in all these subtrees, together with v_f^i and t , constitutes $V_y^{i,f}$. Since the yellow nodes are disjointly partitioned among the different subtrees and each red node is skipped in exactly one traversal, these traversals touch $O(n_y^i(f) + n_r^i(f))$ nodes and edges altogether, where $n_y^i(f) = |V_y^{i,f}|$ and $n_r^i(f)$ is the number of red nodes. Skipping can be performed efficiently by storing the red nodes in a hash table, resulting in the overall complexity of $O(n_y^i(f) + n_r^i(f))$ for computing $V_y^{i,f}$.

Algorithm 1 Yen's Algorithm

```

Require:  $G(V, E)$ ,  $s$ ,  $t$ ,  $K$                                  $\triangleright$  source, target, # paths
1:  $K_{\text{paths}} = \emptyset$ 
2:  $P_0 = \text{sssp}(G, s)$ ,  $\alpha_0 = 0$ ,  $j_0 = 0$ 
3:  $C = \emptyset$ 
4:  $k = 1$ 
5: while  $k < K$  and  $C \neq \emptyset$  do
6:    $P_{\text{root}} = (v_0^{k-1}, \dots, v_{\alpha_{k-1}-1}^{k-1})$ 
7:    $G'(V', E') = G[V \setminus P_{\text{root}}]$                        $\triangleright P_{\text{root}}$  as a set.
8:   for  $i = 0, \dots, k-2$  do
9:     if  $j_i == j_{k-1}$  or  $i == j_{k-1}$  then
10:       $E' = E' \setminus \{(v_{\alpha_i}^i, v_{\alpha_i+1}^i)\}$ 
11:    end if
12:   end for
13:   for  $n = \alpha_{k-1}, \dots, q_{k-1} - 1$  do
14:      $E' = E' \setminus \{(v_n^{k-1}, v_{n+1}^{k-1})\}$ 
15:      $P = \text{sssp}(G', n)$ 
16:      $P = P_{\text{root}} \cup P$ ,  $j = k-1$ ,  $\alpha = n$ 
17:      $C = C \cup \{(P, j, \alpha)\}$ 
18:      $G'(V', E') = G[V \setminus \{v_n^{k-1}\}]$ 
19:   end for
20:    $(P_k, j_k, \alpha_k) = \text{shortest path in } C$ 
21:    $C = C \setminus \{(P_k, j_k, \alpha_k)\}$ 
22:    $K_{\text{paths}} = K_{\text{paths}} \cup \{P_k\}$ 
23:    $k = k + 1$ 
24: end while
Ensure:  $K_{\text{paths}}$ 

```

For computing the set $E_y^{i,f}$, we consider all outgoing edges of yellow nodes. For an outgoing edge (y, u) of a yellow node y , if u is a yellow node, we insert (y, u) in $E_y^{i,f}$. If instead u is a green node, we insert (y, t) in $E_y^{i,f}$ with weight equal to $d(u) + w(u, v)$. Otherwise, we ignore it. Clearly, this requires touching $O(m_y^i(f) + m_r^i(f))$ edges, where $m_y^i(f) = |E_y^{i,f}|$ and $m_r^i(f)$ is the number of edges from yellow nodes to red nodes. This is because each outgoing edge from a yellow node either results in an edge in $E_y^{i,f}$ or leads to a red node.

In addition, we add all edges from v_f^i to yellow nodes in $E_y^{i,f}$. In the computation of the i^{th} shortest path, each node in the graph can appear at most once as v_f^i for some deviation. Thus, summing over all k shortest paths, this requires touching $O(km)$ edges in total. Therefore, the total complexity of creating the yellow graphs over all deviations (excluding the initial SSSP computation) is $O(km + \sum_{i=1}^{k-1} \sum_{f=1}^l \mathfrak{Y}^i(f))$ for

$$\mathfrak{Y}^i(f) := n_y^i(f) + m_y^i(f) + n_r^i(f) + m_r^i(f) \quad (1)$$

where the inner summation is taken over all deviations in the computation of i^{th} shortest path.

4 Average-Case Analysis of Yen's Algorithm

Theorem 1. *For random directed graphs from $\mathcal{D}(n, p, [0, 1])$ with $p = \Omega(\frac{\ln n}{n})$, the average-case complexity of Yen's k -SP algorithm with $k = O(n)$ is $O(km \log n)$ w.h.p.*

Proof. Let us consider the work done by Yen's algorithm when going from the i^{th} shortest path P_i to the $(i+1)^{th}$ shortest path: It computes deviation paths from nodes $dev(P_i)$ in path P_i to t . Each deviating path finding boils down to removing $O(m)$ edges, which can be done in $O(n+m)$ complexity, and subsequently running a SSSP computation. Using Dijkstra's algorithm [11] with Fibonacci heaps [14], the shortest-path computation itself takes $O(n \log n + m)$ time. By Chernoff bounds and the lower bound on the edge probability p we have $m = \Omega(n \log n)$ w.h.p., thus implying that each shortest-path computation will take at most $O(n+m)$ time w.h.p. It remains to show that for $k = O(n)$, the overall number of deviations required by Yen's k -SP algorithm is $O(k \log n)$ w.h.p. In order to see this note in Yen's algorithm, at most one deviation is computed from each node in the i^{th} shortest path when computing the $(i+1)^{th}$ shortest path. Thus, it suffices to prove that the number of hops in the i^{th} shortest path is $O(\log n)$ w.h.p. for $i < k$. To this end we will leverage the following properties proved by Priebe [28]:

(P1) If $p = \Omega(\frac{\ln n}{n})$ for sufficiently large constants then the diameter (i.e., the maximum shortest path weight among all n^2 pairs of vertices) of generated random graphs in $\mathcal{D}(n, p, [0, 1])$ is bounded by $O(\frac{\log n}{n \cdot p})$ w.h.p. (P2) If $p = \Omega(\frac{\log n}{n})$

for sufficiently large constants then shortest paths in $\mathcal{D}(n, p, [0, 1])$ consist of $O(\log n)$ edges w.h.p.¹

The proof for (P2) in [28] is based on the observation that with high probability random weighted graphs from $\mathcal{D}(n, p, [0, 1])$ do not contain simple paths of total weight at most $\delta := C \cdot (\log n)/(n \cdot p) < 1$ that at the same time consist of more than $D \cdot \log n$ edges for appropriately chosen constants C and D . Since the threshold δ from [28] is just an upper bound on the diameter given in (P1), (P2) is also applicable for the i^{th} shortest path, $i < k = O(n)$, as long as its path weight stays below δ . Thus, in the remainder we only need to argue that $\mathcal{D}(n, p, [0, 1])$ contains $\Omega(n)$ edge-disjoint $s - t$ paths of weight less than δ with high probability:

Prior to actually generating the input graph G according to $\mathcal{D}(n, p, [0, 1])$ let us arbitrarily partition its nodes into two equally sized sets while ensuring that s and t belong to different sets. Let the graphs induced by the two partitions be G_s and G_t , respectively. By construction G_s and G_t can be considered to stem from $\mathcal{D}(n/2, p, [0, 1])$. Hence, w.h.p., for $p = \Omega(\frac{\ln n}{n})$ both subgraphs are strongly connected [18] and (P1) & (P2) also hold for them. For each node x in G_s let I_x denote the random binary indicator variable for the event that there is at least one edge from x with a target node y in G_t and edge weight at most $\frac{1}{np}$. There are $n/2$ potential edges from x into G_t each independently showing up with probability p . Hence, the probability for I_v being 1 is at least $1 - (1 - p \cdot \frac{1}{np})^{n/2} = 1 - (1 - \frac{1}{n})^{n/2} \geq 1 - e^{-1/2} > 1/4$. Thus, $E[I_v]$ is a constant between 0 and 1 and therefore we can use Chernoff bounds to show that $\sum_{x \in G_s} I_x$ is $\Theta(n)$ w.h.p. This implies that there are $\Theta(n)$ alternative paths from s to t via edges (x_i, y_i) . Due to (P1) each of these paths consists of at most $2 \cdot D \cdot \log(n/2) + 1 = O(\log n)$ edges w.h.p. and due to (P2) their respective paths lengths are at most $O(2 \cdot \frac{\log(n/2)}{(n/2) \cdot p} + \frac{1}{n \cdot p}) = O(\frac{\log n}{n \cdot p})$ w.h.p., which is below the threshold δ for sufficiently large p . \square

Theorem 2. *For sparse random directed graphs from $\mathcal{D}(n, p, [0, 1])$ with $\frac{4}{n} \leq p \leq \frac{\Theta(\ln n)}{n}$, the average-case complexity of Yen's k -SP algorithm with $k = O(n)$ and both s and t being part of the giant strong component² is bounded by $O(k m \log^4 n)$ w.h.p.*

Proof. This rather crude upper bound can be shown with a similar proof like the one we used for Theorem 1. One extra log factor results from the now dominating priority queue operation costs within Dijkstra's algorithm³. Furthermore, the

¹ In fact, Priebe showed (P1) in the form of his Lemma 3.4 and (P2) in the form of his Lemma 3.10, for any edge weight distribution function F that satisfies the following requirements: F is concentrated on $[0, \infty)$, $F(0) = 0$ and that $F'(0)$ exists and is strictly positive. Since the uniform edge weight distribution between 0 and 1 is compatible with these requirements, (P1) and (P2) hold for $\mathcal{D}(n, p, [0, 1])$, too.

² For $p \geq 4/n$ the size of the giant strong component is $\Omega(n)$ w.h.p. [20].

³ We are aware that there exist improved SSSP algorithms with linear average-case time (e.g., [16, 26]) for initially uniformly distributed edge weights. Unfortunately, for the particular subgraphs on which we would like to apply these better SSSP

underlying proof technique for (P2) does not seem to be fully transferable to the sparse setting where the expected weights for shortest s - t paths are higher: by subdividing these paths into logarithmically many subpaths with smaller weights, we can reuse the old analysis but this incurs an extra multiplicative log factor in our upper bound for the time. Similarly, for sparse random graphs the relative node degree fluctuation is likely to be larger than on their dense counterparts, which accounts for at most another logarithmic factor in case one wants to stick to the old analysis structure. \square

Theorems 1 and 2 carry over to the respective *unweighted* cases. The proofs can be streamlined since shortest path distances now coincide with the number of edges on these paths. As a positive consequence, with high probability, we now find even more equally suited alternative shortest paths between the two graph partitions, which is why the results also hold for higher values of k up to $\Theta(n^2 \cdot p) = \Theta(m)$.

4.1 Empirical Results on Average-Case Behavior of Yen's Algorithm

The bound of $O(k m \log n)$ in Theorem 1 was proved under the assumption of $p > 2 \ln n / n$ (and $k = O(n)$). Next, we provide empirical evidence that similar bound is likely to hold for the setting of even sparser graphs, even though the bound proven in Theorem 2 is off by a poly-logarithmic factor. For this, we consider $n = 2^j \cdot 10^6$ for $j = 0, \dots, 7$, $m = 4n$ and generate directed random graphs with n nodes, m edges following the $G(n, m)$ model [7] without self-loops⁴. For the edge weights, we consider two settings, (i) unweighted and (ii) uniform random edge-weight distribution in $[0, 1]$. After generating a graph for a fixed n , we draw the source and target nodes uniformly at random and average the results over 10 different source-target pairs.

Figure 1 show that the number of hops in the k^{th} shortest path and the number of deviations required to compute the $(k+1)^{th}$ shortest path is quite well approximated by $\log(nk)$, indicating that even in this sparse setting, the average-case complexity bound of $O(k m \log(nk))$ ($O(k m \log n)$ for $k = O(n)$) is likely to hold. Figure 2 shows the same for larger values of k .

5 Average-Case Analysis of Feng's Algorithm

Theorem 3. *On unweighted random undirected graphs with $p = \Omega(\frac{\ln n}{n})$, the average-case complexity of Feng's algorithm for computing the second shortest path is $O(m)$.*

algorithms it seems difficult to prove that their overall edge weight distribution remains sufficiently uniform

⁴ We note that for convex properties, the $G(n, p)$ and $G(n, m)$ random graph models are equivalent up to lower order terms, provided $m \approx p \cdot N$ (where N is the maximum number of edges insertable in a graph, $N = n(n - 1)$ for directed graphs) [7]. Thus, the empirical results shown for the $G(n, m)$ model should also hold for the $G(n, p)$ model.

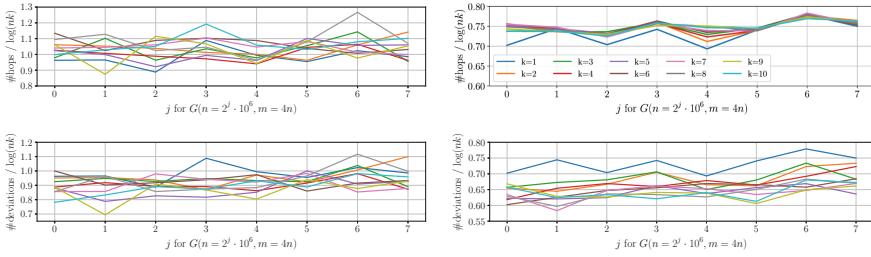


Fig. 1. The number of hops (1^{st} row) of the k^{th} shortest path and the number of deviations (2^{nd} row) to compute the $(k+1)^{th}$ shortest path from the k^{th} with respect to $\log(nk)$ for $G(n, 4n)$ with uniform random edge weights (left) and unweighted (right).

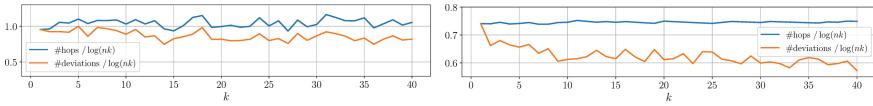


Fig. 2. The number of hops of the k^{th} shortest path and deviation computed from it for the $(k+1)^{th}$ shortest path with respect to $\log(nk)$, computed for a directed random graph from class $G(n = 32 \cdot 10^6, 4n)$ with uniform random edge weights (left) and unweighted (right).

Proof. Consider the three main steps involved in Feng's algorithm: (i) Computing SSSP from t : This requires $O(n + m)$ time using BFS for unweighted graphs. (ii) Computation of yellow graphs: As argued in Sect. 3, this requires $O(km + \sum_{i=1}^{k-1} \sum_{f=1}^l \mathfrak{Y}^i(f))$ operations, where the second summation is taken over all deviations in the computation of i^{th} shortest path. (iii) SSSP computations in the yellow graph: Every shortest path computation can be performed in $O(\mathfrak{Y}^i(f))$ time using BFS. Once again, the summation over all deviations yields a total bound of $O(\sum_{i=1}^{k-1} \sum_{f=1}^l \mathfrak{Y}^i(f))$.

Thus, the total complexity of Feng's algorithm for computing the k shortest paths is $O(n + m + \sum_{i=1}^{k-1} \sum_{f=1}^l \mathfrak{Y}^i(f))$, which is $O(m + \sum_{f=1}^l \mathfrak{Y}^1(f))$ for the second shortest path. Lemma 1 shows that $\mathbf{E}[\sum_{f=1}^l \mathfrak{Y}^1(f)] = O(n + m)$, making the average-case complexity of this algorithm to be $O(m)$.

Lemma 1. If $p \geq C \ln n / n$ for some constant $C > 2$, $\mathbf{E}[\sum_{f=1}^l \mathfrak{Y}^1(f)] = O(n + m)$.

Proof. We first show that $\sum_{f=1}^l n_y^1(f) + n_r^1(f) = O(n)$. Then we will argue that $\sum_{f=1}^l m_y^1(f) + m_r^1(f) = O(m)$.

Consider the i^{th} node $t(i)$ on the shortest path from s to t . We want to show that, in expectation, the number of nodes u such that the shortest path from s to u goes through $t(i)$ decreases roughly exponentially in i . To this end, we analyse the structure of the BFS tree rooted at s . Let Γ_i denote the set of nodes at distance i from s , thus for any $y \in \Gamma_{i+1}$ there exists $x \in \Gamma_i$ such that (x, y)

is an edge. We assume that the parent of y in the BFS tree is randomly chosen among all such x , if there is more than one possibility.

The depth of the BFS tree is $O(\log n)$ w.h.p. [28] and, by Lemma 8 from [8], for each $1 \leq i \leq i_0 = 2/3 \log n / \log(np)$, with probability at least $1 - o(1/n)$, we have $|\Gamma_i| = \Omega((np)^i)$. We would like to have a lower bound on $|\Gamma_i|$ for $i > i_0$ as well. We choose the smallest i_1 such that $|\bigcup_{i=0}^{i_1} \Gamma_i| > n - 2\sqrt{n}$. We claim that now, for any $i = i_0 + 1, \dots, i_1 - 1$, $|\Gamma_i| \geq \sqrt{n}$ with probability $1 - 1/n$. Assume that $|\Gamma_i| < \sqrt{n}$, then we have a partition of the nodes into three sets U, S and V with the following properties: $|S| < \sqrt{n}$, $|U|, |V| \geq 2\sqrt{n}$, and there are no edges between U and V . But the probability of such a partition to exist can be bounded for $|U| = x$ as follows:

$$\begin{aligned} \binom{n}{\sqrt{n}} \binom{n}{x} (1-p)^{x(n-\sqrt{n}-x)} &\leq (\sqrt{n}e)^{\sqrt{n}} (ne/x)^x e^{-p \cdot x(n-\sqrt{n}-x)} \\ &\leq e^{\sqrt{n} \ln n - x \ln x + x - x \ln n + 2c \ln n} \leq e^{\sqrt{n} \ln n - x \ln x + (2c - x/2) \ln n} \end{aligned}$$

which is at most $1/n^2$ for sufficiently large n and $x \geq 2\sqrt{n}$. Summing over all possible $|U|$, the probability is bounded by $1/n$.

We rephrase the random procedure used to create the graph. Instead of fixing every edge, we work in phases corresponding to the layers of the BFS tree. Having chosen Γ_i , we select (with appropriate probability) the nodes of Γ_{i+1} , but do not fix the edges between $x \in \Gamma_i$ and $y \in \Gamma_{i+1}$ yet. Also, we terminate after having obtained Γ_{i_1} , thus there might be still up to $2\sqrt{n}$ unvisited nodes. To fully determine the relevant part of the BFS tree we still need to choose, for every $y \in \Gamma_{i+1}$, for $i = 0, 1, \dots, i_1 - 1$, its parent $x \in \Gamma_i$. We assume that $u \in \Gamma_{i_1}$.

We need to bound the probability that, for a randomly chosen node u , the shortest path from s to u goes through $t(i)$. By union bound, this is at most $1/|\Gamma_i| + 1/|\Gamma_{i+1}| + \dots + 1/|\Gamma_{i_1-1}|$, because for the shortest path from s to u and s to t to meet for the first time in some node $x \in \Gamma_j$ it must hold that two nodes $y, y' \in \Gamma_{j+1}$ have chosen the same parent, which holds with probability $1/|\Gamma_j|$. Summing over all i we obtain that w.h.p. the expected number of such nodes u is

$$\begin{aligned} \sum_{i=0}^{i_1-1} n \sum_{j=i}^{i_1-1} 1/|\Gamma_j| &\leq O(n \log^2 n / \sqrt{n}) + \sum_{i=0}^{i_0} \sum_{j=i}^{i_0} n/(np)^j \\ &\leq O(n \log^2 n / \sqrt{n}) + \sum_{i=0}^{i_0} n/(np)^i = O(n). \end{aligned}$$

It remains to argue that $\sum_{f=1}^l m_y^1(f) + m_r^1(f) = O(m)$. The expected degree of a node is $np \geq c \ln n$, so by standard Chernoff bounds with probability $1 - 1/n$ the degree of every node is $O(np)$. Thus, $\sum_{f=1}^l m_y^1(f) + m_r^1(f) = O(n^2 p)$ with probability $1 - 1/n$. Finally, the expected value of m is $n^2 p/2$, so by Chernoff bounds m is less than $n^2 p/4$ with probability at most $1/n$ for sufficiently large n . Thus, with probability $1 - O(1/n)$, $\sum_{f=1}^l m_y^1(f) + m_r^1(f) = O(n^2 p) = O(m)$.

5.1 Empirical Results on Feng's Average-Case Behavior

Theorem 3 proves that the average-case complexity of computing second shortest path using Feng's algorithm is $O(n + m)$ for $p \geq c \ln n/n$. This is based on Theorem 1 that shows that in this setting, $\mathfrak{Y}^i(f)$ grows exponentially as a function of the relative distance of deviation node to target node and $\sum_{f=1}^l \mathfrak{Y}^i(f) = O(n + m)$. In this section, we show experimental results which indicate that the results are likely to be true for the (i) k^{th} shortest path for $k > 2$ and (ii) for sparser graphs.

The experimental setting is similar to that of the empirical analysis of Yen's algorithm. Consider the computation of candidate deviation paths from the i^{th} shortest path with $\#hops$ hops. Let f be the deviation node as numbered from t ($f = 0$ for s and $f = l - 1$ for the predecessor of t). Figure 3 shows that $\mathfrak{Y}^i(f)$ grows exponentially the closer the deviation node is to the target node (so while f/l gets closer to 1), even when the results are averaged over 10 shortest paths (and not just for the second shortest path). This is true both for unweighted and uniformly random edge weight setting.

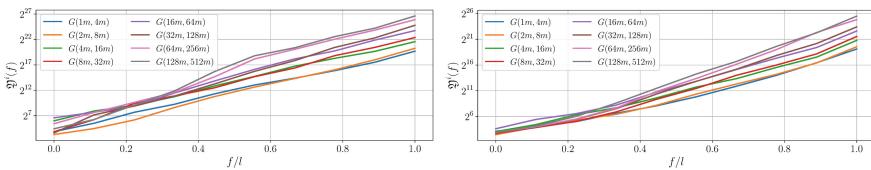


Fig. 3. $\mathfrak{Y}^i(f)$ averaged over the first $k = 10$ shortest paths for $G(n, 4n)$ with uniform random edge weights (left) and unweighted (right). Since all of the k -shortest paths have a different number of hops, the size of the yellow graph is plotted with respect to the number of hops.

Finally, Fig. 4 shows that $\sum_{f=1}^l \mathfrak{Y}^i(f)/(n + m)$ is close to being a constant, even for larger values of k (for both unweighted and uniformly random edge weights). Again, this suggests that Theorem 1 is likely to hold for $k > 2$ as well.

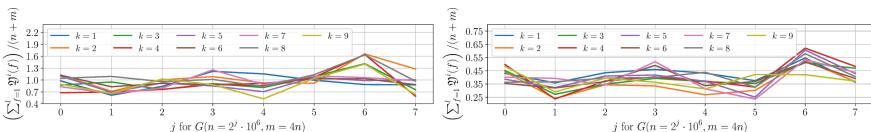


Fig. 4. $(\sum_{f=1}^l \mathfrak{Y}^i(f)) / (n + m)$, the total size of all yellow graphs with respect to n and m during the computation of the $(k + 1)^{th}$ shortest path for $G(n, 4n)$ with uniform random edge weights (left) and unweighted (right).

6 Conclusion

We have shown that in the not-too-sparse setting, the average-case complexity of Yen's algorithm is $O(k m \log n)$ in contrast to its worst-case complexity of $O(k m n)$. For Feng's algorithm, we found that the considerable pruning of the subgraphs for computing deviations is not only a fast heuristic in practice, but it also enables the proof of a better average-case complexity bound of $O(k m)$ for the second shortest path on unweighted graphs. Proving that this bound still holds for $k > 2$ and in the sparser settings remains an open combinatorial problem.

Acknowledgements. We are grateful to Erika Duriakova for providing us the code for the implementation of Yen's algorithm, the SSSP subroutines and her generous help with debugging our usage of her code.

References

1. Agarwal, U., Ramachandran, V.: Finding k simple shortest paths and cycles. In: Proceedings of the 27th ISAAC, LIPIcs, pp. 8:1–8:12 (2016)
2. Agarwal, U., Ramachandran, V.: Fine-grained complexity for sparse graphs. Proceedings of the 50th STOC, pp. 239–252. ACM (2018)
3. Ajwani, D., Duriakova, E., Hurley, N., Meyer, U., Schickendantz, A.: An empirical comparison of k -shortest simple path algorithms on multicores. In: Proceedings of the 47th ICPP (2018)
4. Akiba, T., Hayashi, T., Nori, N., Iwata, Y., Yoshida, Y.: Efficient top- k shortest-path distance queries on large networks by pruned landmark labeling. Proceedings of the 29th AAAI, pp. 2–8 (2015)
5. Bernstein, A.: A nearly optimal algorithm for approximating replacement paths and k shortest simple paths in general graphs. In: Proceedings of the 21st SODA, pp. 742–755 (2010)
6. Bernstein, A., Karger, D.R.: A nearly optimal oracle for avoiding failed vertices and edges. In: Proceedings of the 41st STOC, pp. 101–110 (2009)
7. Bollobás, B.: Models of Random Graphs. Cambridge Studies in Advanced Mathematics, 2nd edn, pp. 34–59. Cambridge University Press, Cambridge (2001)
8. Chung, F., Lu, L.: The diameter of sparse random graphs. Adv. Appl. Math. **26**(4), 257–279 (2001). <https://doi.org/10.1006/aama.2001.0720>
9. Clarke, S., Krikorian, A., Rausen, J.: Computing the N best loopless paths in a network. J. Soc. Ind. Appl. Math. **11**(4), 1096–1102 (1963)
10. Demetrescu, C., Thorup, M., Chowdhury, R.A., Ramachandran, V.: Oracles for distances avoiding a failed node or link. SIAM J. Comput. **37**(5), 1299–1318 (2008)
11. Dijkstra, E.W.: A note on two problems in connexion with graphs. Numer. Math. **1**(1), 269–271 (1959)
12. Eppstein, D.: Finding the k shortest paths. SIAM J. Comput. **28**(2), 652–673 (1998)
13. Feng, G.: Finding k shortest simple paths in directed graphs: a node classification algorithm. Networks **64**(1), 6–17 (2014)
14. Fredman, M.L., Tarjan, R.E.: Fibonacci heaps and their uses in improved network optimization algorithms. J. ACM (JACM) **34**(3), 596–615 (1987)

15. Frieder, A., Roditty, L.: An experimental study on approximating k shortest simple paths. *J. Exp. Algorithmics (JEA)* **19**, 1–5 (2015)
16. Goldberg, A.V.: A practical shortest path algorithm with linear expected time. *SIAM J. Comput.* **37**(5), 1637–1655 (2008)
17. Gotthilf, Z., Lewenstein, M.: Improved algorithms for the k simple shortest paths and the replacement paths problems. *Inf. Process. Lett.* **109**(7), 352–355 (2009)
18. Graham, A.J., Pike, D.A.: A note on thresholds and connectivity in random directed graphs. *Atl. Electron. J. Math.* **3**(1), 1–5 (2008)
19. Hershberger, J., Maxel, M., Suri, S.: Finding the k shortest simple paths: a new algorithm and its implementation. *ACM Trans. Algorithms (TALG)* **3**(4), 45 (2007)
20. Karp, R.M.: The transitive closure of a random digraph. *Random Struct. Algorithms* **1**(1), 73–94 (1990)
21. Katoh, N., Ibaraki, T., Mine, H.: An efficient algorithm for k shortest simple paths. *Networks* **12**(4), 411–427 (1982)
22. Kurz, D., Mutzel, P.: A sidetrack-based algorithm for finding the k shortest simple paths in a directed graph. *CoRR/arXiv:abs/1601.02867* (2016)
23. Lawler, E.L.: A procedure for computing the k best solutions to discrete optimization problems and its application to the shortest path problem. *Manag. Sci.* **18**(7), 401–405 (1972)
24. Martins, E., Pascoal, M.M., Santos, J.: Deviation algorithms for ranking shortest paths. *Int. J. Found. Comput. Sci.* **10**(3), 247–262 (1999)
25. Martins, E.Q., Pascoal, M.M.: A new implementation of Yens ranking loopless paths algorithm. *J. Belg. Fr. Ital. Oper. Res. Soc.* **1**(2), 121–133 (2003)
26. Meyer, U.: Average-case complexity of single-source shortest-paths algorithms: lower and upper bounds. *J. Algorithms* **48**(1), 91–134 (2003)
27. Perko, A.: Implementation of algorithms for k shortest loopless paths. *Networks* **16**(2), 149–160 (1986)
28. Priebe, V.: Average-case complexity of shortest-paths problems. Ph.D. thesis, Saarland University (2001). <http://scidok.sulb.uni-saarland.de/volltexte/2007/1180/>
29. Roditty, L., Zwick, U.: Replacement paths and k simple shortest paths in unweighted directed graphs. *ACM Trans. Algorithms (TALG)* **8**(4), 33 (2012)
30. Sedeño-Noda, A.: An efficient time and space K point-to-point shortest simple paths algorithm. *Appl. Math. Comput.* **218**(20), 10244–10257 (2012)
31. Shih, Y.K., Parthasarathy, S.: A single source k -shortest paths algorithm to infer regulatory pathways in a gene network. *Bioinformatics* **28**(12), i49–i58 (2012)
32. Williams, V.V., Williams, R.: Subcubic equivalences between path, matrix and triangle problems. In: Proceedings of the 51th FOCS, pp. 645–654 (2010)
33. Yen, J.Y.: Finding the k shortest loopless paths in a network. *Manag. Sci.* **17**(11), 712–716 (1971)



Scaling of Random Walk Betweenness in Networks

Onuttom Narayan¹ and Iraj Saniee^{2(✉)}

¹ University of California, 1156 High Street, Santa Cruz, CA 95064, USA
onarayan@ucsc.edu

² Bell Labs, Nokia, 600 Mountain Avenue, Murray Hill, NJ 07974, USA
iraj.saniee@nokia-bell-labs.com

Abstract. We study the betweenness centrality (BC) of vertices of a graph using random walk paths. Random walk BC (RWBC) provides an alternative measure to the shortest path centrality of each vertex in a graph as it aggregates contributions from possibly all vertex-pairs in the graph and not just from those vertex-pairs on whose geodesic path the vertex lies. As such, RWBC is more relevant in the context of information diffusion in virtual networks, such as spread of news or rumors in online social media. We derive a closed-form analytical expression for RWBC using eigenfunctions of the graph Laplacian. We then show the distribution of RWBC scores of the vertices of a graph exhibits a scaling collapse with no adjustable parameters as the graph size N is varied. This means that the distribution of RWBC over all the nodes in a large graph can be obtained in terms of the distribution of RWBC for a prototypical or small graph that is generated using the same model. The exact distribution itself depends on the graph model. A normalized random walk betweenness (NRWBC), that counts each walk passing through a vertex only once, is also defined. This measure is argued to be more useful and robust and is seen to have simpler scaling behavior.

1 Introduction

In physical networks data flow is mostly along shortest or geodesic paths, packet networks being among such networks. This is a natural consequence of minimizing the capacity of the network to meet end to end demand. The relevance or importance of a vertex is then determined by the relative volume of geodesic paths that traverse that vertex. The geodesic betweenness centrality of a vertex has been defined to measure this frequency, volume or importance. In the far more prevalent case of virtual networks, such as friendship, information, co-authorship, biological or rumor networks, however, flows do not move only along geodesic paths. Measures of importance based only on geodesic paths can distort the importance of the graph vertices. Freeman [1] and Newman [2], and more recently Kivimäki et al. [3], among others, consider alternatives that measure centrality based on random walks.

To be specific, we consider a graph with N vertices, with the same rate of traffic flow between all $N(N-1)$ possible source-destination pairs. The dynamics

are discrete time. At every time step, one unit of traffic is injected at each vertex for each other vertex as destination. Any unit of traffic that was already in the graph moves randomly with equal probability to one of the adjoining vertices. If a unit reaches its destination, it is removed from the network at the next time step. The network is assumed to consist of one connected component.

Random walks on networks have been studied earlier using the graph Laplacian [4,5] with methods similar to those we will use in this paper, but the quantities studied before are different. In particular, Ref. [5] defines a random walk centrality by computing the average time to travel from a source to a destination vertex, and the change when the source and destination are reversed. Reference [4] investigates the spectral gap of the graph Laplacian. We use two alternative definitions in this paper.

With random walk routing, it is no longer definite whether a path from a source to a destination will pass through some vertex; each vertex has a probability of being on the path. Also, the load and the betweenness centrality are no longer equivalent. This is because a random walk wandering through the network can pass through a vertex several times. While it is appropriate to count each of these traversals as contributing one unit to the load at the vertex, it is unreasonable to consider them as each adding to the betweenness of the vertex.

In a separate paper [6], we have shown that the load at each vertex with random walk routing is linearly dependent on the degree of the vertex, with a proportionality constant $N \sum_{\alpha} 1/\lambda_{\alpha}$ involving the sum of the inverses of the non-zero eigenvalues of the Laplacian on the graph. We computed how the proportionality constant scales with N for various network models.

In this paper, we study two distinct measures for the importance of vertices of a graph. Random walk betweenness centrality (RWBC) provides an alternative measure to the widely-used shortest path betweenness centrality of each vertex in a graph as it aggregates, with due regard to their probabilities, contributions from possibly all vertex-pairs in the graph and not just from those vertex-pairs on whose geodesic path the vertex lies. The scaling behavior of RWBC is shown to have a simple form for some basic graph models. We also study a normalized random walk betweenness (NRWBC) measure that counts each walk passing through a vertex only once. This measure is more useful, more robust and is seen to have even a simpler scaling behavior for many graph models. The rationale for such a measure is the intuitive importance score of intermediate vertices in social graphs. Information initiated by a source A and received via a random path to destination Z , corroborates the participation of intermediate vertices on the random path but we only need the list of such vertices to give them credit. Repeated cycles through the same set of intermediaries do not add extra value to those intermediate vertices. Our goal is to account for the intermediate vertices' participation, which counts only once.

In the next section of this paper, we start from the definition of the random walk betweenness due to Newman [7], RWBC. We obtain an expression for this quantity in terms of the eigenfunctions of the Laplacian, from which we numerically obtain the distribution of random betweenness as a function of the number of vertices N in the network. By mapping the random walk problem to current

flowing in an electrical circuit, a prescription to achieve a scaling collapse of the distribution is obtained and verified for simple lattice graphs such as square and triangular lattices. This relies on the fact that the continuum limit for current flow on these graphs is diffusion on a plane. Surprisingly, the same prescription works for Erdos Renyi [8] and extended Barabasi Albert [9] graphs even though there is no underlying continuum limit. Thus for all the graph models, a scaling collapse is obtained with no adjustable parameters, using only the measured average distance between vertex pairs l_N in a N -vertex graph.

In Sect. 3 we present an alternative definition of the random walk betweenness, NRWBC, for which a random walk contributes only once to the betweenness of every vertex it passes through, regardless of the number of times it does so. With this more natural, and arguably more useful definition, an even simpler scaling collapse is obtained for the betweenness distribution as a function of the graph size N .

2 Random Walk Betweenness

As in Ref. [7], we consider the random walk process described in the Introduction: with discrete time dynamics, one random walker is injected into the network at time $t = 1, 2, 3 \dots$ at each source vertex for each other destination vertex; thus there are $(N - 1)$ walkers injected at each vertex at each time step. Any walker that is present at vertex i at time t is removed from the network at time $t + 1$ if i is its destination. If not, one of the neighbors of i is chosen randomly, and the walker moves there at time $t + 1$. The probability of choosing each of the neighbors of i is $1/d_i$, where d_i is the degree of the i 'th vertex. Note that a walker that returns to its source as it moves around randomly continues as it would from any other vertex.

For any given source vertex k and destination vertex l , the *net* time-averaged current through each of the edges connected to a vertex i is computed, and the magnitudes of all these are added. After adding the magnitudes of the net currents for the edges connecting to vertex i , the result is averaged over all source vertices k and destination vertices l to define the random walk betweenness of i . With this definition, if a random walk that reaches vertex i moves out at the next time step to vertex j through the edge (ij) , and returns to i through the edge (ji) at a later time, the outward flow along (ij) and the return along (ji) cancel each other. However, if the random walk leaves the vertex i along the edge (ij) and returns later through a different edge (ki) , the two do not cancel out, but are instead added. Although this definition only partially cancels the effect of a random walk looping through a vertex multiple times, it has the advantage that it can be mapped to an electrical circuit and solved using Kirchoff's laws [7].

To obtain an analytical expression for the random walk betweenness, we follow the approach of Ref. [6]. For the source k and destination l , let p_i^{kl} be the number of walkers at vertex i in steady state, averaged over all the random paths that the walkers can take. Let A_{ij} be the adjacency matrix of the graph. Then

$$p_i^{kl} = \delta_{ik} - \delta_{il} + \sum_j A_{ij} \frac{p_j^{kl}}{d_j} \quad (1)$$

for all i , where $\delta_{i\neq k} = 0; \delta_{ii} = 1$ is the Kronecker delta function. This is a degenerate set of equations: if p_i^{kl} is a solution to the equation, so is $p_i^{kl} + ad_i$ for any a . The condition $p_l^{kl} = 0$ breaks the degeneracy.

We define $p_i^{kl} = d_i r_i^{kl}$ and $L_{ij} = d_i \delta_{ij} - A_{ij}$. Then

$$\sum_j L_{ij} r_j^{kl} = \delta_{ik} - \delta_{il} \quad (2)$$

with $r_l^{kl} = 0$. Here L_{ij} is the negative Laplacian for the graph, with a complete set of eigenvectors with non-negative eigenvalues. Since the graph has been assumed to have one component, there is only one zero eigenvalue λ_0 with normalized eigenvector $\xi^0 = (1, 1, 1, \dots, 1)/\sqrt{N}$. One can verify that

$$r_j^{kl} = \sum_{\alpha=1}^{N-1} \frac{\xi_k^\alpha - \xi_l^\alpha}{\lambda_\alpha} \xi_j^\alpha + c^{kl} \xi_j^0, \quad (3)$$

where c^{kl} has to be determined from the condition $r_l^{kl} = 0$. The net inflow to any vertex i from a neighbor j is then $A_{ij}(r_j^{kl} - r_i^{kl})$. Adding the magnitudes of the currents along all the edges attached to the vertex i (with an extra factor of half),

$$b_i^{kl} = \frac{1}{2} \sum_j A_{ij} |r_i^{kl} - r_j^{kl}| + \frac{1}{2} (\delta_{ik} + \delta_{il}). \quad (4)$$

The random walk betweenness of the vertex i is then obtained by averaging over all possible sources and destinations

$$b_i = \frac{1}{2N(N-1)} \sum_{k\neq l} \sum_j A_{ij} |r_i^{kl} - r_j^{kl}| + \frac{1}{N}. \quad (5)$$

Using Eq. (3), since $\xi_j^0 = \xi_i^0$, this yields

$$b_i = \frac{1}{2N(N-1)} \sum_{k\neq l} \sum_j A_{ij} \sum_{\alpha\neq 0} \left| (\xi_k^\alpha - \xi_l^\alpha) \frac{1}{\lambda_\alpha} (\xi_i^\alpha - \xi_j^\alpha) \right| + \frac{1}{N}. \quad (6)$$

We define the operator $M = L + P$, where P is the projection operator onto the zero eigenvector of the Laplacian: $P_{ij} = 1/N$. Then

$$[M^{-1}]_{ij} = \sum_{\alpha} \xi_i^\alpha \frac{1}{\lambda_\alpha + \delta_{\alpha 0}} \xi_j^\alpha = \sum_{\alpha\neq 0} \xi_i^\alpha \frac{1}{\lambda_\alpha} \xi_j^\alpha + \frac{1}{N} \quad (7)$$

and therefore

$$b_i = \frac{1}{N(N-1)} \sum_{k>l} \sum_j A_{ij} |M_{ki}^{-1} - M_{kj}^{-1} - M_{li}^{-1} + M_{lj}^{-1}| + \frac{1}{N}. \quad (8)$$

Numerical results are obtained for various models using Eq. (8).

2.1 Numerical Results

We first consider square and triangular lattice graphs with overall shapes that are square and triangular respectively. Thus the square lattice graphs have $L \times L$ vertices and the triangular lattice graphs have $L(L + 1)/2$ vertices, with various values of L . The random walk betweenness of all the vertices in a graph are sorted in decreasing order.

As mentioned near the beginning of this section, there is a direct mapping between the RWBC and the flow of current in an electric circuit [7]. Exploiting this mapping, we consider the case of current flow on a continuum surface with uniform resistivity, when the current is injected from the outside at one point and extracted at another point. The surface current density $\mathbf{K}(\mathbf{r})$ as a function of the position on the surface \mathbf{r} has a unique solution from the continuum limit of Kirchoff's laws. If one measures the current flowing across any line segment on the surface, one obtains a unique result. Now if we discretize the surface using a square or triangular lattice with a very fine mesh, we expect that a continuum limit should exist in the limit as the mesh size tends to zero. If the linear dimension of the mesh is ϵ , the current that was flowing through a continuum line segment now flows through $O(1/\epsilon)$ edges, i.e. the current through any edge is proportional to ϵ . Since the number of nodes in the discretization of the surface is proportional to $1/\epsilon^2$, this is equivalent to the current through an edge being proportional to $1/\sqrt{N}$. More generally, for a graph that can be viewed as a discretization of a d -dimensional continuum space, the RWBC of the edges should be proportional to $N^{1/d-1}$. If we sort all the edges of the graph according to their RWBC, the i 'th node in the sorted list will have a betweenness centrality of the form

$$b_i = N^{1/d-1} \tilde{b}(i/N). \quad (9)$$

This means that, for instance, if one considers the RWBC of the middle node in the sorted list, or at the x 'th percentile for any given value of x , the RWBC as a function of N is proportional to $N^{1/d-1}$. Equivalently, if one plots $N^{1-1/d} b_i$ as a function of i/N , then the resultant curves for all values of N should lie on top of each other. The practical significance of this “scaling collapse” is that, if one is interested in the distribution of RWBC for an extremely large graph from a family of graphs, it can be obtained from the corresponding distribution of a much smaller graph from the same family, simplifying the computational task. Note that there are no adjustable parameters at our disposal when verifying if Eq. (9) is satisfied; the curves for various values of N either lie on top of each other, or they do not.

Figure 1 shows this scaling collapse for the square and triangular lattices (with $d = 2$ and $N^{1-1/d} = \sqrt{N}$). The fact that all the curves for square lattices of various N lie on top of each other, and similarly all the curves for triangular lattices, validates Eq. (9), although we cannot predict the functions $\tilde{b}(i/N)$ for square and triangular lattices which determine the shape of the two curves shown in Fig. 1.

Even though most graph models cannot be viewed as a discretization of a continuum manifold, one can optimistically generalize Eq. (9) to hypothesize that

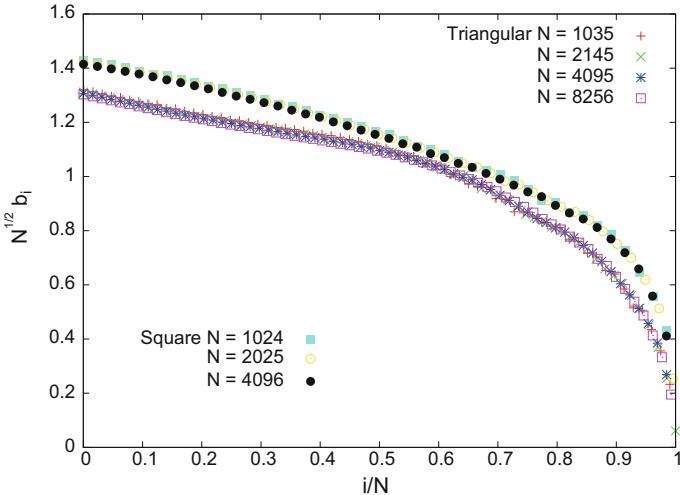


Fig. 1. Scaling collapse of the sorted random walk betweenness for square and triangular lattice graphs. The form in Eq. (9) is used. The vertical axis is multiplied by 1.05 for all the square lattice plots in order to separate the curves for the two lattices.

if one sorts the nodes in a graph according to their RWBC, then the RWBC for the i 'th sorted node in a graph with N nodes has the functional form

$$b_i = \frac{l_N}{N} \hat{b}(i/N). \quad (10)$$

where l_N is the average distance between vertex pairs in the graph and \hat{b} is a function that depends on the graph model. This is a conjecture for which—unlike for graphs which are discretizations of continuum manifolds—there is no proof. As with Eq. (9) for the square and triangular lattice, Eq. (10) is validated by plotting $N b_i / l_N$ as a function of i/N for various N 's, and verifying that the resultant curves are all superimposed.

Empirically, Eq. (10) is found to be satisfied for Erdos Renyi [8] random graphs in the sparse regime (average nodal degree $d_a \sim O(1)$) and in the dense regime ($d_a \sim \ln N$), as seen in Fig. 2. The same is true for preferential attachment graphs [9, 10], where each vertex is born with m edges that link to preexisting vertices, and the probability of linking to a preexisting vertex of degree j is $k+j$, with (m, k) parameters of the model. This is seen in Fig. 3.

We note in passing that Eq. (10) implies that the maximum nodal betweenness is $b_{max} \sim l_N / N \hat{b}(i/N \rightarrow 0)$. For the square and triangular lattices, $b(i/N \rightarrow 0)$ is a constant and $b_{max} \sim 1/\sqrt{N}$. For preferential attachment graphs, $\hat{b}(1/N)$ is seen from Fig. 3 to have a power-law form, so that—ignoring the weak N dependence of l_N — b_{max} scales as a power of N , which is found to only depend on k/m . The situation is more complicated for Erdos Renyi graphs, with logarithmic corrections.

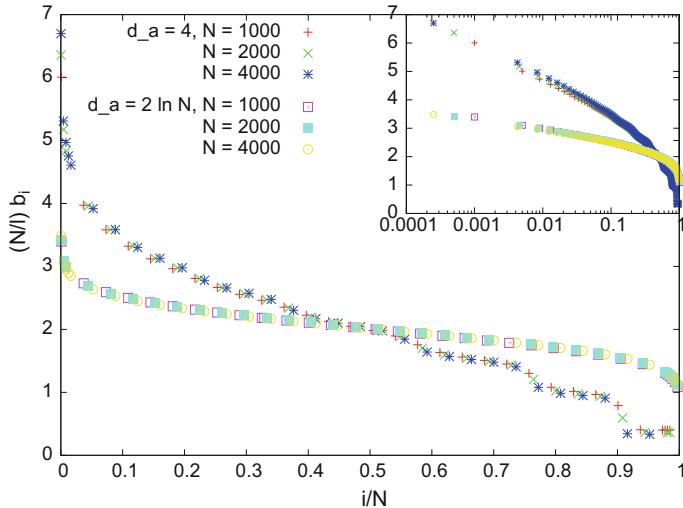


Fig. 2. Scaling collapse of the sorted random walk betweenness for Erdos Renyi graphs in the sparse and dense regimes. Equation (10) is used to construct the scaling plot, with l_N measured from the graphs; there are no adjustable parameters. The results from eighty random graphs have been averaged for each set of points. The inset shows the same distributions but with a log scale on the x axis.

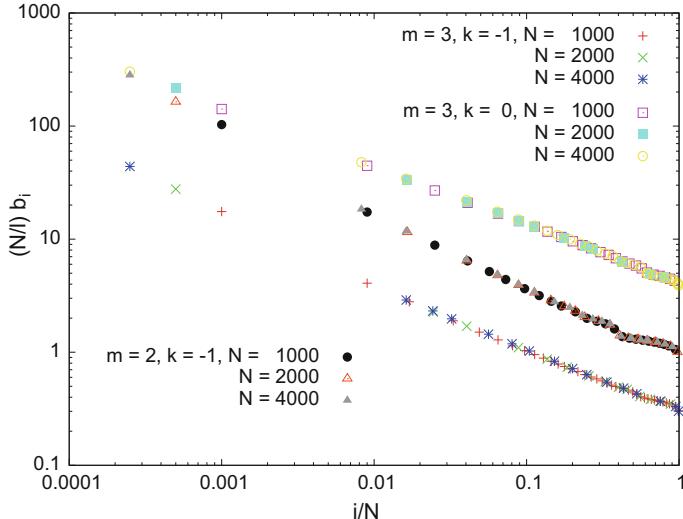


Fig. 3. Scaling collapse of the sorted random walk betweenness of all the vertices in a scale-free graph. The results from eighty random graphs have been averaged for each set of points. The three scaling plots correspond to $(m, k) = (3, 0), (2, -1)$ and $(3, -1)$. In order to separate the plots, the betweenness in the first and third plots is divided and multiplied by 3.6 respectively.

3 Normalized Random Walk Betweenness

The random walk betweenness defined in Ref. [7] and studied in the previous section ensures that if a random walk goes back and forth between two vertices, this does not increase their betweenness. However, if the walk goes round a loop repeatedly, the betweenness of all the vertices in the loop is increased by unity for each round-trip. One might instead define betweenness so that each random walk contributes only once to the vertices it passes through. To avoid confusion, we call this quantity the normalized random walk betweenness.

At every time step, one walker is injected into the network at the vertex k . If the walker reaches the destination vertex l or the query vertex m , it disappears from the network at the next time step. The rate at which walkers are destroyed at the vertex m is the probability that a random walker from k to l will pass at least once through m . In steady state, we have

$$p_i^{kl} = \delta_{ik} - (1 - \mu_m^{kl})\delta_{il} - \mu_m^{kl}\delta_{im} + \sum_j A_{ij} \frac{p_j^{kl}}{d_j} \quad (11)$$

where μ_m^{kl} is the rate at which random walkers from k to l reach m (for the first time), which has to be determined self-consistently. Equation (11) comes with the boundary conditions $p_l^{kl} = p_m^{kl} = 0$.

With $p_i^{kl} = d_i r_i^{kl}$, we obtain the solution

$$r_j^{kl} = \sum_{\alpha=1}^{N-1} \frac{\xi_k^\alpha - (1 - \mu_m^{kl})\xi_l^\alpha - \mu_m^{kl}\xi_m^\alpha}{\lambda_\alpha} \xi_j^\alpha + c^{kl} \xi_j^0. \quad (12)$$

Applying the condition $r_m^{kl} - r_l^{kl} = 0$, we obtain

$$\sum_{\alpha=1}^{N-1} \frac{\xi_k^\alpha - (1 - \mu_m^{kl})\xi_l^\alpha - \mu_m^{kl}\xi_m^\alpha}{\lambda_\alpha} (\xi_m^\alpha - \xi_l^\alpha) = 0 \quad (13)$$

which fixes μ_m^{kl} :

$$\sum_{\alpha=1}^{N-1} \frac{\xi_k^\alpha - \xi_l^\alpha}{\lambda_\alpha} (\xi_m^\alpha - \xi_l^\alpha) = \mu_m^{kl} \sum_{\alpha=1}^{N-1} \frac{(\xi_m^\alpha - \xi_l^\alpha)^2}{\lambda_\alpha}. \quad (14)$$

Note that $\mu_k^{kl} = 1$, as it should be. The expression for μ_l^{kl} is indeterminate; we fix it to be 1.

In order to obtain the normalized random walk betweenness n_m of the vertex m , we average μ_m^{kl} over all l and $k \neq l$, with m fixed. Thus

$$n_m = [\sum_{l \neq m} \sum_{k \neq l} \mu_m^{kl} + (N-1)]/(N(N-1)). \quad (15)$$

In the numerator, the sum over k can be made unrestricted, because $\mu_m^{ll} = 0$ if $l \neq m$. Since $\sum_k \xi_k^\alpha = 0$ for $\alpha \neq 0$, we have

$$\sum_k \mu_m^{kl} = N \left[\sum_{\alpha \neq 0} \frac{(\xi_l^\alpha - \xi_m^\alpha) \xi_l^\alpha}{\lambda_\alpha} \right] \left[\sum_{\alpha \neq 0} \frac{(\xi_l^\alpha - \xi_m^\alpha)^2}{\lambda_\alpha} \right]^{-1}. \quad (16)$$

Therefore

$$n_m = \frac{1}{N-1} \sum_{l \neq m} \left[\sum_{\alpha \neq 0} \frac{(\xi_l^\alpha - \xi_m^\alpha) \xi_l^\alpha}{\lambda_\alpha} \right] \left[\sum_{\alpha \neq 0} \frac{(\xi_l^\alpha - \xi_m^\alpha)^2}{\lambda_\alpha} \right]^{-1} + \frac{1}{N}. \quad (17)$$

Using Eq. (7) we have

$$n_i = \frac{1}{N} + \frac{1}{N-1} \sum_{l \neq i} \frac{M_{il}^{-1} - M_{li}^{-1}}{M_{ii}^{-1} + M_{ii}^{-1} - 2M_{li}^{-1}} \quad (18)$$

where we have replaced the subscript m with i to match the expression for the random walk betweenness b_i in Sect. 2.

3.1 Numerical Results

Equation (18) was used to numerically evaluate the normalized random walk betweenness n_i for various models. As in Sect. 2, the n_i values for all the vertices in a graph were sorted. For the random graph models, the sorted list was averaged over eighty realizations of the random graph. Figure 4 shows the normalized random walk betweenness for preferential attachment networks, the Erdos Renyi model in the sparse regime, and lattice graphs. For all these cases, n_i is only a function of i/N . For clarity, the figure only shows one example of each class of graph models, but a similar data collapse (to different curves) is seen if the model parameters are varied. For the Erdos Renyi model in the dense regime, the scaling collapse is only approximate.

In Fig. 4, we see that the maximum normalized betweenness is close to 1 for the preferential attachment graphs. In fact, as seen in Fig. 5, $n_{max} \rightarrow 1$ as $N \rightarrow \infty$ for these models. This may seem surprising: $n_{max} = 1$ implies that essentially *all* the random walks pass through some vertex. When shortest path routing is used instead of random walks, this is impossible, and one might expect even less ‘focusing’ on any vertex with random walk routing. To understand how this could happen, we consider a simpler graph: a regular tree that descends h levels from a root vertex. The maximum normalized betweenness is of the root vertex. A random walk from a source vertex k to a destination vertex l must pass through their common ancestor m . The probability that the random walk thereafter reaches the origin before reaching the vertex l is $1 - d(0, m)/d(0, l)$, where $d(i, j)$ is the shortest path distance between vertices i and j . For typical vertex pairs, this is $1 - O(1/\ln N)$. It is intriguing that the root vertex is even more central for random walks on preferential attachment graphs than on trees, with $1 - n_{max} \sim 1/N^c$ (see Fig. 5).

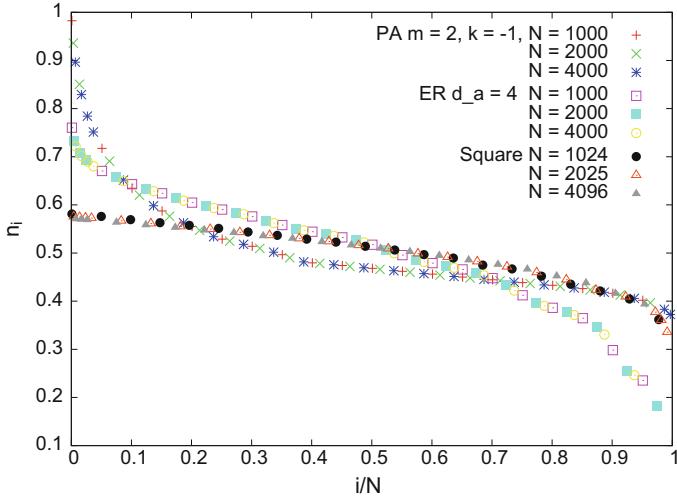


Fig. 4. Normalized random walk betweenness for various graph models. The vertices are sorted according to decreasing betweenness. Results for preferential attachment graphs with $m = 2, k = -1$, Erdos Renyi graphs with $d_a = 4$, and square lattice graphs of various sizes are shown. Except for the square lattices, each data point represents the average of eighty random graphs.

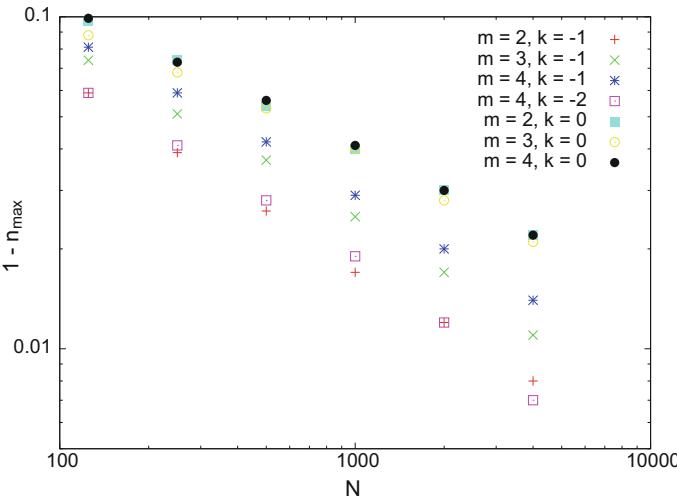


Fig. 5. Deviation from unity of the maximum normalized random walk betweenness, n_{max} , for preferential attachment graphs with various parameters. The plots show that $1 - n_{max}$ decreases as the size of the graph N is increased as $\sim 1/N^c$, with the exponent c depending on the ratio of model parameters, k/m .

4 Conclusions

To summarize, we have considered two definitions of the random walk betweenness of the vertices in a graph: an unnormalized version which can be represented in terms of currents flowing in an electrical circuit, and a normalized version which does not measure the number of passes for a random walk through a vertex but instead uses a binary measure that denotes passage. For both cases, we obtain a parameter free scaling collapse of the distribution of vertex betweenness as a function of the size N of the graph for several graph models. The scaling function is singular for scale-free graphs, resulting in the maximum unnormalized betweenness being of the form $\sim 1/N^a$ with nontrivial values for a . Although the scaling collapse can be understood for lattice graphs through a continuum limit, we show that it even works for random graph models for which there is no theoretical explanation of this result.

Acknowledgments. This work was supported by grants FA9550-11-1-0278 and 60NANB10D128 from AFOSR and NIST, respectively.

References

1. Freeman, L.C.: Centrality in social networks: conceptual clarification. *Soc. Netw.* **1**, 215–239 (1979)
2. Newman, M.E.J.: A measure of betweenness centrality based on random walks. *Soc. Netw.* **27**, 39–54 (2005)
3. Kivimäki, I., Lebichot, B., Saramäki, J., Saerens, M.: Two betweenness centrality measures based on randomized shortest paths. *Nat. Sci. Rep.* **6**, 19668 (2016)
4. Samukhin, A.N., Dorogovtsev, S.N., Mendes, J.F.F.: Laplacian spectra of and random walks on, complex networks: are scale-free architectures really important? *Phys. Rev. E* **77**, 036115 (2008)
5. Noh, J.D., Rieger, H.: Random walks on complex networks. *Phys. Rev. Lett.* **92**, 118701 (2004)
6. Narayan O., Sanjee I. and Marbukh V.: Congestion due to random walk routing (2013). <https://arxiv.org/abs/1309.0066>
7. Newman, M.E.J.: The structure and function of complex networks. *SIAM Rev.* **45**, 167–256 (2003)
8. Erdős, P., Rényi, A.: On random graphs. *Publ. Math.* **6**, 290–297 (1959)
9. Barabasi, A.-L., Albert, R.: Emergence of scaling in random networks. *Science* **286**, 509–512 (1999)
10. Dorogovtsev, S.N., Mendes, J.F.F., Samukhin, A.N.: Structure of growing networks with preferential linking. *Phys. Rev. Lett.* **85**, 4633–4636 (2000)



Fast Approximated Betweenness Centrality of Directed and Weighted Graphs

Angelo Furno^{1(✉)}, Nour-Eddin El Faouzi¹, Rajesh Sharma², and Eugenio Zimeo³

¹ University of Lyon, IFSTTAR, ENTPE, LICIT UMR.T9401, Lyon, France
{angelo.furno,nour-eddin.elfaouzi}@ifsttar.fr

² University of Tartu, Tartu, Estonia
rajesh.sharma@ut.ee

³ University of Sannio, Benevento, Italy
zimeo@unisannio.it

Abstract. Node betweenness centrality is a reference metric to identify the most critical spots of a network. However, its exact computation exhibits already high (time) complexity on unweighted, undirected graphs. In some domains such as transportation, weighted and directed graphs can provide more realistic modeling, but at the cost of an additional computation burden that limits the adoption of betweenness centrality for real-time monitoring of large networks. As largely demonstrated in previous work, approximated approaches represent a viable solution for continuous monitoring of the most critical nodes of large networks, when the knowledge of the exact values is not necessary for all the nodes.

This paper presents a fast algorithm for approximated computation of betweenness centrality for weighted and directed graphs. It is a substantial extension of our previous work which focused only on unweighted and undirected networks. Similarly to that, it is based on the identification of pivot nodes that equally contribute to betweenness centrality values of the other nodes of the network. The pivots are discovered via a cluster-based approach that permits to identify the nodes that have the same properties with reference to clusters' border nodes. The results prove that our algorithm exhibits significantly lower execution time and a bounded and tolerable approximation with respect to state-of-the-art approaches for exact computation when applied to very large, weighted and directed graphs.

Keywords: Betweenness centrality · Directed weighted graphs · Fast computation · Large scale networks · Real-time monitoring

1 Introduction

Real-time monitoring of large networks for detecting and predicting critical spots is a compelling challenge due to the high complexity of computing robustness metrics. Graph models have proven to be a valid approach to study topological bottlenecks of many kinds of networks via centrality indicators such as betweenness centrality. However, while undirected and unweighted graphs represent a basic abstraction of these networks, weighted graphs can better capture edge diversity, especially in some application domains, such as transportation.

Betweenness centrality (BC) [14] is a very popular network metric to characterize nodes that are most traversed by shortest paths connecting pairs of other nodes of a graph. It has been widely adopted to study many application domains but the high computation time limits its adoption for real-time monitoring of very large networks. The fastest algorithm to compute the exact value of betweenness centrality has been proposed by Brandes [9]. Given a graph $G(V, E)$, it exhibits $O(n + m)$ space and $O(nm)$ time complexities for unweighted graphs and $O(nm + n^2 \log(n))$ for weighted ones, where $n = |V|$ is the number of nodes and $m = |E|$ the number of edges. To achieve this performance, Brandes adopts a single-source shortest-paths (SSSP) algorithm based on breadth-first graph search or on Dijkstra algorithm for unweighted and weighted graphs, respectively. Each exploration is computed with a complexity $O(m)$ and $O(m + n \cdot \log(n))$, for unweighted and weighted graphs respectively. This is typically good for sparse graphs (where $m \ll n^2$) but not sufficient for real-time monitoring of very large networks. A faster approach, useful for some kinds of applications, allows achieving lower computation time by calculating approximated BC values. These strategies try to penalize some shortest paths or to exploit topological properties to identify only $k \ll n$ pivot nodes as sources for the computation of SSSP. While several attempts exist for computing approximated values of betweenness centrality of unweighted graphs, a few of them focus on directed and weighted graphs, which better model several real-world networks.

In this paper, we propose an adaptable algorithm for computing approximated values of BC of directed and weighted networks. The performance of the algorithm can be tuned based on the amount of error we can tolerate on the approximation. It is an extension of the algorithm originally proposed in [15, 16] for undirected and unweighted graphs to directed and weighted ones. The algorithm, similarly to the ones proposed in [10] and [18], exploits topological characteristics of the graph in order to classify nodes for their selection as pivots. Differently from the papers above, it exploits clustering to identify reference nodes (clusters' border nodes) to perform a topological analysis. The solution can calculate an almost exact value of betweenness centrality for several nodes, i.e., the most critical ones, while keeping a good approximation for the others, with an execution time that strictly depends on the number of retained pivots.

Our algorithm is validated by using real-world transportation networks. By assuming that edge weights represent the traveled distance or the free-flow-travel

time, betweenness centrality provides indications about redistributions of the traffic flow (or potential congestion), if we make the assumption that people prefer the shortest (fastest) paths to reach their destinations. Our analysis is based on a real large-scale road-network and Global Positioning System (GPS) taxi traces. We leverage geo-referenced, time-stamped taxi trips to reconstruct per-link median travel time/speed with a hourly frequency. Such traffic indicator is used as the edge weight in the modeling graph. However, it is not possible to estimate such traffic indicator for all network edges: some portions of the road networks are never traversed by taxis in the observed time period. Therefore, for such regions of the network, we used an interpolation technique based on supervised machine learning. The performance analysis shows that the proposed algorithm is a valid solution for real-time monitoring of large-scale graphs.

The rest of the paper is organized as follows. Sect. 2 presents related work, while Sect. 3 describes our fast BC algorithm. In Sect. 4, we evaluate our approach on a large-scale transportation dataset. Section 5 concludes the work and highlights future directions.

2 Related Work

Betweenness centrality, originally proposed in [13] for undirected graphs was extended to directed graph in [31]. Brandes in [9] proposed a faster algorithm which also works for weighted networks. The idea was based on the adoption of SSSP algorithm based on breadth-first graph search or on Dijkstra algorithm, for unweighted and weighted graphs respectively. Several approaches, aiming to evaluate exact or approximated solutions, have been developed to further reduce the computation time. The proposed solutions can be classified according to three main categories: (i) exploiting and increasing parallelism, (ii) estimating BC values through a partial exploration of the graph, (iii) calculating BC values of fraction of nodes in dynamically evolving graphs.

Parallel approaches: In [4], the first parallel implementation for computing betweenness centrality is presented, which handles weighted graphs as well. It is based on a fine-grained multi-level parallelism, in which the neighbors of a given node are traversed concurrently on a shared data structure with granular locking. The algorithm has been successively improved [23] by removing the need for locking in the dependency accumulation stage of Brandes' algorithm through the adoption of a successor list instead of a predecessor list for each node.

Incremental approaches: A different set of approaches (stream-based) tries to avoid recomputing the BC values of all the nodes of a graph $G' \equiv G + \Delta G$ when they are known for a previous configuration G . For example, in [21], researchers proposed an efficient approach that reduces the search space by focusing only on the vertices whose betweenness centralities get changed as a consequence of an update in the graph. Similarly, in [19], by using the hypergraph sketch data structure, i.e., a weighted hyper-edge representation of shortest paths, computation time was reduced. Based on sampling based techniques [28], Bergamini et al. first proposed a semi-dynamic [6] and, later, a fully dynamic approach [5] for dynamic networks (both weighted and unweighted), capable of performing

in-memory computation with millions of edges. Recently, an efficient algorithm for incremental BC computation [20] has been proposed. The algorithm exhibits good performance when the graph changes for a very limited number of nodes. Conversely, the high speedup drastically reduces when the graph is subject to significant changes of its topology.

Approximated approaches: The third research trend focuses on achieving low computation time by calculating approximated BC values. These strategies try to penalize some shortest paths, whose computation is the most expensive task in the whole process. For example, in [32], the authors only consider paths up to fixed length k . Brandes and Pich [10] also proposed an approximated algorithm for faster BC calculation by choosing only $k \ll n$ pivots as sources for the SSSP algorithm through different strategies, showing that random selection of pivots can achieve accuracy levels comparable to other heuristics. However, this approach overestimates the BC of unimportant nodes that are near a pivot. To overcome this problem, various solutions have been proposed, e.g., a generalization framework for betweenness approximation has been proposed in [18]. The idea is to scale BC values in order to reduce them with reference to nodes close to pivots. In another paper, a solution to reduce the pivots for nodes with high centrality is proposed via adaptive sampling techniques [3]. A recent work [27] based on approximation shows large fluctuations of accuracy over the top-100 nodes on a scale-free graph. A random, shortest path based [26] approximation approach was presented in [28].

For directed and unweighted networks an approach is presented in [8], where, similarly to [11], authors precompute reachable vertices for all the graph nodes. However, at the best of our knowledge, there is a scarcity of contributions focusing on both weighted and directed networks. BC has been proven to be a proxy for traffic volume in [2], but this paper does not address the problem of performing fast computation of BC in large dynamic networks, which is the main objective and contribution of this paper.

3 Fast BC Computation of Weighted and Directed Graphs

In this section, we present the *W2C-FastBC* algorithm, the weighted and directed version of the one previously proposed in [15, 16]. It allows reducing BC computation time of weighted and directed graphs in a parametric way, i.e., by acting on the accuracy of BC values. The algorithm is based on the Brandes' one for weighted graphs and on the heuristic proposed in [30] for identifying graph pivots. As in our previous work, we exploit a fast clustering algorithm based on modularity for identifying graph communities and their related border nodes. Specifically, we used a distributed implementation [29] of Louvain method for weighted and directed graphs [24]. In the following subsections, we briefly introduce the adopted notation, the Brandes' algorithm and discuss modularity for weighted graphs.

3.1 Notation

We assume the following definition throughout the paper. Let $G(V, E, T, W, f(E, T))$ be a dynamic, weighted, directed graph, where V denotes the set of nodes and $E \subseteq V \times V$ the one of edges. $N = |V|$ denotes the number of nodes in the graph. W represents the set of weights and T the set of time units. For instance, for very large networks, T may represent hours of the day. We highlight that the length of the considered time unit (e.g., 1 h) represents the period of observations before a new computation of BC is launched, and translates therefore into a time constraint for computing betweenness centrality. The function $f : E \times T \rightarrow W$ maps each edge $e_{ij} \in E$ at time slot $t \in T$ to a weight $w \in W$. We denote as $\hat{G}(V, E, \hat{W})$ a directed and weighted instance of the dynamic graph G related to a specific time slot \hat{t} and therefore associated to a subset of weights $\hat{W} \subseteq W$. The algorithms reported in the following are related to a specific instance \hat{G} of the dynamic graph G , i.e., BC computation is iteratively performed (in a quasi-real time fashion) at the beginning of time slot $\hat{t} + 1$ on the instance of the dynamic graph related to time slot \hat{t} .

A path $p(v_i, v_j)$, between two nodes v_i and v_j of \hat{G} , consists of a set of nodes and edges that connect these two nodes. The length of a path between any two nodes v_i and v_j , represented by $\text{len}(p(v_i, v_j))$, is the sum of the weights of the edges (or hops) to reach v_j from v_i . If nodes v_i and v_j are directly connected, then the path length is the weight of the link, or 1 for unweighted graphs. A shortest path between any two nodes v_i and v_j , denoted as $sp(v_i, v_j)$, is a path with the minimum length, among all the paths connecting the two nodes. Multiple shortest paths may exist between the same pair of nodes, i.e., multiple paths having the same length. The distance $d(v_i, v_j) = \text{len}(sp(v_i, v_j))$ is the length of the shortest path between nodes v_i and v_j . We denote $\sigma_{v_i v_j}$ as the number of shortest paths between v_i and v_j , while $\sigma_{v_i v_j}(v_k)$ represents the number of shortest paths from v_i to v_j that cross node v_k .

3.2 Brandes' Algorithm

Given a graph \hat{G} , the *pair-dependency* of a *source* node s on an another node v for a *destination* t of the graph is defined as $\delta_{st}(v) = \frac{\sigma_{st}(v)}{\sigma_{st}}$. The betweenness centrality of any node v can be expressed in terms of *dependency score* $\delta_{s\bullet}(v) = \sum_{t \in V} \delta_{st}(v)$, obtained by summing the pair-dependencies of each pair of nodes on v that has s as source node. To compute this score, Brandes' algorithm exploits a recursive relation that is motivated by this observation: let $R = \{w : v \in P_s(w)\}$ be the set of nodes w such that v is a predecessor of w along a shortest path that starts from node s , and $P_s(w) = \{v \in V : \{v, w\} \in E, d(s, w) = d(s, v) + d(v, w)\}$ the set of direct predecessors of a generic node w in the shortest paths from the source node s to w ; then, v is a predecessor also in any other shortest path

starting from s and passing through a different $w \in R$ [9]. Consequently, we have:

$$\delta_{s\bullet}(v) = \sum_{w:v \in P_s(w)} \frac{\sigma_{sv}}{\sigma_{sw}} (1 + \delta_{s\bullet}(w)), \quad (1)$$

Finally, the betweenness centrality BC of node v is obtained as:

$$BC(v) = \sum_{s \in V} \delta_{s\bullet}(v). \quad (2)$$

For scaling purpose, BC values are often normalized by dividing them by $(n - 1) \cdot (n - 2)/2$ for undirected graphs and by $(n - 1) \cdot (n - 2)$ for directed ones.

Conceptually, Brandes' algorithm runs in two phases. During the first phase, it performs a search on the whole graph to find all the shortest paths starting from every node s , considered as source of the breadth-first exploration of the whole graph. In the second phase, it performs dependency accumulation by backtracking along the discovered shortest paths. During these two phases, the algorithm maintains four data structures for each node found on the way: a predecessor list $P_s(v)$, the distance $d_s(v)$ from the source, the number of shortest paths from the source $\sigma_{st}(v)$ and the dependency accumulation when backtracking at the end of the search.

3.3 Weighted Modularity and Louvain Method

Modularity is a metric, defined as a value between -1 and 1 , to measure how tightly the nodes are attached with each other in the network. It was introduced to identify communities in undirected and unweighted (or weighted) networks [25]. Given a graph \hat{G} , partitioned into a set of communities $C = \{c_1, c_2, \dots, c_D\}$, formally, modularity [22] of graph \hat{G} is defined as follows:

$$Q = \frac{1}{m} \sum_{i,j \in V} \left[A_{ij} - \frac{k_i^{in} k_j^{out}}{m} \right] \delta(c_i, c_j) \quad (3)$$

where δ is 1 if $c_i = c_j$ (nodes i and j belong to the same community) or 0 otherwise, m is the sum of all of the edge weights in the graph, k_i^{in} , k_j^{out} are the sum of the weights of the edges entering node i and the edges exiting node j , respectively; A_{ij} is 0 if nodes n_i and n_j are not connected. In case the nodes n_i and n_j are connected then A_{ij} is $w_{i,j}$, that is the weight of the edge connecting nodes i and j .

We exploit modularity for clustering weighted directed graphs with the Louvain method [7, 12]. The algorithm initially searches for *small* communities. Then, it creates a new graph whose nodes are the communities identified in the previous step. These two steps are iteratively run until there is no modularity gain derived by aggregating clusters in larger communities. In our

implementation, the weights used to compute weighted modularity are assumed as in the notion of closeness (nodes are tighter if they have lower distance or travel time), i.e. “smaller is tighter”. This choice is motivated by the fact that we want to reduce the number of border nodes for each cluster. Therefore, we generate communities whose nodes are highly locally inter-connected with short (or fast to travel) local paths. Conversely, when computing shortest paths in SSSPs, weights are assumed as in the notion of length (or travel time), i.e., “higher is farther”. We use a distributed variant of the Louvain algorithm for weighted and directed graphs [24, 29]: all vertices select a new community simultaneously, updating the local view of the graph after each change. Even though some choices will not maximize modularity, after multiple iterations, communities will typically converge thus producing a final result relatively close to the sequential version of the algorithm.

3.4 W2C-FastBC

Given a graph \hat{G} , we split it into a set of clusters (i.e., \mathbf{C}) by using the Louvain (Algorithm 1, line 2) method for weighted graphs [29]¹. The main result of clustering is the identification of *border nodes* (an array for each cluster). A border node is a node having at least one neighbor node in a different cluster (line 4). Then, a parallel execution of Brandes’ algorithm (based on Dijkstra²) is performed inside each cluster to compute the *local BC* (lines 6–11). This computation generates the partial inner-cluster contribution to the BC of each node and additional information, such as the weighted shortest paths and the distances from a node of a cluster towards each border node of the same cluster.

The information above is used to identify the nodes inside each cluster that equally contribute to the dependency score of each node of the graph (equivalence class, see [15, 30] for more details). Taking into account that nodes belonging to the same class produce the same dependency score on each node of the graph, one representative node should be identified as a source node for applying Dijkstra’s algorithm (line 19). This node is called class *pivot*. The partial dependency score calculated for the pivot is then multiplied by the cardinality of the pivot class (line 20). This method avoids re-applying Dijkstra’ algorithm to another node of the same class, thus ensuring fast calculation of BC if $P \ll N$, where P represents the set of pivots selected and N represents the number of nodes of the graph.

¹ The implementation leverages a Scala parallel solution partially based on the Distributed Graph Analytics (DGA) by Sotera: <https://github.com/Sotera/distributed-graph-analytics>.

² The adoption of Dijkstra algorithm instead of breadth first search represents a main variant of our FastBC algorithm proposed in this paper.

Algorithm 1 W2C-Fast-BC: pseudo-code of the main function

```

1: function W2C-FASTBC( $\hat{\mathbf{G}}$ , C, kFrac)
2:   C  $\leftarrow$  weightedLouvainClustering( $\hat{\mathbf{G}}$ )
3:   map i  $\leftarrow$  1, |C| do
4:     bordernodesi  $\leftarrow$  findBorderNodes( $\hat{\mathbf{G}}$ , Ci)
5:   end map
6:   map i  $\leftarrow$  1, |V| do
7:     local $\delta_i$   $\leftarrow$  computeLocal $\delta$ (i, C, bordernodes)
8:   end map
9:   reduce i  $\leftarrow$  (1, |V|), local $\delta_s$ , local $\delta_z$ , i = j do
10:    localBCi  $\leftarrow$  local $\delta_s$ (i) + local $\delta_z$ (j)
11:  end reduce
12:  map i  $\leftarrow$  1, |C| do
13:    superClassesi  $\leftarrow$  WkMeansClustering(Ci, classesi, kFrac)
14:  end map
15:  map i  $\leftarrow$  1, |superClasses| do
16:    Pi  $\leftarrow$  selectPivotOf(superClassesi, localBC)
17:  end map
18:  map i  $\leftarrow$  1, |P| do
19:     $\delta_i$   $\leftarrow$  compute $\delta$ From(Pi)
20:     $\delta_i$   $\leftarrow$  ( $\delta_i$  - localBC)  $\cdot$  |superClassesi|
21:  end map
22:  reduce i  $\leftarrow$  (1, |V|),  $\delta_s$ ,  $\delta_z$ , i = j do
23:    BCi  $\leftarrow$   $\delta_s$ (i) +  $\delta_z$ (j)
24:  end reduce
25:  for i  $\leftarrow$  1, |V| do
26:    BCi  $\leftarrow$  BCi + localBCi
27:  end for
28:  return BC
29: end function

```

The final value of BC is obtained for each node by summing up all partial contributions (produced by the reduce operation, lines 22:24) with local BC values (lines 25:27). To further reduce the computation time, we extended the concept of *class* by introducing *super classes* through an additional clustering operation inside each initial Louvain-derived cluster (line 13). A super class is a group of classes, belonging to the same Louvain cluster and obtained by clustering (via K-means) the vectors generated by considering, for each node, the normalized distances from the Louvain cluster's bordernodes and the amount of shortest paths towards them. To perform class grouping, we exploit a parallel K-means algorithm by using a different K for each initial Louvain cluster. K is defined as a fraction (*K-Fraction*) of the initial number of classes belonging to each Louvain cluster. For example, by considering a fraction equals to 0.4, the algorithm adopts a 0.4 fraction of the number of classes in each Louvain cluster. By this approach, we are able to drive the behavior of the algorithm towards the desired computation time. However, when the computation time decreases the approximation worsens, as deeply illustrated in our previous work [15, 16, 30].

4 Evaluation: Dynamic Analysis of a Real-World Road Network

We implemented our algorithms using *Scala* with the *Apache-Spark* framework. Spark was configured to work in the standalone cluster mode on two Intel Xeon E5 2640 2.4 GHz multi-core machines, each equipped with 56 virtual cores and 128 GB of DDR4 RAM. All algorithms for BC computation leverage 10 cores by spawning the map-reduce tasks on two Spark workers, each equipped with 5 executors.

To evaluate our W2C-Fast-BC algorithm, we leverage a large-scale transportation graph, namely **Rhone-ROADS**, corresponding to the entire road network of the Rhone-Alpes region, France [17]. The graph includes the agglomeration of Lyon and its surroundings and has a geographical extent of approximately 3,300 km². The network is directed and unweighted, with 117,605 nodes and 248,337 edges. We transformed the Rhone-ROADS graph into a dynamic weighted graph by relying on an additional dataset, namely **Rhone-TAXIS**, which reports on anonymized GPS traces of taxis active in the Rhone-Alpes region. Rhone-TAXIS has been collected by the French operator Radio Taxi via a fleet of approximately 400 taxis during 2011-2012. Geo-referenced taxi trips are collected according to a variable sampling interval (between 10 and 60 s), with a global average of 800,000 measurements per day. The generic sample of the Rhone-TAXIS dataset, i.e., an *elementary taxi trip*, includes the time-stamped start and arrival GPS positions of a small segment traveled by the associated taxi identifier. These measures permit to roughly estimate the traveled distance and the instant speed of the taxi moving along a given road segment. In order to improve the quality of the Rhone-TAXIS dataset and properly compute the edge weights, we have filtered out elementary trips with unrealistic speeds (i.e., higher than 130 km/h).

We map-matched all the elementary trips of the Rhone-TAXIS dataset and computed hourly median speeds for the edges of the Rhone-ROADS graph, thus generating a (discrete) dynamic weighted graph. More specifically, we considered 24 hourly time slots for a typical day, and computed the weighted graph instance corresponding to each time slot t . To that purpose, we retain only the edges with non-null value of the median speed during t , as estimated from the map-matched taxi trips related to the same edge and time slot t . Thus, we calculate the weight at time slot t of each edge (which corresponds to a road link with known length) as the estimated travel time to cross the corresponding road segment, i.e., the ratio of the length of the link to the median speed estimated on that link. By iterating this process for the different time slots, we obtain the final dynamic weighted graph (i.e., our graph G).

The W2C-Fast-BC and Brandes algorithms are applied iteratively to each hourly instance of the dynamic graph (i.e., a snapshot \hat{G}). This is conceptually equivalent to an on-line operational situation, where the graph naturally emerges from sensor-collected data used to continuously compute up-to-date traffic information on each edge with hourly periodicity. It is worth noting that, given the relatively small size of the observed taxi fleet and the circadian rhythm

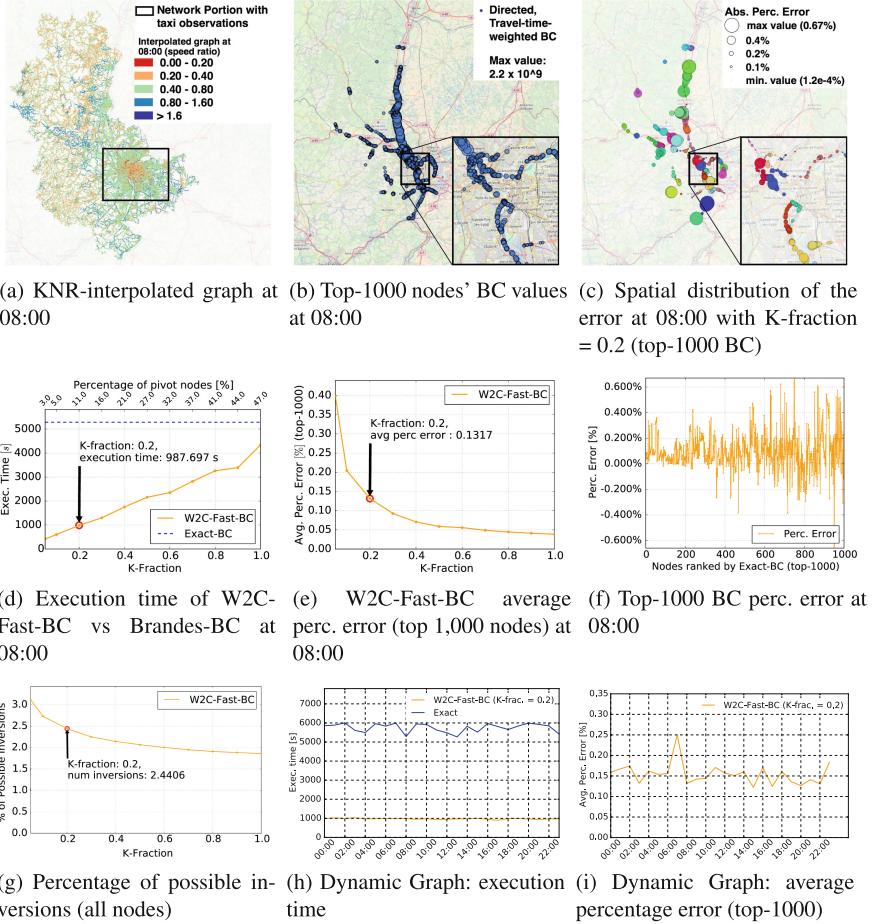


Fig. 1. The interpolated dynamic taxi graph: median-speed-to-max-speed ratios, top-1000 BC nodes and performance evaluation of W2C-Fast-BC at different hours of the day.

characterizing human mobility, snapshots of the dynamic graph related to rush hours (e.g., 7:00-09:00 and 17:00-19:00) have a much smaller size with respect to the original Rhone-ROADS graph, i.e., approximately 30,000 nodes and 60,000 edges (see the framed portion of the graph in Fig. 1a). Such size further reduces for graph snapshots related to non-rush hours. Indeed, most of the observed elementary trips are condensed within the city center of Lyon, with only few observations recorded in the outskirts and within rural areas as well as during night time. However, since the goal of the paper is to prove the efficiency of our solutions with respect to very-large scale weighted networks, we have decided to increase the scale of the dynamic graph by means of a spatial interpolation technique.

To obtain a dynamic, realistic, weighted network, larger than the one directly observed from taxi trips, we leveraged an interpolation technique that we call *KNR-interpolation*³. The technique allows estimating the hourly value of median speed (and thus the median travel-time weight) for those edges of the original Rhone-ROADS network with no available observation from taxi trips at time slot t . Figure 1a graphically shows the KNR-interpolated snapshot at 08:00 of a typical working day. Figure 1a also presents speed-ratios (i.e., median speed divided by road speed limit) either estimated via taxi traces (for the framed portion of the graph) or via the KNR interpolation technique. Red and orange colors indicate highly-congested situations on the edge, i.e., lower values of the speed ratio, while greens and blues indicate a smooth, non-congested situation at time t . The resulting graph has approximately the same size of the Rhone-ROADS network.

The values of BC for the top-1000 nodes are reported in Fig. 1b (nodes with larger circles have higher BC) for the snapshot related to 08:00. As it can be observed from Fig. 1d, the exact algorithm for computing BC on the weighted graph requires a computation time of more than one hour, therefore unable to complete within the duration of the time slot. Remarkably, our W2C-Fast-BC computes in only 987 s (i.e., approximately 15 min) when using a K-fraction equal to 0.2, showing an average percentage error of 0.13% (Fig. 1e) and a maximum percentage error of $\pm 0.7\%$ (Fig. 1f) over the top-1000 BC nodes. The number of clusters obtained via the Louvain method is 127, while the number of classes, as retrieved by the K-means algorithm, corresponds to 12494 (which also represents the number of pivot) out of 117605 nodes. The spatial distribution of the absolute percentage error is reported in Fig. 1c, with a node size proportional to the error, and a different color to represent the cluster each node belongs to. The figure highlights a scattered distribution of the percentage error over the graph. Finally, we also report in Fig. 1g the percentage of inversions against the maximum number of possible inversions [16], which clearly highlights the capability of our solution to preserve a good ranking (i.e., low percentage of inversions) of all the network nodes in terms of their BC values, even when using low values of the K-fraction parameter.

Similar results have been observed over the whole dynamic graph (i.e., the 24 hourly time slots, as reported in Fig. 1h and i), thus proving the adequacy of our solution for quasi real-time monitoring of dynamic, directed, weighted road-networks.

³ KNR-interpolation is based on K-nearest-neighbor regression [1], a non-parametric supervised machine-learning technique. Each edge is modeled as a data point with multiple topological features. The median speed at time slot t , available for some edges (labeled instances) and missing for other ones (unlabeled instances), represents the target interpolated feature.

5 Conclusion

We presented an approximated betweenness centrality computation method for weighted and directed graphs. By exploiting representative pivot nodes, identified through the definition of a class of equivalence, the proposed algorithm is able to find the most critical nodes in a directed and weighted graph with a significant speedup and a negligible error if compared to the exact Brandes' algorithm. The algorithm has been evaluated on a real transportation network dataset, where dynamic weights were derived from the analysis of GPS-taxi data. The results reported in the paper show that directed and weighted graphs provide further information useful for a more realistic interpretation of high BC values. Moreover, the dynamic analysis exhibits continuously changing values of BC that are useful for predicting possible critical spots. Therefore, the algorithm represents an important milestone towards the objective of defining a complete framework for monitoring road networks and predicting traffic flows.

In the future, we aim to leverage additional solutions to further reduce computation time and error by exploiting, for instance, the hierarchical information produced by the Louvain community detection algorithm. Moreover, in relation to the transportation case study, we will consider additional information (e.g., dynamic traffic volumes) in order to exploit dynamically computed BC values as effective predictors of network congestion. Finally, we aim to generalize the study of the performance of the algorithm, by means of a thorough evaluation with respect to other approximated solutions, by using different kinds of network (e.g., small-world, random networks) related to different application domains (e.g., social networks, brain networks, etc.).

Acknowledgment. This work has been supported by the French research project PROMENADE (grant number ANR-18-CE22-0008), the H2020 framework project SoBigData, grant number 654024, and the GAUSS project (MIUR, PRIN 2015, Contract 2015KWREMX).

References

1. Altman, N.S.: An introduction to Kernel and nearest-neighbor nonparametric regression. *Am. Stat.* **46**(3), 175–185 (1992)
2. Altshuler, Y., Puzis, R., Elovici, Y., Bekhor, S., Pentland, A.S.: Augmented betweenness centrality for mobility prediction in transportation networks. In: International Workshop on Finding Patterns of Human Behaviors in Network and Mobility Data (NEMO) (2011)
3. Bader, D.A., Kintali, S., Madduri, K., Mihail, M.: Approximating betweenness centrality. In: Proceedings of the 5th International Conference on Algorithms and Models for the Web-Graph, WAW 2007, pp. 124–137. Springer, Berlin (2007)
4. Bader, D.A., Madduri, K.: Parallel algorithms for evaluating centrality indices in real-world networks. In: International Conference on Parallel Processing, 2006. ICPP 2006, pp. 539–550. IEEE (2006)
5. Bergamini, E., Meyerhenke, H.: Approximating betweenness centrality in fully dynamic networks. *Internet Math.* **12**(5), 281–314 (2016)

6. Bergamini, E., Meyerhenke, H., Staudt, C.L.: Approximating betweenness centrality in large evolving networks. In: 17th Workshop on Algorithm Engineering and Experiments, ALENEX 2015, pp. 133–146. SIAM, Philadelphia (2015)
7. Blondel, V.D., Guillaume, J.L., Lambiotte, R., Lefebvre, E.: Fast unfolding of communities in large networks. *J. Stat. Mech. Theory Exp.* **2008**, P10008 (2008)
8. Borassi, M., Natale, E.: KADABRA is an adaptive algorithm for betweenness via random approximation. In: ESA, LIPIcs, vol. 57, pp. 20:1–20:18. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik (2016)
9. Brandes, U.: A faster algorithm for betweenness centrality. *J. Math. Sociol.* **25**(2), 163–177 (2001)
10. Brandes, U., Pich, C.: Centrality estimation in large networks. *Int. J. Bifurc. Chaos* **17**(07), 2303–2318 (2007)
11. Chehreghani, M.H., Bifet, A., Abdessalem, T.: Efficient exact and approximate algorithms for computing betweenness centrality in directed graphs. In: Pacific-Asia Conference on Knowledge Discovery and Data Mining. Springer, Cham, pp. 752–764 (2018)
12. Dugué, N., Perez, A.: Directed Louvain: maximizing modularity in directed networks. Ph.D. thesis, Université d’Orléans (2015)
13. Freeman, L.C.: Centrality in social networks conceptual clarification. *Soc. Netw.* **1**(3), 215–239 (1978)
14. Freeman, L.C.: A set of measures of centrality based on betweenness. *Sociometry* **25**, 35–41 (1997)
15. Furno, A., El Faouzi, N.E., Sharma, R., Zimeo, E.: Reducing pivots of approximated betweenness computation by hierarchically clustering complex networks. In: International Conference on Complex Networks and Their Applications, pp. 65–77. Springer (2017)
16. Furno, A., El Faouzi, N.E., Sharma, R., Zimeo, E.: Two-level clustering fast betweenness centrality computation for requirement-driven approximation. In: 2017 IEEE International Conference on Big Data (Big Data), pp. 1289–1294. IEEE (2017)
17. Furno, A., El Faouzi, N.E., Sharma, R., Cammarota, V., Zimeo, E.: A graph-based framework for real-time vulnerability assessment of road networks. In: 2018 IEEE International Conference on Smart Computing (SMARTCOMP), pp. 234–241. IEEE (2018)
18. Geisberger, R., Sanders, P., Schultes, D.: Better approximation of betweenness centrality. In: Proceedings of the Meeting on Algorithm Engineering and Experiments, pp. 90–100. Society for Industrial and Applied Mathematics (2008)
19. Hayashi, T., Akiba, T., Yoshida, Y.: Fully dynamic betweenness centrality maintenance on massive networks. *Proc. VLDB Endow.* **9**(2), 48–59 (2015)
20. Kourtellis, N., Morales, G.D.F., Bonchi, F.: Scalable online betweenness centrality in evolving graphs. In: 2016 IEEE 32nd International Conference on Data Engineering (ICDE), pp. 1580–1581 (2016)
21. Lee, M.J., Lee, J., Park, J.Y., Choi, R.H., Chung, C.W.: Qube: a quick algorithm for updating betweenness centrality. In: Proceedings of the 21st International Conference on World Wide Web, WWW 2012, pp. 351–360. ACM, New York (2012)
22. Leicht, E.A., Newman, M.E.J.: Community structure in directed networks. *Phys. Rev. Lett.* **100**(11), 118703 (2008)
23. Madduri, K., Ediger, D., Jiang, K., Bader, D.A., Chavarria-Miranda, D.: A faster parallel algorithm and efficient multithreaded implementations for evaluating betweenness centrality on massive datasets. In: 2009 IEEE International Symposium on Parallel and Distributed Processing, pp. 1–8. IEEE (2009)

24. Newman, M.E.: Analysis of weighted networks. *Phys. Rev. E* **70**(5), 056131 (2004)
25. Newman, M.E.J., Girvan, M.: Finding and evaluating community structure in networks. *Phys. Rev. E* **69**, 026113 (2004)
26. Newman, M.J.: A measure of betweenness centrality based on random walks. *Soc. Netw.* **27**(1), 39–54 (2005)
27. Ohara, K., Saito, K., Kimura, M., Motoda, H.: Accelerating computation of distance based centrality measures for spatial networks. In: International Conference on Discovery Science. Springer, Cham (2016)
28. Riondato, M., Kornaropoulos, E.M.: Fast approximation of betweenness centrality through sampling. *Data Min. Knowl. Discov.* **30**(2), 438–475 (2016)
29. Sotera: dga-graphx: Graphx algorithms. <https://github.com/Sotera/spark-distributed-louvain-modularity>. Accessed Oct 2018
30. Suppa, P., Zimeo, E.: A clustered approach for fast computation of betweenness centrality in social networks. In: 2015 IEEE International Congress on Big Data, pp. 47–54 (2015)
31. White, D.R., Borgatti, S.P.: Betweenness centrality measures for directed graphs. *Soc. Netw.* **16**(4), 335–346 (1994)
32. White, S., Smyth, P.: Algorithms for estimating relative importance in networks. In: Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 266–275. ACM (2003)



Node Ordering for Rescalable Network Summarization (or, the Apparent Magic of Word Frequency and Age of Acquisition in the Lexicon)

Violet Brown¹, Xi Chen², Maryam Hedayati^{1,2}, Camden Sikes², Julia Strand¹,
Tegan Wilson², and David Liben-Nowell^{2(✉)}

¹ Department of Psychology, Carleton College, Northfield, MN 55057, USA

² Department of Computer Science, Carleton College, Northfield, MN 55057, USA
dln@carleton.edu

Abstract. How can we “scale down” an n -node network G to a smaller network G' , with $k \ll n$ nodes, so that G' (approximately) maintains the important structural properties of G ? There is a voluminous literature on many versions of this problem if k is given in advance, but one’s tolerance for approximation (and the resulting value of k) will vary. Here, then, we formulate a “rescalable” version of this approximation task for complex networks. Specifically, we propose a *node ordering* version of graph summarization: permute the nodes of G so that the subgraph induced by the first k nodes is a good size- k approximation of G , averaged over the full range of possible sizes k . We consider as a case study the phonological network of English words, and discover two natural word orders (word frequency and age of acquisition) that do a surprisingly good job of rescalably summarizing the lexicon.

Keywords: Network summarization · Node ordering
Phonological networks

1 Introduction

At SIGGRAPH 2007, Shai Avidan and Ariel Shamir presented a remarkable technique for “content-aware image resizing” [5]: shrink the size of an image while preserving, to the greatest extent possible, its important visual qualities. This problem can be solved crudely by simple cropping or rescaling, but Avidan and Shamir’s approach is more subtle: they identify an ordering of the “seams” (contiguous edge-to-edge paths through the image) from least important to most

This work grew out of portions of a research project that was carried out by the authors of this work in collaboration with Aman Panda and Duo Tao. We gratefully acknowledge their contributions. This work was supported in part by Carleton College. Comments are welcome.

important. Their algorithm allows a user to shrink an image from n pixels on a side to any size $k \leq n$, with k chosen by the user in real time, by eliminating the $n - k$ least important seams.

What would it mean to perform an analogous “resizing” for a complex network? Is there a meaningful way to shrink an n -node network to any size $k \leq n$, with k chosen on the fly by the user—say, ordering the nodes by “representativeness”—while preserving important graph-theoretic qualities, so that the graphs in the resulting nested sequence are “as much like” the original as possible?

Approximating complex networks. When members of the complex-networks community describe a network as “complex,” we seem to have in mind a fuzzy constellation of properties, expecting the network to exhibit many of these desiderata: e.g., “small-world” properties [49], a heavy-tailed degree distribution [12], community structure [17], and degree assortativity [38]. We probably also expect the network to be “large.” (Despite the now-ostentatious attention to Zachary’s Karate Club [51], few researchers would argue for it as a paradigmatic complex network; how much complexity can 34 nodes admit?) For a variety of reasons, though, the large size of a network can be problematic. This issue is immediate in the sense of computational complexity—one cannot afford $\Omega(n^2)$ time on a billion-node social network—and it is even more of an issue if one seeks some kind of real-world intervention.

With these sorts of motivations in mind, many researchers have performed significant work on the task of taking a large complex network and performing a type of lossy compression on it; that is, identifying some smaller graph (either by deleting or aggregating nodes) that is a useful approximation to the original. But this problem is difficult for a number of reasons: algorithms for the network approximation problem itself often have running times that grow unfavorably in the size of the full network; the resulting smaller network may vary widely depending on the size of the desired subnetwork; and it is unclear as to the right way to assess the quality of the smaller subnetwork. (See [1, 33, 34] and Sect. 3.)

The present work: node ordering as (rescalable) network summarization. The goal of these graph summarization algorithms is to preserve “interesting” properties of the graph, while reducing the size of the graph as much as possible. But a major challenge here—highlighted clearly by Liu et al. [34]—is that what counts as “interesting” will differ from one researcher to another (and, for that matter, so will what counts as “preserved”). And size-reduction algorithms may well require us to precommit to the size of the desired smaller network *and* to the network properties of interest, both of which may be undesirable. (Though see [35].)

Here, we propose a task that embraces these differences in the desired level of approximation: given a complex network G , we seek to identify an *order* v_1, v_2, \dots, v_n of the nodes of G such that the “prefix graph” for a given size k —that is, the subgraph induced by the node set $\{v_1, v_2, \dots, v_k\}$ —is as close

an approximation to G as possible, for any desired size k . We quantify success for this *Node Ordering Problem* in both the sense of global statistics about the graph, and local statistics about the importance of individual nodes in the full graph and the subgraphs.

We will focus on a particular complex network as a case study for our discussion: the *phonological network*, in which nodes correspond to words in a natural language (here, English), and edges connect pairs of nodes whose pronunciations differ by a single edit [4, 43, 47, e.g.]. We will describe some natural node orderings in this network, including two derived from external data sources—word frequency and age of acquisition—that do a remarkably good job of “unkinking” the phonological network, producing a nested sequence of graphs that reproduce to a surprising extent the statistical properties of the lexicon as a whole.

2 The Node Ordering Problem: Approximating Degree

Our framework of successively approximating a graph via one-by-one additions of nodes is quite broad; we could apply it with a variety of graph-theoretic quantities, and also with a variety of ways to quantify the difference between two graphs with respect to any particular quantity. But, to start, we will formalize one specific version of the Node Ordering Problem, using what is perhaps the simplest nontrivial way to compare graphs: the *degree* of the graphs’ nodes. Let $\delta(u, G)$ be a function reporting the degree of the node u in any graph G .

Prefix graphs. First, we fix a bit of terminology. We are given an undirected graph $G = \langle V, E \rangle$, called the *full network*. Denote by $n = |V|$ the number of nodes in G . We will refer to a permutation of the vertices $\pi = \pi_1, \pi_2, \dots, \pi_n$ as a *node ordering*.

Any particular permutation π defines a sequence of n *prefix networks*, one of each size between 1 and n ; specifically, the k -node prefix network of G under π is the subgraph of G induced by the nodes $\{\pi_1, \pi_2, \dots, \pi_k\}$. (The *subgraph of $G = \langle V, E \rangle$ induced by a set $A \subseteq V$* is the graph G_A with nodes A and containing all edges in E that join two nodes in A ; that is, $G_A = \langle A, \{(u, v) \in E : u \in A \text{ and } v \in A\} \rangle$.)

See Fig. 1 for a small example. (Note that, as always, the last prefix network is the full network—i.e., in Fig. 1, we have $G = G_{\{1,2,3,4\}}$.)

Measuring structural quality of a subnetwork. Any node ordering defines a nested sequence of prefix graphs, starting with a single isolated node and ending with G itself. We must now describe the objective function—i.e., how do we assess the quality of a particular permutation? Our evaluation is guided by three principles. First, we seek *low discrepancy* between the sequence of prefix graphs and the full network (averaged over all n different prefix sizes). Second, we measure discrepancy using *relative error*: if the prefix graph exhibits a value x and the full network a value x^* , then we compute the error as $\frac{|x - x^*|}{x^*}$.

Third, we want to capture both *global error* (does the prefix graph have similar average statistics to the full network?) and *local error* (do those nodes that appear in the prefix graph have similar statistics there as they do in the full network?). In keeping with these principles, we define two notions of error:

Definition 1 (Global Error). For a prefix graph G_A , the global (relative) degree error is the relative error of the mean degree of G_A compared to that of $G = \langle V, E \rangle$:

$$\text{global error of } G_A = \frac{\left| \frac{1}{|A|} \cdot [\sum_{v \in A} \delta(v, G_A)] - \frac{1}{|V|} \cdot [\sum_{v \in V} \delta(v, G)] \right|}{\frac{1}{|V|} \cdot [\sum_{v \in V} \delta(v, G)]}.$$

Definition 2 (Local Error). For a prefix graph G_A , the local (relative) degree error is the mean relative error of each node in G_A compared to $G = \langle V, E \rangle$, averaged across all nodes present in G_A :

$$\text{local error of } G_A = \frac{1}{|A|} \cdot \sum_{v \in A} \frac{|\delta(v, G_A) - \delta(v, G)|}{\delta(v, G)}. \quad (1)$$

The errors for the prefix graphs for our small sample graph are also given in Fig. 1.

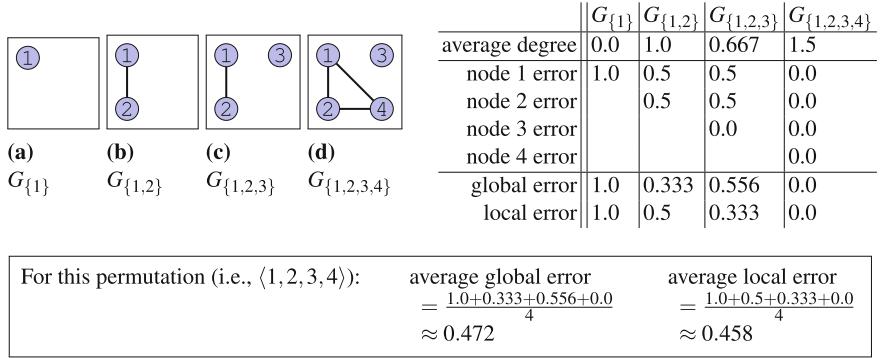


Fig. 1. Consider the full network G with nodes $\{1, 2, 3, 4\}$ and edges $\{(1, 2), (2, 4), (1, 4)\}$, under the permutation $\langle 1, 2, 3, 4 \rangle$. First, we show the four prefix graphs for this network under this permutation. Then, we show the stage-by-stage local and global errors. (For example, node 2's local error in $G_{\{1,2\}}$ is 0.5 because it has degree 1 in $G_{\{1,2\}}$ and degree 2 in G , so its local error is $|2 - 1| / 2 = 0.5$.) Note that average global and local error are permutation dependent, so these values would be different for a different ordering of the nodes.

Note that global and local error measure different things: low global error corresponds to the *density* of G_A matching that of G , while low local error corresponds to specific node degrees in G_A matching their degrees in G . It is possible for a graph G_A to have low global error while simultaneously having high local error (i.e., this graph maintains the average degree of G , but the average arises from different local connections among nodes in the two graphs), or vice versa. See Fig. 2.

Here, then, is the formal statement of our problem:

Definition 3 (Node Ordering Problem [Degree Version]). *Given an undirected graph $G = \langle V, E \rangle$, output the permutation of V that minimizes the average total error (global + local), where the average is taken across all $|V|$ prefix graphs.*

Although we have focused on degree as the node-level measure of interest, all of our definitions apply for an arbitrary node-level function. Many other measures are at least as interesting to consider as degree—but even this “simple” measure will reveal some surprisingly complex and subtle network features.

3 Related Work

Here we will (nonexhaustively) highlight some of the work in the many areas of related research. First, though, we note that our Node Ordering Problem is fundamentally different from ranking the nodes by some kind of centrality measure, in which the most important nodes appear earliest; rather, we are trying to produce an order of the nodes that is “always” roughly as important as the list as a whole; how good a node is to include next depends on which nodes have already been selected.

Summarizing and sampling in networks. The most closely related body of research—and also the most voluminous—studies algorithms to shrink large complex networks. Closest is work on *simplifying* graphs through the removal of nodes (or, less similarly, edges) [19, 40, 46]. There is also a great deal of work on *sampling* graphs, in which one tries to choose a good set of representative nodes from a large network on the fly, generally without knowing the full graph [20, 30, 35, 36, 39]. The excellent surveys of Liu et al. [34], Lin et al. [33], and

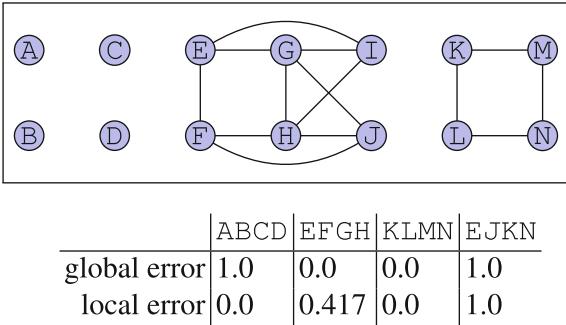


Fig. 2. A graph with 14 nodes and 14 edges, and four different sets of four nodes whose induced graphs exhibit all four combinations of high/low local error and high/low global error.

Ahmed et al. [1] describe much more of this line of work—beyond the highly incomplete list cited here—including research on other kinds of graph compression (e.g., the aggregation of many nodes into supernodes) that are further afield from our task.

Modeling the evolution of graphs. In Sect. 4, we examine how a particular graph G evolves—namely, how the phonological network changes as we add words, one by one, in the order that an average person learns them. When nodes are ordered by arrival time, the prefix graphs form a flipbook of G 's temporal evolution. Questions about how particular complex networks evolve over time are well studied, ranging from the local (which new links will form, and when?) [8, 29, 32] to the global (how will density and diameter change over time?) [31]. That work generally considers *both nodes and edges arriving over time*; here, we “know the future” of the network—the edge (u, v) forms at precisely the moment that the second of the two nodes arrives in the graph—so the kind of graph evolution that we see is generally quite different from the changes studied in this literature.

Graph drawing, minimum linear arrangement, and comparing permutations with costs varying depending on position. Multiple computational communities (from graph drawing to VLSI design) have considered the task of ordering the nodes of a graph so that edges connect nodes at nearby positions in the ordering. In the graph-drawing context, the resulting images are called *arc diagrams* [37, 48]; ordering nodes to minimize the total length of edges in an arc diagram is called “minimum linear arrangement (MinLA)” [14, 16], which is NP-hard. (This problem is also similar to that of approximating a general metric as a line [15].)

MinLA is a close match for our notion of local error. In MinLA, one seeks to minimize $\sum_{(u,v) \in E} |\pi_u - \pi_v|$; here, the number of prefix graphs in which exactly one of $\{u, v\}$ appears is precisely $|\pi_u - \pi_v|$. But there are important differences: we consider relative, not absolute, error, and we average error across all prefix graphs rather than summing error over edges; a single node’s local error counts less when there are more nodes in the ordering (because that cost is divided by a larger population size). In most ordering problems, as in MinLA, the cost measure does not depend on the location of any errors, though a few researchers have recently studied scenarios that, like ours, penalize errors differently depending on where the error sits [21, 25].

4 Case Study: Word Recognition and the Phonological Network

In this section, as a case study, we consider a particular medium-sized complex network. Our network comes from the psycholinguistics literature on *spoken-word recognition* [4, 47]: nodes represent the words in the language, and we join two words w and w' by an edge if their pronunciations differ only by a single phonemic insertion, deletion, or substitution. For example, neighbors of *cowl* /kaʊl/

include *scowl* /skəʊl/ (an insertion), *owl* /əʊl/ (a deletion), and *fowl* /fəʊl/ (a substitution). This network has many of the properties that we discussed previously: a giant component, small average path lengths, high clustering coefficients, degree assortativity, etc. [4, 43, 47]. We obtained our list of words and pronunciations from the English Lexicon Project [6]. We discarded words for which we had no *word frequency* or *age of acquisition* data (see below), and removed homophones, keeping only the highest frequency word with each pronunciation. The resulting graph G_{lex} contains $n = 30,515$ words, with an average degree ≈ 3.5 .

Ordering nodes randomly. As a baseline, we begin with a *random* ordering of the nodes of G_{lex} . Figure 3 shows both global and local error rates for 16 random orderings of the words in G_{lex} , as the fraction of nodes included in the graph ranges from 0 to 100%. To calculate a single measure of the quality of each order π , we compute the average error rates across all n prefix graph sizes of π , resulting in a pair of numbers per ordering. (We calculate this average approximately, averaging the error for prefix graphs of size 100, 200, \dots , n , and round to the hundredths place.)

When the nodes are ordered randomly, we see a linear trend for both global and local error, as we could expect. Let $G[\alpha]$ denote the prefix graph resulting from including a random α -fraction of the nodes of G_{lex} . The global error of $G[\alpha]$ is the fraction by which $G[\alpha]$'s average degree is lower than G_{lex} 's average degree; in expectation, this fraction drops linearly with α . (A particular edge from G_{lex} is included in $G[\alpha]$ with probability $\approx \alpha^2$ —both endpoints must appear in $G[\alpha]$ —so $G[\alpha]$ will contain $n \cdot \alpha$ nodes and, on average, $|E| \cdot \alpha^2$ edges, and thus an average degree of $(2|E|\alpha^2)/(n\alpha) = (2|E|/n) \cdot \alpha$. The average degree of G_{lex} is $2|E|/n$.) Average local error is similarly linear, though starting at about 0.7 instead of

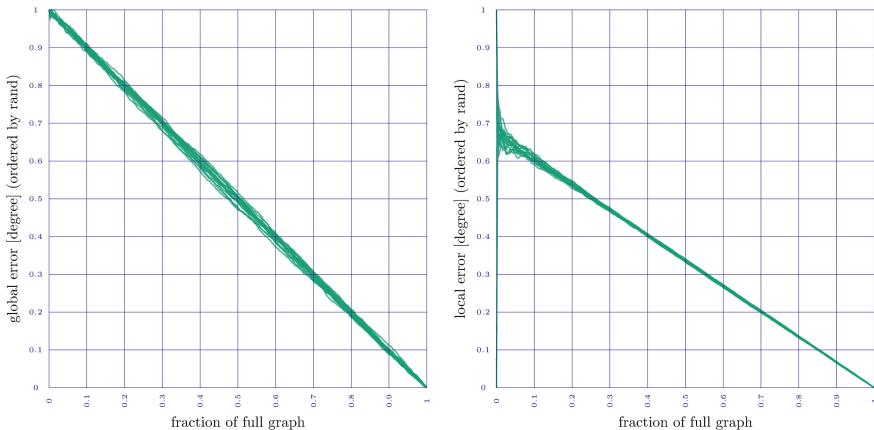


Fig. 3. Global error (left) and local error (right) rates for 16 random orderings of the words in the lexicon. In these orders, the average global error was always in $\{0.49, 0.50, 0.51\}$ (median = 0.50); local error was always 0.33 or 0.34 (median = 0.34).

1.0: about 30% of the nodes in G_{lex} are isolated (i.e., have no neighbors); these nodes have zero local error at the very moment that they are added into the graph.

Ordering nodes by network properties. To minimize local error for a node u as quickly as possible, we want as many edges incident to u to appear in the graph as soon after u in the order as possible. Adding nodes greedily by degree, highest degree first, tends to achieve this goal, because G_{lex} is assortative with respect to degree. Thus ordering nodes by degree, breaking ties randomly, seems promising. While this greedy-by-degree order is good for local error, though, it does poorly with global error: the average degree rapidly shoots far above G_{lex} 's 3.5, and stays well above that target for almost all prefix sizes. Greedy-by-degree ordering yields local error 0.10 and global error 1.19 (median across 16 different random tiebreakers), substantially worse than the random ordering. (Note that relative errors exceed one when the prefix graph's average degree is more than twice that of G_{lex} .)

We find a similar effect when we order the nodes by *closeness* or *betweenness centrality*: the most central nodes' degrees are too high, and the global average goes, and stays, too high (closeness global error 1.18, local 0.10; betweenness global error 0.85, local 0.17). We could also greedily add nodes by degree, *lowest* first; this strategy has an analogous problem, but with a persistently too-low global average.

Ordering nodes by external properties. Indeed, it is hard to formulate a network-theoretic property that would intuitively yield good performance. Somehow we need a sequence of nodes in which we tend to add “regions” of the graph at a similar time (so that newly added nodes' local error drops quickly), while also ensuring that those regions have nodes that are typical of the whole graph (so that the global error stays low). Of course, one could explicitly select for these desired properties—e.g., repeatedly greedily removing the node whose removal increases total error by the least—but here we consider another option: ordering the nodes by psycholinguistic properties that are, at least nominally, independent of graph position:

- *Frequency.* We obtained word frequency counts for all the words in our lexicon from the SUBTLEX_{US} corpus of 51 million words of American subtitles [9], stored as frequency per million words.
- *Age of acquisition.* We used ratings of the age at which a given word was learned, its *age of acquisition* (*AoA*), from Kuperman et al. [26]. These data were obtained by adults retrospectively self-reporting the age at which they learned a given word; the data are expanded so that w 's AoA is recorded based on the “lemma” of w —e.g., *endorsed* is recorded as being acquired at the same age as *endorse*. Despite the inherent limitations of such self-reporting, Kuperman et al. [26] argue that these estimates accurately reflect the order in which words were learned, and the data have proven predictive in other psycholinguistic settings [10, 13].

Although higher frequency words tend to be acquired earlier, these two quantities capture different phenomena, particularly for the lemma-expanded version of AoA. Many pairs of words (29.5%) are inverted in the AoA vs. frequency orders (e.g., *water* is early in both lists, *watered* is early AoA [because its lemma is *water*] but low frequency, and *business* is high frequency but acquired fairly late).

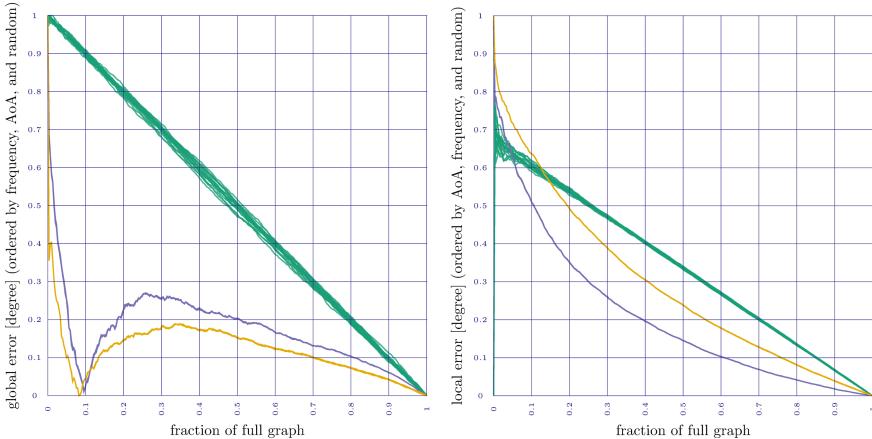


Fig. 4. Global (left) and local (right) error rates for degree for three word orders: random (green), frequency [high to low] (gold), and AoA [low to high] (blue). Frequency has the lower global error and AoA the lower local error; both are better than random in both measures.

The analogue to Fig. 3 for these two orderings is shown in Fig. 4. (We resolve any ties in the ordering by randomizing, executing 16 distinct runs for each measure.) We see notable improvement in the global error over the random ordering: the average global error for frequency is 0.12, and for AoA is 0.17 (vs. 0.50 for random). For local error, the difference is less pronounced, but error is still smaller than that for the random ordering: frequency's average local error is 0.29 and AoA's is 0.21 (vs. 0.34 for random). Note that the points at which AoA and frequency's global error first hits zero, at $\approx 10\%$ of the full graph, are the points at which the prefix graphs' average degree first exceeds G_{lex} 's average: to the left of that point, the prefix graphs are too sparse; to the right, the prefix graphs are too dense.

We also tried ordering the nodes by two other properties that are nominally unrelated to network position: in increasing order of *orthographic length* (how many letters are in the spelling of the word?) and *phonological length* (how many phonemes are in the pronunciation of the word?). These orderings suffer from the same problem as ordering by centrality: the fraction of possible k -phoneme strings that are actually words decreases with k , so short words have many more

neighbors than average; the global error for word-length orderings is quite high (> 0.8) as a result.

Going beyond degree: clustering coefficient. Although we introduced it strictly in the context of degree, we can consider versions of the Node Ordering Problem for any node-level property. Here, we consider *clustering coefficient*: the fraction of pairs of a node's neighbors that are directly joined by an edge. (Clustering coefficient of G_{lex} has been studied in several psycholinguistic contexts [2, 11, 50].) See Fig. 5: frequency and AoA both vastly outperform random ordering for global error, and are roughly comparable in local error. (Figure 5 shows that they achieve this local error in different ways, though: the random ordering benefits from the fact that about 65% of words have degree ≤ 1 , and ergo clustering coefficient = 1. Frequency and AoA tend to do poorly on their early prefix graphs, before overtaking the random ordering about a third of the way through the graph.)

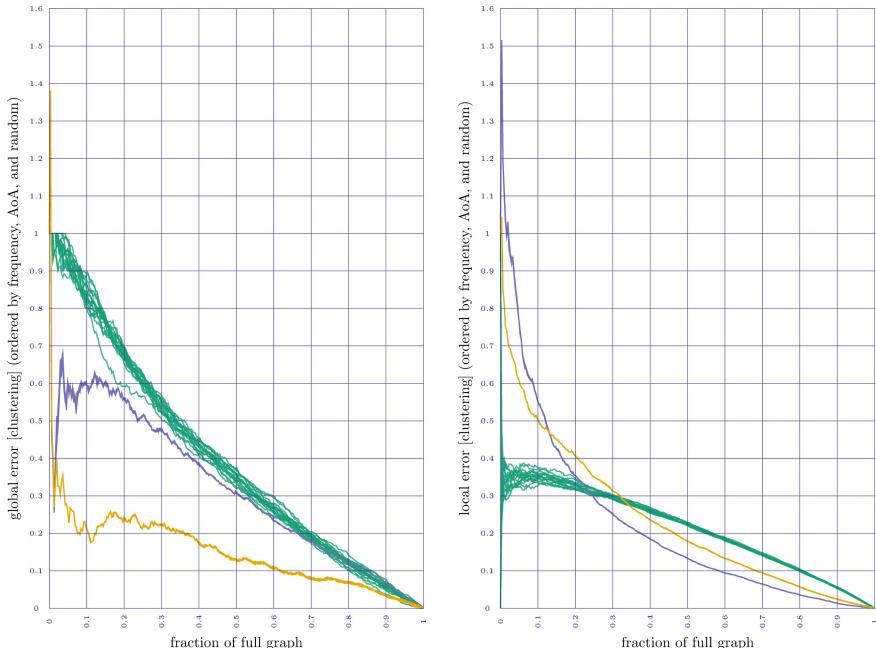


Fig. 5. Global (left) and local (right) error rates for clustering coefficient for words ordered randomly (green), by decreasing frequency (gold), and increasing AoA (blue). All other orders mentioned previously (degree, betweenness, closeness, and word length) have worse average total error (global+local) than random ($0.40 + 0.21$); both frequency ($0.15 + 0.23$) and AoA ($0.32 + 0.22$) are again better than random.

5 Discussion and Future Directions

The task that we introduced in this paper is a broad one: order the nodes of a given complex network in a permutation π such that the prefix graph induced by $\{\pi_1, \pi_2, \dots, \pi_k\}$ is a good (global and local) approximation to the full network, averaging over the possible values of k . There are, of course, a slew of ways to measure the similarity of the prefix graph and the full network, many more than the degree and clustering coefficient measures that we examined here. Understanding the extent to which a good node order for these two measures is also a good node order for other key graph-theoretic properties is an obvious next step. Many of the summarization and sampling algorithms for complex networks (see Sect. 3) could potentially be adapted to this setting, too.

For any particular fixed node-level property, there is a natural greedy algorithm that applies: starting from the full graph, repeatedly put at the end of the node order that vertex whose removal increases total error by the least. Hill-climbing algorithms could also be adapted fairly straightforwardly to this setting. These approaches differ from the measure-agnostic view of the ordering task that we have taken so far (we seek a node order that “in all important ways” reproduces the full graph), but one may approach the problem from a measure-specific perspective (see [35]). This style of algorithm may be computationally prohibitive, though; incremental algorithms for the measure in question (e.g., [28, 41]) are necessary, but not sufficient, to make the computation feasible.

Node ordering in the phonological network. The most salient fact from our examination of the phonological network is that ordering words by frequency or age of acquisition results in remarkably low error, both global and local, and that these orderings outperform the random baseline in both degree and clustering coefficient. Shorter words tend to appear early in these lists, and shorter words tend to have higher degree—but both frequency and AoA outperform degree-based and word-length-based orders for the nodes.

Why might AoA and frequency do such a good job in ordering the nodes of the phonological network for degree and clustering coefficient? In part, it seems, their success stems from a sense in which these word properties interpolate between two competing goods: soon after a node u appears in the order, we want to add many of u ’s neighbors (so that u ’s local error drops quickly), *but* we must avoid too much BFS-style exhaustive exploration of a dense “community” involving u (which would cause the global average, and thus the global error, to spike).

These two successful orderings generally do some BFS-style exploration around words as they are added: e.g., the frequency of *write* and an inflected form like *writes* are quite similar—and their lemma-expanded AoAs are exactly identical—so *writes* comes along soon after *write*. But AoA and frequency avoid immediately flooding the immediate neighborhood: most phonological neighbors of *write* are semantically (and thus morphologically) unrelated to writing, and therefore would not generally have a particularly similar frequency or AoA to *write*. (Note, then, that frequency is on the slightly more global side of the

global/local tradeoff, and AoA is on the slightly more local side. This observation is consistent with their local and global error rates.) Ordering nodes by degree does much worse than ordering by frequency, e.g., despite the positive correlation between degree and frequency [27], the degree order is too local in its exploration and thus suffers in global error.

Node ordering in other complex networks. There is a simple and more basic observation implied by the good results of AoA and frequency in ordering the nodes of the phonological network: that there *exists* some ordering of its nodes that “unkinks” its nodes in a way that leads to a sequence of good approximations to the network as a whole. That observation may say something important about AoA and frequency—or it may say something important about G_{lex} . Indeed, “unkinkability” may point to some extraordinarily odd features of the network. (Some recent research has begun to ask key questions about whether graph-theoretic properties of G_{lex} reflect interesting facts about English, or whether they are simply an artifact of the way that the network is constructed [18, 42, 44, 45].)

Perhaps the most compelling direction for further research on the node ordering problem is this: is there any meaningful analogue to Age of Acquisition in other kinds of complex networks? What happens if one tries to order the nodes of, say, a social network instead?

Although the superficial processes are quite different, after some reflection on the two just-discussed ways that node orders can perform poorly (being too local or not being local enough), an analogy between AoA and “social influence” begins to emerge. For example, in models of the spread of some behavior like the adoption of some new technology, that behavior can fail to spread widely by being too local (a small community adopts but it never spreads beyond that corner of the graph) or by not being local enough (adopters are too far apart, leading to isolated early adopters that have no common neighbors to jointly influence into adopting). It is an interesting open question as to whether ordering nodes by their order of adoption in, e.g., an Independent Cascade-style spread of behavior [22] (or perhaps the “backbone” of a network’s systemic communication lines [24]) might yield good performance.

Indeed, there are several real-world phenomena that seem to exhibit complex, high-dimensional behavior—and yet there is a way to unwind them into linear orders that approximate them remarkably well. This is true of postal codes in the United States, in which physical distance between locations is well approximated by the numerical difference in ZIP codes [3, 23]. It is also true of the web graph, in which links can be represented very efficiently if the underlying graph is stored with its nodes sorted lexicographically by URL [7]. Is there a way to linearly order the nodes of a social network, or indeed any other complex network, in a similar way?

References

1. Ahmed, N., Neville, J., Kompella, R.: Network sampling: from static to streaming graphs. *ACM Trans. Knowl. Discov. Data (TKDD)* **8**(2), 7 (2014)
2. Altieri, N., Gruenenfelder, T., Pisoni, D.: Clustering coefficients of lexical neighborhoods: Does neighborhood structure matter in spoken word recognition. *Mental Lex.* **5**(1), 1–21 (2010)
3. Arbesman, S.: The fractal dimension of ZIP codes. *WIRED* (2012)
4. Arbesman, S., Strogatz, S., Vitevitch, M.: The structure of phonological networks across multiple languages. *Int. J. Bifurc. Chaos* **20**(03), 679–685 (2010)
5. Avidan, S., Shamir, A.: Seam carving for content-aware image resizing. *ACM Trans. Graph.* **26**(3), 10 (2007)
6. Balota, D., et al.: The English Lexicon project. *Behav. Res. Methods* **39**(3), 445–459 (2007)
7. Boldi, P., Vigna, S.: The webgraph framework I: compression techniques. In: *WWW 2004*
8. Brot, H., Muchnik, L., Goldenberg, J., Louzoun, Y.: Evolution through bursts: network structure develops through localized bursts in time and space. *Netw. Sci.* **4**(3), 293–313 (2016)
9. Brysbaert, M., New, B.: Moving beyond Kucera and Francis: a critical evaluation of current word frequency norms and the introduction of a new and improved word frequency measure for American English. *Behav. Res. Methods* **41**(4), 977–990 (2009)
10. Brysbaert, M., Van Wijnendaele, I., De Deyne, S.: Age-of-acquisition effects in semantic processing tasks. *Acta Psychol.* **104**(2), 215–226 (2000)
11. Chan, K., Vitevitch, M.: The influence of the phonological neighborhood clustering coefficient on spoken word recognition. *J. Exp. Psychol. Hum. Percept. Perform.* **35**(6), 1934–1949 (2009)
12. Clauset, A., Shalizi, C., Newman, M.: Power-law distributions in empirical data. *SIAM Rev.* **51**(4), 661–703 (2009)
13. Cortese, M., Khanna, M.: Age of acquisition predicts naming and lexical-decision performance above and beyond 22 other predictor variables: an analysis of 2,342 words. *Q. J. Exp. Psychol.* **60**(8), 1072–1082 (2007)
14. Devanur, N., Khot, S., Saket, R., Vishnoi, N.: Integrality gaps for sparsest cut and minimum linear arrangement problems. In: *STOC 2006*
15. Dhamdhere, K.: Approximating additive distortion of embeddings into line metrics. In: *APPROX/RANDOM 2004*
16. Feige, U., Lee, J.: An improved approximation ratio for the minimum linear arrangement problem. *Inf. Process. Lett.* **101**(1), 26–29 (2007)
17. Girvan, M., Newman, M.: Community structure in social and biological networks. *Proc. Natl. Acad. Sci.* **99**(12), 7821–7826 (2002)
18. Gruenenfelder, T., Pisoni, D.: The lexical restructuring hypothesis and graph theoretic analyses of networks based on random lexicons. *J. Speech Lang. Hear. Res.* **52**(3), 596–609 (2009)
19. Hennessey, D., Brooks, D., Fridman, A., Breen, D.: A simplification algorithm for visualizing the structure of complex graphs. In: *INFOVIS 2008*
20. Hübner, C., Kriegel, H.P., Borgwardt, K., Ghahramani, Z.: Metropolis algorithms for representative subgraph sampling. In: *ICDM 2008*
21. Järvelin, K., Kekäläinen, J.: IR evaluation methods for retrieving highly relevant documents. In: *SIGIR 2000*

22. Kempe, D., Kleinberg, J., Tardos, É.: Maximizing the spread of influence through a social network. In: KDD 2003
23. Kosara, R.: US ZIPScribble map. <https://eagereyes.org/zipscribble-maps/united-states> (2006)
24. Kossinets, G., Kleinberg, J., Watts, D.: The structure of information pathways in a social communication network. In: KDD 2008
25. Kumar, R., Vassilvitskii, S.: Generalized distances between rankings. In: WWW 2010
26. Kuperman, V., Stadthagen-Gonzalez, H., Brysbaert, M.: Age-of-acquisition ratings for 30,000 English words. *Behav. Res. Methods* **44**(4), 978–990 (2012)
27. Landauer, T., Streeter, L.: Structural differences between common and rare words: failure of equivalence assumptions for theories of word recognition. *J. Mem. Lang.* **12**(2), 119 (1973)
28. Lee, M.J., Lee, J., Park, J.Y., Choi, R.H., Chung, C.W.: Qube: a quick algorithm for updating betweenness centrality. In: WWW 2012
29. Leskovec, J., Backstrom, L., Kumar, R., Tomkins, A.: Microscopic evolution of social networks. In: KDD 2008
30. Leskovec, J., Faloutsos, C.: Sampling from large graphs. In: KDD 2006
31. Leskovec, J., Kleinberg, J., Faloutsos, C.: Graph evolution: densification and shrinking diameters. *ACM Trans. Knowl. Discov. Data* **1**(1), 2 (2007)
32. Liben-Nowell, D., Kleinberg, J.: The link-prediction problem for social networks. *J. Am. Soc. Inf. Sci. Technol.* **58**(7), 1019–1031 (2007)
33. Lin, S.D., Yeh, M.Y., Li, C.T.: Sampling and summarization for social networks (Tutorial). In: Pacific Asia Knowledge Discovery and Data Mining (2013)
34. Liu, Y., Safavi, T., Dighe, A., Koutra, D.: Graph summarization methods and applications: a survey. *ACM Comput. Surv.* **51**(3), 62 (2018)
35. Maiya, A., Berger-Wolf, T.: Benefits of bias: towards better characterization of network sampling. In: KDD 2011
36. Maiya, A., Berger-Wolf, T.: Sampling community structure. In: WWW 2010
37. Nagel, T., Duval, E.: A visual survey of arc diagrams. In: IEEE Visualization (2013)
38. Newman, M.: Assortative mixing in networks. *Phys. Rev. Lett.* **89**(20), 208,701 (2002)
39. Rafiee, D., Curial, S.: Effectively visualizing large networks through sampling. In: VIS 2005
40. Ruan, N., Jin, R., Huang, Y.: Distance preserving graph simplification. In: ICDM 2011
41. Sariyuce, A., Kaya, K., Saule, E., Catalyürek, U.: Incremental algorithms for closeness centrality. In: IEEE International Conference on Big Data (2013)
42. Shoemark, P., Goldwater, S., Kirby, J., Sarkar, R.: Towards robust cross-linguistic comparisons of phonological networks. In: Computational Research in Phonetics, Phonology, and Morphology (2016)
43. Siew, C.: The orthographic similarity structure of English words: insights from network science. *Appl. Netw. Sci.* **3**(1), 13 (2018)
44. Stella, M., Brede, M.: Patterns in the English language: phonological networks, percolation and assembly models. *J. Stat. Mech. Theory Exp.* **2015**(5), P05,006 (2015)
45. Turnbull, R., Peperkamp, S.: What governs a language's lexicon? Determining the organizing principles of phonological neighbourhood networks. In: International Workshop on Complex Networks and Their Applications (2016)
46. Vattani, A., Chakrabarti, D., Gurevich, M.: Preserving personalized PageRank in subgraphs. In: ICML 2011

47. Vitevitch, M.: What can graph theory tell us about word learning and lexical retrieval. *J. Speech Lang. Hear. Res.* **51**(2), 408–422 (2008)
48. Wattenberg, M.: Arc diagrams: visualizing structure in strings. In: INFOVIS 2002
49. Watts, D., Strogatz, S.: Collective dynamics of ‘small-world’ networks. *Nature* **393**(6684), 440 (1998)
50. Yates, M.: How the clustering of phonological neighbors affects visual word recognition. *J. Exp. Psychol. Learn. Mem. Cogn.* **39**(5), 1649–1656 (2013)
51. Zachary, W.: An information flow model for conflict and fission in small groups. *J. Anthropol. Res.* **33**(4), 452–473 (1977)



Systematic Biases in Link Prediction: Comparing Heuristic and Graph Embedding Based Methods

Aakash Sinha¹, Rémy Cazabet^{2(✉)}, and Rémi Vaudaine³

¹ Department of Computer Science and Engineering, I.I.T. Delhi, Hauz Khas,
New Delhi 110016, India

² University of Lyon, UCBL, CNRS, LIRIS UMR 5205, 69621 Lyon, France
remy.cazabet@gmail.com

³ University of Lyon, UJM-Saint-Etienne, CNRS, Saint-Etienne, France

Abstract. Link prediction is a popular research topic in network analysis. In the last few years, new techniques based on graph embedding have emerged as a powerful alternative to heuristics. In this article, we study the problem of systematic biases in the prediction, and show that some methods based on graph embedding offer less biased results than those based on heuristics, despite reaching lower scores according to usual quality scores. We discuss the relevance of this finding in the context of the *filter bubble* problem and the *algorithmic fairness* of recommender systems.

Keywords: Graph embedding · Link prediction · Systematic biases · Filter bubble

1 Introduction

Graph data occurs in various real-world applications such as social networks, biological networks, communication networks and many more.

In this article, we focus on the problem of *link prediction* (LP). LP is a classic problem on graph data, with numerous applications, from the identification of missing data to recommender systems. Several surveys have been written on the topic [2, 17, 18, 25]. These articles have notably discussed the best way to evaluate and compare LP methods, and have studied the properties, advantages and drawbacks of different approaches. Studied approaches can be **unsupervised** or **supervised**, and be based on **heuristics** (e.g. the number of common neighbors, Adamic Adar index, etc.) or other approaches (block modeling, random walks, etc.).

Graph embedding is a new technique that gained significant popularity amongst the research community in recent years. Embeddings convert graph data into vectors, creating a representation of nodes in a lower dimensional vector space, on top of which various prediction and detection algorithms can be

applied. These embeddings can be used, for instance, in node classification [6, 13], community detection [12, 23] and role detection [21]. Recent surveys [9, 11, 13] present in details the rationale, the different methods and applications of graph embeddings.

In recent years, several articles (e.g., [13, 14, 23]) have proposed to use **graph embedding** for link prediction. Such papers often claim that these approaches outperform the state of the art. It must be noted, however, that these articles are focused on proposing new embedding approaches, and do not proceed to in-depth evaluation. In particular, they often do not use the quality scores recommended in the literature, and do not investigate the results behind computing such a score.

This article is organized in three parts

1. Using a rigorous evaluation framework, we compare quantitatively recent approaches based on graph embeddings on the task of link prediction with earlier methods
2. We evaluate systematic biases introduced by each approach according to three aspects: the distance in the graph, the degrees of nodes and the community structure
3. We discuss the potential effect of the observed biases in the context of the *filter bubble* problem and the *algorithmic fairness* of recommender systems.

2 Link Prediction Evaluation Framework

In this section, we define a rigorous framework to compare link prediction methods. According to our experience, many recent articles on link prediction do not specify precisely their experimental sections, leading to great difficulties in reproducing the reported results. We split the evaluation framework into three independent steps, each described in the corresponding section:

1. Creation of training and test sets
2. Link Prediction
3. Evaluation using an appropriate score function

2.1 Creation of Learning and Prediction Sets

To evaluate link prediction methods, one starts from a network dataset, and split it into a **learning set** and a **prediction set**, even if the method to test is considered as unsupervised. The learning set is considered the *current state* of the network, based on which predictions are made. The prediction set is considered as the *future* evolution of the network, i.e. the list of edges that will appear and that should be predicted. In this article as in most of the literature, we focus only on *added* edges, and not *removed* ones.

Note that since methods are *supervised*, they need to split the learning set into a training and test set in order to learn where edges should appear. This is independent of the described distinction between learning and prediction set. In

real applications, the input is an observed graph –that can be static or dynamic– and that graph corresponds to our learning set.

Static graph. If the original dataset used for evaluation is a single, static graph, we remove randomly a fraction of edges of a given size. These removed edges constitute the **prediction set**. Following a common practice [13], we ensure that the resulting network is composed of a single connected component, using the following procedure:

- Remove randomly the desired number of edges
- Conserve only the largest component of the resulting graph

Dynamic graph. For dynamic graphs, the order of edge apparition is known. A date is chosen to split the dataset in two: all edges appearing before or at the chosen date constitute the learning set, and all those appearing after the date constitute the prediction set. If the original data contains information on edges that disappear, this information is ignored. To stay coherent with the static case, the single connected component constraint also applies: in the learning set graph, only the largest connected component is conserved.

In our experiments, the learning set is composed of **80%** of the original dataset.

2.2 Link Prediction

A classifier is trained from examples to recognize the pairs of nodes that are the most likely to see edges appear between them. Examples are composed of 50% of *positive* examples (i.e. examples of pairs of nodes that are connected by a link) and 50% of *negative* examples (i.e. pairs of nodes that are **not** connected by a link). Both positive and negative examples are picked randomly from the learning set graph. In this article, we pick $\frac{1}{4}$ of edges in the learning set as positive examples, and an equal number of non-edges as negative examples.

The trained classifier could be used to predict, given an unseen pair of nodes, if it should be connected by an edge or not (Yes/No prediction). However, this does not make sense in realistic settings, since we often want to answer questions such as *which edges might appear in the next hours/day/month?* or *What are the three most likely edges to appear connecting node n_x ?* As a consequence, following previous works [2,14], we assign to each pair of nodes a score corresponding to the *decision function* of the classifier for this sample.

For each experiment, we run the link prediction process 5 times and report the average and standard deviation values.

2.3 Choice of an Appropriate Score Function

To evaluate the quality of the prediction provided by a link prediction method, it is necessary to use a relevant score function.

It must be noted that link prediction is characterized by an extreme class imbalance. On realistic networks of large size, the *density* is low (<0.001%), thus

the number of edges that will appear is much lower than the number of edges that will not appear.

To evaluate a model correctly, the prediction should be made on all possible $|V| * |V| - |E|$ edges. Since this test size can be very large for real graphs, we need to reduce the size of the test set by taking a random sample without introducing any or minimal bias. In this article, we fix a sample size of 500 000. Two important observations must be made about the constitution of this sample:

- Most scores yield very different results according to the fraction of negative examples in the sample. Unlike for classifier training, it is thus essential [25] to keep its original ratio, and not take a 50%/50% sample, which is highly unrealistic.
- Due to the extreme imbalance, the number of positive examples can be extremely low even in a large sample, leading to noise in the results. Therefore we fix a lower value on the number of positive samples (10 in this article).

Two metrics, the Average Precision (AP) and the Area Under Receiver Operating Characteristic curve (AUROC) have been identified [18, 25] to be relevant for this task. AUROC has the property of being independent of the fraction of positive examples in the test set, while AP is favored by some because it gives a higher importance to the first few predictions, that are the most useful in link prediction settings.

3 Methods Evaluation

In this article, we present results on three graphs previously used in articles on graph-embedding techniques.

We selected large graphs, the first two being static and the last one dynamic. These graphs are FACEBOOK [16], ASTROPH [15] and VK [23]. Due to space constraint, the reader can refer to their original descriptions in the referenced articles.

3.1 Method Based on Heuristics

The heuristics used in all experiments are Common Neighbors, Adamic Adar, Preferential attachment, Jaccard Coefficient, nodes degree (for both endpoints) and Resource allocation index (Table 1).

Heuristics can be used for unsupervised prediction if a single of them is used: the score of the heuristic is used directly to rank pairs of nodes. They can also be used for supervised prediction by using some or all of them as features, together with a classifier. In this article, we use all heuristics together and a logistic classifier (implementation of sklearn). This supervised approach has been shown to give the best results [1].

Table 1. Heuristic scores for Link prediction. u, v represent the nodes, $\Gamma(u)$ represents the set of neighbors of node u .

Heuristics	Definition
Common neighbors	$ \Gamma(u) \cap \Gamma(v) $
Adamic Adar	$\sum_{w \in \Gamma(u) \cap \Gamma(v)} \frac{1}{\log \Gamma(w) }$
Preferential attachment	$ \Gamma(u) * \Gamma(v) $
Jaccard coefficient	$\frac{ \Gamma(u) \cap \Gamma(v) }{ \Gamma(u) \cup \Gamma(v) }$
Resource allocation index	$\sum_{w \in \Gamma(u) \cap \Gamma(v)} \frac{1}{ \Gamma(w) }$

3.2 Methods Based on Graph Embeddings

Different embedding algorithms preserve different properties (first/second/higher order proximities) and capture different features. The four embeddings we test are:

- Laplacian Eigenmaps (LE) [4]
- High-Order Proximity preserved Embedding (HOPE) [19]
- node2vec [14]
- Versatile Graph Embeddings from Similarity Measures (VERSE) [23]

Different operators are used to combine the vectors of nodes into edge vectors: *Hadamard* for LE and VERSE, *Normalized Hadamard* for HOPE and node2vec. These operators are those observed by the authors to yield the best results. We use embeddings in 128 dimensions, as is a common practice in the literature.

Due to space constraints, we refer the reader to the original article for each method detailed description. We have chosen these methods due to their frequent use and good results observed in the recent literature, e.g., [13]. We have used the most common parameters. In particular for node2vec, we have systematically tested with parameter sets ($p = 4, q = 0.5$), ($p = 0.5, q = 4$), ($p = 1, q = 1$) and we report the highest score.

3.3 Results

Each method has been tested on each graph. Results are summarized in Table 2. We can observe that, compared with state of the art supervised heuristics approaches, graph embeddings do not yield clear better results. Using the AP score, heuristics always perform best. Using the ROC score, the VERSE algorithm is the only algorithm to obtain higher results.

Table 2. Results for each method on each graph. Results are significant (i.e. Variance is inferior to reported precision).

Graph	Method	Heuristics	LE	HOPE	n2v	VERSE
AP	FACEBOOK	0.74	0.50	0.68	0.44	0.60
	ASTROPH	0.79	0.06	0.43	0.22	0.48
	VK	0.063	—	—	0.006	0.021
ROC	FACEBOOK	0.995	0.994	0.981	0.988	0.995
	ASTROPH	0.988	0.922	0.943	0.973	0.992
	VK	0.87	—	—	0.77	0.89

To confirm this result, we plot the *precision@k* score. In many applications, only the first few predictions are used, thus the relevance of this analysis.

To perform this experiment, we select randomly pairs of nodes that are not linked in the training set until we obtain 1000 positive examples, i.e., edges that do appear according to the ground truth. which allows us to preserve a balance between positive and negative examples coherent with the dataset.

Results are plotted in Fig. 1. We observe that results are coherent with the scores obtained, i.e., heuristics and VERSE tend to give the best scores both for the first few predictions and for a realistic number of predictions (1000, corresponding to the number of real positive examples in the sample). In some settings, some algorithms do not yield useful predictions after the first few hundreds one (LE for ASTROPH dataset, node2vec for VK).

4 Analysis of Systematic Biases

New edges in a network can appear for different reasons, and between nodes with different properties. For instance, in a social network such as Twitter, a new follower relationship might be due to two close friends following each other, to a user following a raising celebrity they recently discovered, to strangers connecting because they think they have a common topic of interest, etc.

A link prediction algorithm might have a tendency to predict some types of links more than others, a phenomenon that we call *systematic bias*. In this article, we focus on biases induced by the network topology. More particularly, we focus on three types of biases:

- Graph distance
- Node degree
- Community structure

To evaluate the biases, we study the evolution of the fraction of new edges verifying a given property. Since predicted edges are ordered, from most probable to less probable, we define the *fraction@k*, corresponding to the ratio of pairs of nodes satisfying this property among the k most likely edges. The dataset is selected as previously explained for the *precision@k*.

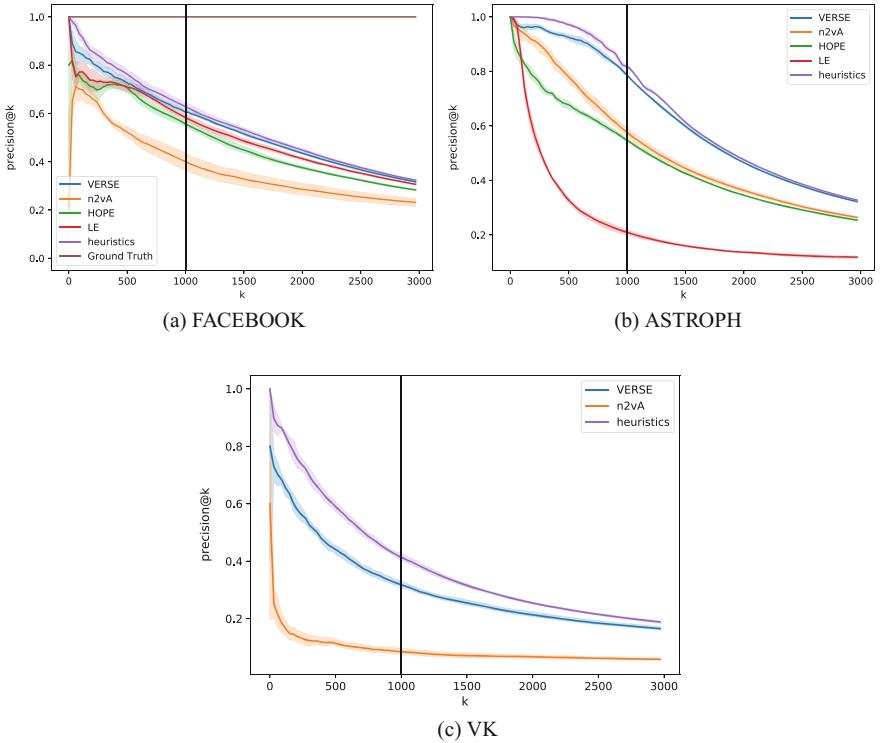


Fig. 1. Precision@k

We define the reference value of *fraction@k* as the value among all edges that do appear in the ground truth (positive examples in the test set). In the scenario of a perfect prediction, the *fraction@k* curve should follow the ground truth scenario until 1000 (corresponding to the real number of observed edges in the test sample), and then move towards the value corresponding to the whole dataset (all pairs of nodes not linked in the training set).

4.1 Graph Distance

Edges can appear between nodes that were *close* or *far* in the graph in term of graph distance, i.e. length of the shortest path between them. It has been observed that most edges appear between nodes at a short distance, a phenomenon often called *triangle closure*. It is intuitively known in social network analysis by the saying "friends of my friends are my friends". Since the number of edges appearing at a distance more than two is usually very low, we consider only two cases:

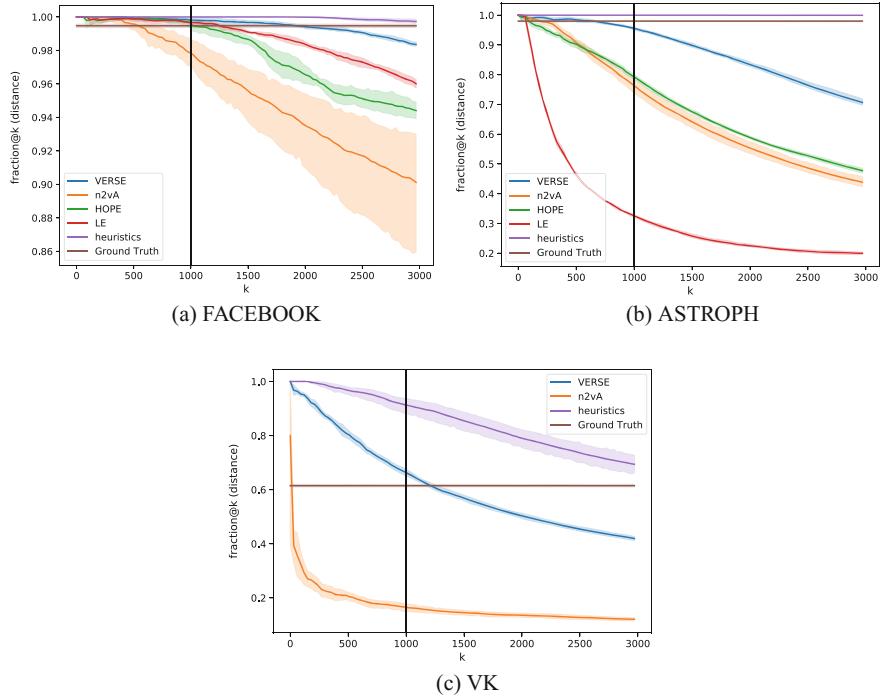


Fig. 2. Ratio@ k for the graph distance.

- Short distance link: the new edge appear between nodes that were previously at distance two in the graph.
- Long distance link: the new edge appear between nodes that were previously at distance three or more.

Figure 2 presents the $fraction@k$ of short distance link for the different methods.

We can make the following observations:

- In the ground truth, most edges appear between nodes at distance 2, although the value is much lower for VK dataset.
- The Heuristic-based approach is highly biased towards predicting short distance links.
- Most other approaches tend to be biased towards short distance links in the first (most probable) predictions, this value later decreases and the fraction often becomes lower than expected when the expected number of edges (1000) is reached.

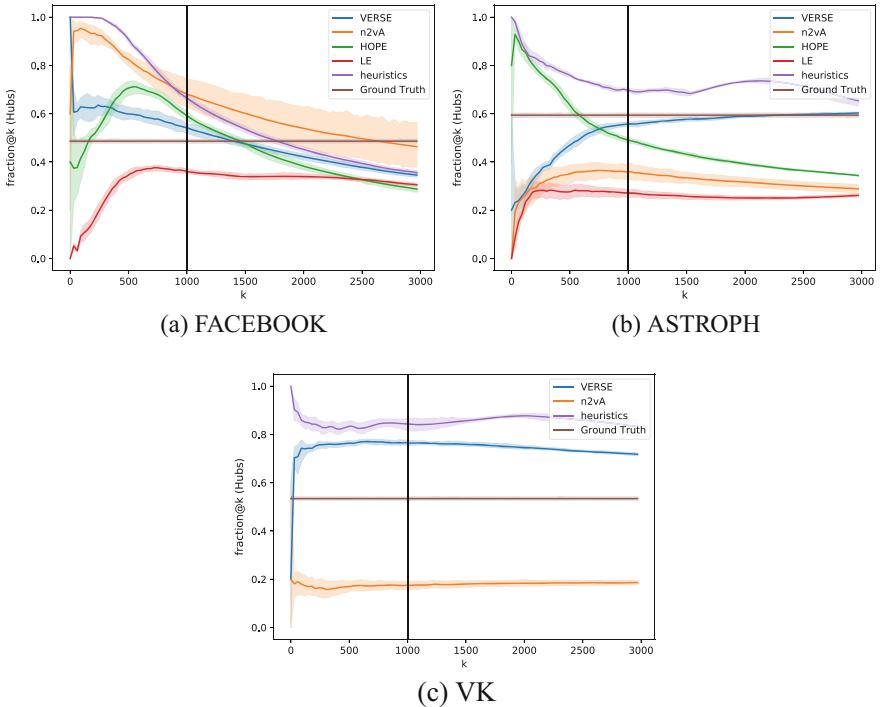


Fig. 3. Fraction@ k for High degree nodes (hubs)

4.2 Node Degree

Most real networks have heterogeneous degree distributions, that can often be approximated by scalefree distributions [3]. In those networks, there is a small fraction of nodes of high degrees that concentrate most of the edges. We define this class of nodes, called *Hubs*, as the 10% of nodes of highest degrees.

Figure 3 presents the *fraction@k* of new links that have at least a Hub among their endpoints.

We can make the following observations:

- In the three studied datasets, the *fraction@k* is between 0.5 and 0.6
- The Heuristic-based approach is highly biased towards high *fraction@k*, i.e. predicting too many edges involving hubs.
- One method seems to be clearly biased towards underestimation (LE), most other methods do not have clear tendencies.
- The VERSE method is the closest to the Ground Truth at the realistic threshold (1000) in all 3 settings.

4.3 Community Structure

Most real networks have mesoscale structures known as communities [12]. The field of community detection and analysis is very active, and there is an intricate relationship between link prediction and community structure [10]. Because communities are dense groups of nodes, edges are more likely to appear inside communities than between them, a property that can be used both for link prediction and for community discovery in dynamic settings [22].

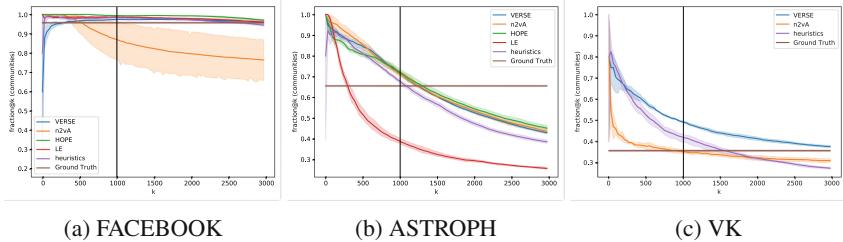


Fig. 4. Fraction@ k for the community structure

We therefore distinguish between two types of edges, those appearing inside communities and those appearing between them.

To discover the community structure, we apply the Louvain algorithm [7], which is the most often used in the literature. This algorithm is based on Modularity optimization, thus, by definition, discover communities denser than the rest of the network. Note that the problem could be studied in more details using overlapping communities, that can introduce different effects, such as higher probabilities of edges on the overlapping parts [24]

Figure 4 presents the *fraction@ k* of new links that appear inside communities, as found by the Louvain Algorithm.

We can make the following observations:

- In different datasets, the fraction of edges in the ground truth appearing inside communities varies widely.
- The Heuristic-based approach is systematically biased towards high *fraction@ k* , i.e. predicting too many edges inside communities.
- Most methods (except n2v in FACEBOOK and LE in ASTROPH) are also biased towards overprediction of internal edges.
- The bias is particularly strong among the first few prediction

5 Discussion: Effect of Biases on Recommender Systems

In recent years, both the scientific community and the civil society have started to pay attention to the problem of biased results provided by machine learning algorithms in real-life applications. Two problems have particularly been identified:

- The filter bubble phenomenon is known to occur at least in two settings: in Online Social Networks such as Facebook and Twitter, and on web search engines such as Google or Bing. The problem comes from a reinforcement phenomenon: machine learning algorithms learn the preferences of users, and show them more results according to these preferences. As a consequence, users see results that are less diverse than they should, but rather biased towards the opinions or interests that the algorithm has inferred they had in the beginning. This process has been accused to amplify the *polarization* phenomenon in political and social opinion [20].
- The problem of fairness of algorithms [5] arises in many real-life applications of machine learning algorithms. In its most famous occurrences, a decision is taken by an algorithm that will impact individuals (who gets a loan, a job, is accepted at a university, is put in a list of potential terrorists, etc.), but the decisions taken by the algorithms are unfair towards a category of people. Typically, the algorithm learns that an ethnic or social group –or, if not present in the data, another highly correlated attribute such as locations or name– correlates with unfavorable outcomes, and therefore learns to discriminate based on this property.

These two problems are now widely known, and solutions have been proposed to mitigate them [8], in particular by ensuring the preservation of some identified properties.

The systematic biases highlighted in the present article however have several interesting features:

- They arise without labeled data, but simply due to the network structure
- The problem is not that unwanted problematic correlations are preserved, but rather that some network properties that were not explicitly required to be preserved are lost.

We can note that the efficiency of the link prediction is not necessarily linked to the absence of bias, on the contrary, methods based on heuristics clearly yield the best results in term of overall link prediction compared with recent techniques based on embeddings. However, in term of biases, we have observed that heuristics usually suffer from the highest biases. Furthermore, those biases are constantly of the same types: they tend to favor nodes at a short distance in the graph, large nodes, and nodes belonging to the same communities. In other terms, these biases can be associated with a **loss in diversity**. In a social network, recommendations of new contacts will favor exaggeratedly either the most similar profiles (same community, closer distance) or the most popular profiles (hubs), but will fail to identify more nuanced possibilities that would add diversity. In product or content recommendation, an algorithm based on link prediction with heuristics will propose the items the most similar to those already consumed, or the most consumed overall, ignoring the relevant possibilities in-between.

New methods based on embeddings might be part of the solution to this problem, as some of them tend to reduce those biases. In particular, it seems

that the recent VERSE algorithm, that offers the best overall results among embedding techniques for link prediction accord to scores and *precision@k*, also consistently show lesser biases than previous techniques.

On top of that, one could follow techniques already proposed [8] to fight against the fairness of algorithm issue, for instance by training separately for the different properties to preserve (e.g., training different classifiers to predict edges at distance 2 and at distance 3 or more).

6 Conclusion

We have shown that the most widely used techniques for link prediction suffer from systematic biases towards some network properties, and that those biases might play a role in the filter bubbles and algorithmic fairness problems.

We have also identified that some of the most recent techniques proposed based on graph embedding are able to somewhat reduce that problem, although they do not yet manage to improve over heuristic based methods for the overall quality of the link prediction.

We think that this work can be extended in two directions:

- On the one hand, we need to understand where do those biases come from. While it can be intuitively understood for heuristics –the features are not learned but selected according to prior knowledge, and they have been designed with triangle closure and preferential attachment properties in mind– the reason why feature learning based methods have similar biases is, to the best of our knowledge, unknown.
- On the other hand, we must develop link prediction methods adapted to mitigate or eliminate those systematic biases. Ideally, such a method should not only remove the three biased identified in this article, but ensure using statistical methods that the network formed by adding the predicted edges has a similar network structure profile than the original network.

References

1. Al Hasan, M., Chaoji, V., Salem, S., Zaki, M.: Link prediction using supervised learning. In: SDM06: Workshop on Link Analysis, Counter-Terrorism and Security (2006)
2. Al Hasan, M., Zaki, M.J.: A survey of link prediction in social networks. In: Social Network Data Analytics, pp. 243–275. Springer (2011)
3. Barabási, A.L.: Scale-free networks: a decade and beyond. *Science* **325**(5939), 412–413 (2009)
4. Belkin, M., Niyogi, P.: Laplacian eigenmaps and spectral techniques for embedding and clustering. In: Advances in Neural Information Processing Systems, pp. 585–591 (2002)
5. Berk, R., Heidari, H., Jabbari, S., Kearns, M., Roth, A.: Fairness in criminal justice risk assessments: the state of the art. arXiv preprint [arXiv:1703.09207](https://arxiv.org/abs/1703.09207) (2017)

6. Bhagat, S., Cormode, G., Muthukrishnan, S.: Node classification in social networks. In: Social Network Data Analytics, pp. 115–148. Springer (2011)
7. Blondel, V.D., Guillaume, J.L., Lambiotte, R., Lefebvre, E.: Fast unfolding of communities in large networks. *J. Stat. Mech. Theory Exp.* **2008**(10), P10,008 (2008)
8. Bolukbasi, T., Chang, K.W., Zou, J.Y., Saligrama, V., Kalai, A.T.: Man is to computer programmer as woman is to homemaker? debiasing word embeddings. In: Advances in Neural Information Processing Systems, pp. 4349–4357 (2016)
9. Cai, H., Zheng, V.W., Chang, K.: A comprehensive survey of graph embedding: problems, techniques and applications. *IEEE Trans. Knowl. Data Eng.* **30**, 1616–1637 (2018)
10. Clauset, A., Moore, C., Newman, M.E.: Hierarchical structure and the prediction of missing links in networks. *Nature* **453**(7191), 98 (2008)
11. Cui, P., Wang, X., Pei, J., Zhu, W.: A survey on network embedding. *IEEE Trans. Knowl. Data Eng.* (2018)
12. Fortunato, S.: Community detection in graphs. *Phys. Rep.* **486**(3–5), 75–174 (2010)
13. Goyal, P., Ferrara, E.: Graph embedding techniques, applications, and performance: a survey. *Knowl.-Based Syst.* **151**, 78–94 (2018)
14. Grover, A., Leskovec, J.: node2vec: scalable feature learning for networks. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 855–864. ACM (2016)
15. Leskovec, J., Kleinberg, J., Faloutsos, C.: Graph evolution: densification and shrinking diameters. *ACM Trans. Knowl. Discov. Data (TKDD)* **1**(1), 2 (2007)
16. Leskovec, J., Mcauley, J.J.: Learning to discover social circles in ego networks. In: Advances in Neural Information Processing Systems, pp. 539–547 (2012)
17. Lichtenwalter, R.N., Lussier, J.T., Chawla, N.V.: New perspectives and methods in link prediction. In: Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 243–252. ACM (2010)
18. Lü, L., Zhou, T.: Link prediction in complex networks: a survey. *Phys. A Stat. Mech. Appl.* **390**(6), 1150–1170 (2011)
19. Ou, M., Cui, P., Pei, J., Zhang, Z., Zhu, W.: Asymmetric transitivity preserving graph embedding, In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 1105–1114. ACM (2016)
20. Pariser, E.: The Filter Bubble: What the Internet is Hiding from You. Penguin, London (2011)
21. Ribeiro, L.F., Saverese, P.H., Figueiredo, D.R.: struc2vec: learning node representations from structural identity. In: Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 385–394. ACM (2017)
22. Rossetti, G., Cazabet, R.: Community discovery in dynamic networks: a survey. *ACM Comput. Surv. (CSUR)* **51**(2), 35 (2018)
23. Tsitsulin, A., Mottin, D., Karras, P., Müller, E.: Verse: versatile graph embeddings from similarity measures. In: Proceedings of the 2018 World Wide Web Conference on World Wide Web, pp. 539–548. International World Wide Web Conferences Steering Committee (2018)
24. Yang, J., Leskovec, J.: Community-affiliation graph model for overlapping network community detection. In: 2012 IEEE 12th International Conference on Data Mining, pp. 1170–1175. IEEE (2012)
25. Yang, Y., Lichtenwalter, R.N., Chawla, N.V.: Evaluating link prediction methods. *Knowl. Inf. Syst.* **45**(3), 751–782 (2015)



Stability and Similarity in Networks Based on Topology and Nodes Importance

Fuad Aleskerov^{1,2} and Sergey Shvydun^{1,2(✉)}

¹ National Research University Higher School of Economics, Moscow, Russia
shvydun@hse.ru

² V.A. Trapeznikov Institute of Control Sciences of Russian Academy of Sciences, Moscow, Russia

Abstract. We propose a model that evaluates how much a network has changed over time in terms of its structure and a set of central elements. The difference of structure is evaluated in terms of node-to-node influence using known nodes correspondence models. To analyze the changes in nodes centralities we adapt an idea of interval orders to the network theory. Our approach can be used to investigate dynamic changes in temporal networks and to identify suspicious or abnormal effects in terms of the topology and its critical members. We can also transform the stability measure to the similarity measure in order to cluster the network in some homogeneous periods. To test our model, we consider the international migration network from 1970 to 2015 and attempt to analyze main changes in migration patterns.

Keywords: Network stability · Similarity measure · Temporal networks · Topology · Dynamics

1 Introduction

Complex systems arise in many applications. The emergence of new data and the increase of its diversity made possible to move from the analysis of monoplex networks to the analysis of more complicated types (multilayer, multiplex, temporal, etc.). However, since most networks evolve either by adding or by removing nodes or links over time, there is a need to understand how much the network has actually changed. For instance, the analysis of traffic networks allows to elucidate suspicious actions by users (Lakhina et al. 2004; Akouglu et al. 2014).

The problem of networks similarity has been studied from many aspects. One of the simplest ways to do it is to calculate correlation coefficients of two adjacency matrices. However, this approach does not take into account topological structure of the network and evaluates all edges equally with no respect whether they connect two disconnected components or two nodes in a dense network. The second approach is to compute the edge and node overlap between two graphs (Broder et al. 1997), to compute the graph edit distance (Bunke et al. 2007), or to compute maximum common edge subgraph (Raymond et al. 2002). However, these approaches also do not take into account topological structure of networks and, moreover, it is not clear how to adapt them to weighted networks. The third approach is to calculate correlation of nodes ranking,

node/edge weight vectors or other characteristics (Borodin et al. 2005; Papadimitriou et al. 2010). In Papadopoulos and Manolopoulos (1999) there was proposed a model that compares nodes degree histograms of two networks. Again, these models lack the topological changes of the network.

In Koutra et al. (2013) the authors proposed a model called Deltacon that, first, computes the pairwise node affinities to preserve the topology of the network using Fast-Belief Propagation algorithm proposed in Koutra et al. (2011) and, second, computes the distance between these affinities for each pair of nodes. The proposed methods consider the topological structure of the network; however, they do not capture how centrality of nodes differs in two networks. Another promising approach for graph similarity is to use Laplacian matrix instead of the adjacency matrix and to use the idea of Von Neumann entropy, which was proposed in De Domenico et al. (2015) for reducibility of multilayer networks. Unfortunately, this model does not take into account individual attributes of nodes and a possibility of the group influence, which makes it necessary to consider edges with the same weights differently for more accurate influence estimation. Thus, we attempt to improve existing measures of networks similarity by taking into account both their topological structure and a set of pivotal members.

Classical measures define the similarity of two graphs based on the closeness of their topological structures. However, for undirected weighted graphs very small changes in the structure or in the weights of the network might lead to crucial or fundamental change in the set of central elements. For this reason, we add one more parameter to evaluate the similarity of the graph in terms of similarity of its central elements. Thus, we define the similarity of two networks using the similarity of their structure and similarity in their central elements. The main idea is that two networks are similar and stable if they have comparable node-to-node influence structure and the same central elements.

The motivation of our paper is that in many applications there is a need to detect the most important elements in temporal networks. Moreover, many anomalies in network may lead to the change of centralities of some members. However, since existing measures do not consider this feature, it is possible that two networks will be topologically similar by classical methods but completely different in terms of the main participants. Our goal is to avoid this problem.

Our model can be divided into the following steps. First, we transform the initial connections in a network to the network of influence using previously proposed approach in Aleskerov et al. (2016). This step can be applied to solve the problem of different scales which we discuss in the next Section. Second, we calculate node-to-node correspondence using known techniques as well as the centrality measure of each node in these two networks. Third, we compare nodes affinity as well as the ranking of nodes using the idea of interval orders. The obtained parameters indicate how a network changed in terms of nodes ranking as well as its topology. Finally, we transform these parameters to the similarity measure, which can be used for networks clustering.

The structure of the paper is organized as follows. First, we discuss the model that allows to transform the initial network to the network of influence and evaluate the centrality of each node. Second, propose an idea how to compare nodes importance using interval orders and how to compare the network structure. We also consider how

to transform the proposed measures to the similarity measure and discuss its properties. Third, we apply the proposed models to the international migration network and provide the obtained results. The final Section concludes.

2 Stability and Similarity of Network

2.1 Network Transformation

We describe the temporal network as a set of T graphs $\{G_t = (V_t, E_t)\}_{t=1}^T$ where V_t is a set of vertices in the network at time t while $E_t \subseteq V_t \times V_t$ is the set of weighted edges. Each edge $(i, j)_t \in E_t$ in a network is associated with weight w_{ij}^t that characterizes the intensity of connection among the vertices. We consider the temporal network as directed, so an edge $(i, j)_t \in E_t$ do not implies the existence of edge $(j, i)_t \in E_t$.

In many applications, the difference between two networks G_t and G_{t+1} is caused by the problem of different scales. For instance, scale, volume and efficiency of international merchandise and commercial services trade have all continued to increase over the last several decades. Therefore, if we consider the international trade network as our temporal network, it is clear that G_{2000} and G_{2018} may be completely different in terms of classical network similarity measures as the total trade value between countries have increased over the last years. However, if we analyze the structure of two networks, it is possible that they will be similar. Since existing similarity measures do not capture this feature, we need to transform the initial network in order to make a more suitable comparison.

In some other applications, there is a problem that the initial connections in a network do not represent the initial picture of influence among the nodes. For instance, the flow of 1,000 people can be important for a very small town (for instance, with population of 2,000 people) but insignificant for a larger city (for instance, with population of 2,000,000 people). Since most real-life networks are not homogeneous, there is a need to take into account metadata of nodes, which can vary in different time slots t , to compare two networks more accurately.

To do this, we use the idea of network transformation proposed in Aleskerov et al. (2016). According to the model, each node i have an individual level of influence q_i^t where it becomes affected. To evaluate influence of node i to node j , there is a need to find a minimal group of nodes that affects node j with a restriction that without node i this group have no influence on node j . If such group exists, the node i is called pivotal and its influence on node j is proportional to the total weight of this group. On the other hand, if no such group is found, node i has no influence on node j , i.e., the connection between these nodes is insignificant and can be removed.

The illustration of the model for a simple network is provided in Fig. 1. We can observe that node 10 does not influence other nodes for $q_i^t = 25\%$ as connections from node 10 to nodes 6–9 do not contribute to the total influence of these nodes. On the other hand, we observe that connections from node 2 to node 1 and from node 1 to node 10 have different weights in the initial network. However, we consider these

connections as equivalent since they individually exceed the required threshold. Finally, connections from node 10 to node 6, from node 8 to node 2 and from node 2 to node 3 have the same weights; however, they are evaluated differently as nodes have different thresholds of influence as well as distinct sets of neighbors.

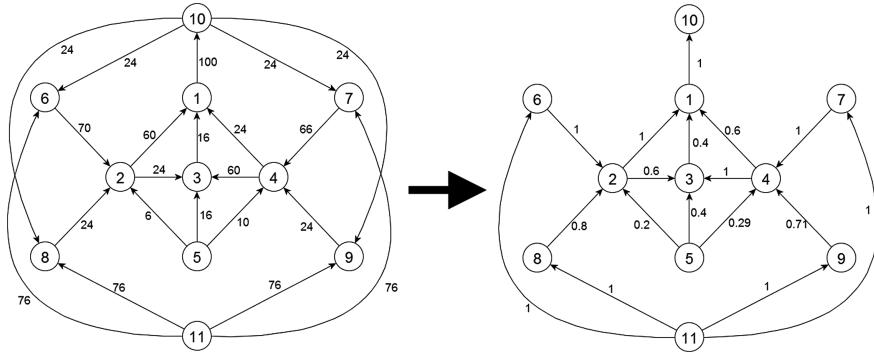


Fig. 1. Network transformation for a simple network if each node has 25% threshold of influence of incoming edges

Thus, we can transform each weight w_{ij}^t of a temporal network into the influence value denoted as c_{ij}^t . A more detailed description of the model is provided in Aleskerov et al. (2017b). We should note the procedure of network transformation is not obligatory; however, it can be used to take into account metadata of nodes and to avoid the problem of different scales, which is frequent in many temporal networks.

2.2 Nodes Similarity Using Interval Orders

As it was mentioned above, we consider two graphs similar if they have comparable topological structure as well as the set of the most important nodes. Since there are numerous approaches to define the central node in a network, we assume that the centrality of nodes was calculated using the most appropriate (for the problem) valuation technique. We denote the centrality of node i by c_i^t while total number of nodes is denoted by n .

There are numerous ways to compare the centralities of nodes in two networks. For instance, we can calculate Pearson's correlation coefficient for networks G_t and G_{t+1} . Unfortunately, the obtained measure do not take into account how the ranking of nodes differ from each other. On the contrary, we can evaluate Spearman's correlation coefficient which is designed to evaluate statistical dependence between rankings; however, the latter method is vulnerable to small variations in data when nodes have similar centrality values.

To avoid the problem, we adapt an idea of interval orders (proposed in Wiener

1914) and further analyzed in Aleskerov et al. (2007) to the network theory. Initially, an interval order is a binary relation P which is constructed as

$$xPy \Leftrightarrow u(x) - u(y) > \varepsilon(y),$$

where x and y are two alternatives, $u(x)$ and $u(y)$ are their utilities, while $\varepsilon(y)$ is an additional parameter ($\varepsilon(y) > 0$), which can be interpreted as a measurement error.

In our case, we introduce a matrix $R^t = [r_{ij}^t]$ that represents information about relative ranking of nodes based on their centralities at time t

$$r_{ij}^t = \begin{cases} 1, & c_i^t - c_j^t > \varepsilon \\ 0, & \text{otherwise.} \end{cases}$$

The choice of parameter ε depend on the problem. In case $\varepsilon = 0$, even small changes in data are important for the problem. In general, we can set the parameter value being equal to $k\%$ of variance in centrality.

The distance between two rankings for networks G_t and G_{t+1} can be evaluated using the Hamming distance

$$d(R^t, R^{t+1}) = \frac{\sum_{j \neq k}^n |r_{ij}^t - r_{ij}^{t+1}|}{n \cdot (n - 1)}.$$

There are several reasons why we use the idea of interval orders. First, it accounts for the relative position of the elements. Second, contrary to Spearman's correlation coefficient, it is not vulnerable to small variations in data as we can set the parameter ε . Third, the proposed method is symmetric and varies from 0 to 1. The obtained distance value have a clear interpretation. If $d(R^t, R^{t+1}) = 0$, the rankings of nodes are equal at time slots t and $t + 1$. On the other way, $d(R^t, R^{t+1}) = 1$ means that the ranking of nodes at time t is inverse to the ranking of nodes for the next period.

2.3 Similarity of Network Topology

There are several ways how to compare network topology at different time slots. For instance, methods described above can be used. However, since we transformed initial connections in a network to the network of influence where connections among nodes are in the same scale and vary from 0 to 1 for all time slots, we propose another model of networks comparison.

First, each edge c_{ij}^t in a network of influence at time t represent direct influence among the nodes. To evaluate node-to-node correspondence (both direct and indirect influence) and preserve information about the network structure, we may calculate different paths between nodes as it was discussed in Aleskerov et al. (2017b). Additionally, we can limit the paths length as in many application long paths have no impact to the real influence. Another way to evaluate nodes affinity is to calculate Personalized Random Walks with Restarts, Belief Propagation or some other models, which were discussed in Koutra et al. (2013).

Denote by \tilde{c}_{ij}^t a value that varies from 0 to 1 and represents information about direct and indirect influence of node i on node j . The distance between networks at two time slots t and $t + 1$ can be calculated as

$$\delta(\tilde{C}_t, \tilde{C}_{t+1}) = \frac{\sum_{j,k}^n |\tilde{c}_{ij}^t - \tilde{c}_{ij}^{t+1}|}{n^2 \cdot \gamma},$$

where

$$\gamma = \max_{j,k}(\tilde{c}_{ij}^t, \tilde{c}_{ij}^{t+1}).$$

The distance $\delta(\tilde{C}_t, \tilde{C}_{t+1})$ is symmetric and varies from 0 to 1. It also has a clear interpretation: $\delta(\tilde{C}_t, \tilde{C}_{t+1}) = 1$ for the case of completely different networks, while $\delta(\tilde{C}_t, \tilde{C}_{t+1}) = 0$ for identical networks. Indeed, if we compare a complete graph and an empty graph (or graphs and their inverse analogues), the distance will be equal to 1.

2.4 Stability and Similarity Measure

We have introduced two parameters that characterize the distance between network at two periods t and $t + 1$. The first parameter $d(R^t, R^{t+1})$ indicates how the set of central nodes has changed in a network. The second parameter $\delta(\tilde{C}_t, \tilde{C}_{t+1})$ indicates the changes in the network structure. We can present the overall changes in a graph as a vector $\vec{s}(G^t, G^{t+1}) = (\delta, d)$ in two-dimensional space (see Fig. 2).

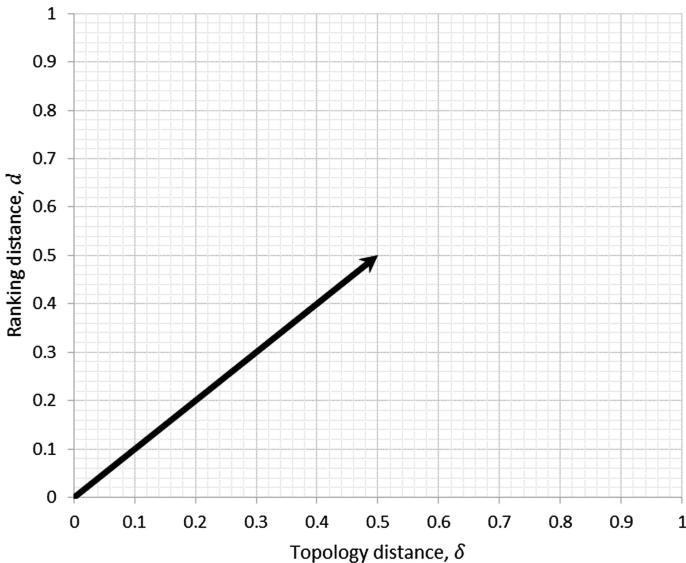


Fig. 2. Stability in networks

Figure 2 represents how network has changed in different periods. In case the obtained vector is a null vector, $\vec{s}(G^t, G^{t+1}) = \vec{0}$, two networks are considered as identical. In case $\vec{s}(G^t, G^{t+1}) = (1, 0)$, the network is completely different in terms of the structure but identical in terms of central elements (for instance, an empty graph and a complete graph). If $\vec{s}(G^t, G^{t+1}) = (1, 1)$, two networks are completely different in terms of both topology and a set of key elements (for instance, chain and inverse chain of nodes). The case when $\vec{s}(G^t, G^{t+1}) \rightarrow (0, 1)$ is more complicated since small changes in a network resulted in a complete instability in terms of its central elements.

The proposed approach can be used in many applications. For instance, we can analyze the changes in a network in order to identify main shifts in a network. We can also use the model to identify anomalies in a network and to prevent possible attacks. Moreover, we can analyze the data consistency and identify cases where some important connections are missed for some period.

We can also transform the proposed measures into the similarity measure. For instance, we can calculate the length of vector \vec{s} which corresponds to the variance in a network.

$$d(G^t, G^{t+1}) = \sqrt{\frac{d(R^t, R^{t+1})^2 + \delta(\tilde{C}_t, \tilde{C}_{t+1})^2}{2}},$$

where $d(G^t, G^{t+1}) = 0$ means that two networks are identical while $d(G^t, G^{t+1}) = 1$ for completely different networks.

However, it that case topology and rankings distances are considered equally, which is not true in many applications. Thus, we propose the following similarity measure between networks G^t, G^{t+1} .

$$d(G^t, G^{t+1}) = \alpha \cdot d(R^t, R^{t+1}) + (1 - \alpha) \cdot \delta(\tilde{C}_t, \tilde{C}_{t+1}),$$

where parameter α corresponds to relative importance of the ranking distance.

Indeed, if $\alpha = 0$, the similarity measure is similar to classical measures as it is based on the structure of the network. If $\alpha = 1$, two networks are similar, if they have identical rankings of nodes disregarding the topology of the network. Finally, $\alpha = 0.5$ for the case when both parameters are considered equally.

The proposed measure can be applied to identify main trends in a network or we can make a pairwise comparison of temporal network at time slots $1, 2, \dots, T$ and apply some clustering procedure (for instance, hierarchical clustering (Johnson 1967; Mirkin 2012) in order to find homogeneous periods or life cycles in a network.

3 Empirical Application: International Migration Network

We have applied the proposed model to the network of international migration. We used the data on dyadic migrant flows from 1970 to 2015 provided by United Nations (2015) where the migration flow is defined as the total number of people arriving to

country or leaving it in a given time period. In Aleskerov et al. (2017a) we have applied the methodology of network transformation based on information about population and the economic situation in a particular country. As a result, we identified the most critical elements for migration process. In Aleskerov et al. (2018) we have also investigated the overall picture of international migration in various decades.

Based on our previous analysis, we can evaluate how the international migration network has changed in terms of migration patterns and the most critical important elements.

In Fig. 3 we present information on how the structure of migration network has changed in adjacent years with respect to the ranking distance $d(R^t, R^{t+1})$ and topology distance $\delta(\tilde{C}_t, \tilde{C}_{t+1})$.

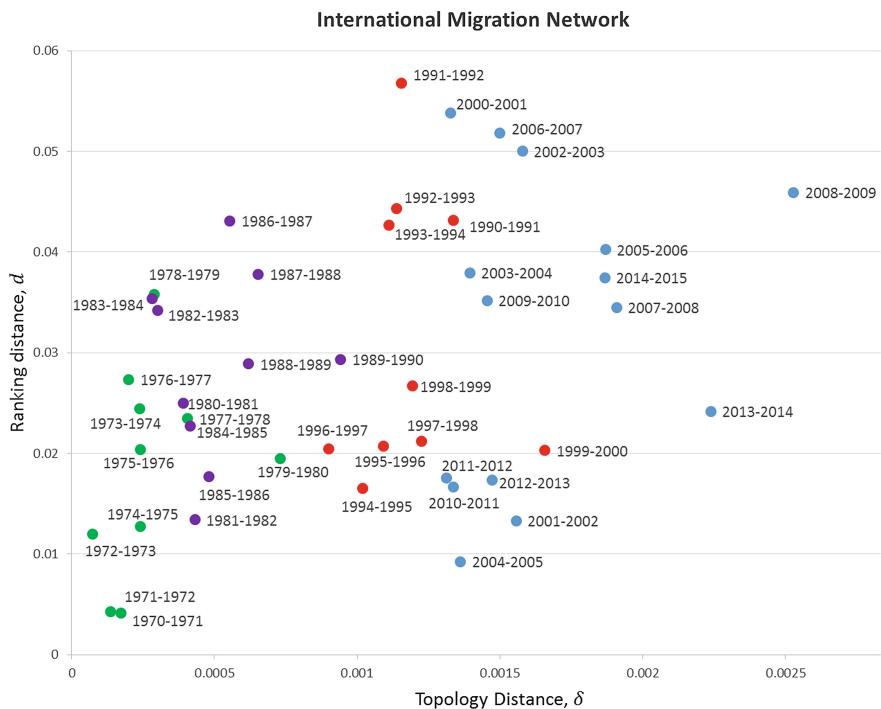


Fig. 3. Ranking and topology distance in international migration network

According to Fig. 3, 1991–1992 is the period with the largest changes in the set of central elements. The collapse of the USSR and appearance of new countries and new international migration flows as well as the increase of existing ones have led to large difference of the set of central elements. We can also observe that although the changes in a network structure are quite similar the following years, the set of the most important elements stabilized by the end of 1990s.

Additionally, we can see that the growth of migrant flows have led to appearance of new migration routes which reflected to the larger changes of the network structure. Interestingly, we can also observe that adjacent years are similar in terms of changes of both topology and ranking distance. This fact might be explained that migration is a gradual process.

We have also calculated the aggregated similarity measure by two proposed approaches with an equal importance of rankings and the network structure (see Table 1). Since the difference in distance is much higher than the difference in a network structure, less similar periods are the periods with the largest difference of centralities.

Table 1. Periods with the lowest similarity measure (TOP-5)

Period	Ranking distance	Topology distance	Similarity 1	Similarity 2
1991–1992	0.0568	0.0012	0.0401	0.0290
2000–2001	0.0538	0.0013	0.0381	0.0276
2006–2007	0.0518	0.0015	0.0367	0.0267
2002–2003	0.0500	0.0016	0.0354	0.0258
2008–2009	0.0459	0.0025	0.0325	0.0242

Thus, we can analyze and compare different periods of the global migration process.

The results are preliminary and required a more detailed study. An analysis of how non-adjacent years are related to each other in terms of the structure and its key elements should be also performed. A statistical validation of the model should be also executed. For the future research, we would be also interested in application of clustering techniques to obtain periods with similar migration trends.

4 Conclusion

We have proposed a measure that evaluates how a network has changed in terms of both its structure and a set of central elements. To avoid the problem of different scales and remove insignificant connections in a network, we transform it to the network of influence. To evaluate how the ranking of nodes are different from each other, we adapt the idea of interval orders, which is not vulnerable to small variations in data. The proposed model was applied to international migration network in order to identify abrupt changes in migration process.

Acknowledgments. The article was prepared within the framework of the Basic Research Program at the National Research University Higher School of Economics (HSE) and supported within the framework of a subsidy by the Russian Academic Excellence Project '5-100'. The empirical application to international migration network was funded by the Russian Science Foundation under grant № 17-18-01651.

References

- Akoglu, L., Tong, H., Koutra, D.: Graph-based anomaly detection and description: a survey (2015). arXiv preprint arXiv: 1404.4679
- Aleskerov, F., Bouyssou, D., Monjardet, B.: Utility Maximization, Choice and Preference, vol. 16. Springer Science & Business Media (2007)
- Aleskerov, F., Meshcheryakova, N., Shvydun, S.: Centrality measures in networks based on nodes attributes, long-range interactions and group influence (2016). arXiv preprint arXiv: 1610.05892
- Aleskerov, F.T., Meshcheryakova, N.G., Rezyapova, A., Shvydun, S.V.: Network analysis of international migration. In: Kalyagin, V.A., Nikolaev, A.I., Pardalos, P.M., Prokopyev, O. (eds.) Models, Algorithms, and Technologies for Network Analysis. Springer Proceedings in Mathematics & Statistics, vol. 197, pp. 177–185. Springer (2014)
- Aleskerov, F.T., Meshcheryakova, N.G., Shvydun, S.V.: Power in network structures. In: Kalyagin, V.A., Nikolaev, A.I., Pardalos, P.M., Prokopyev, O. (eds.) Models, Algorithms, and Technologies for Network Analysis. Springer Proceedings in Mathematics & Statistics, vol. 197, pp. 79–85. Springer (2017)
- Aleskerov, F.T., Meshcheryakova, N.G., Rezyapova, A., Shvydun, S.V.: Network analysis of international migration (2016). arXiv preprint arXiv:1806.06705
- Borodin, A., Roberts, G.O., Rosenthal, J.S., Tsaparas, P.: Link analysis ranking: algorithms, theory, and experiments. ACM Trans. Internet Technol. **5**(1), 231–297 (2005)
- Broder, A., Glassman, S., Manasse, M., Zweig, G.: Syntactic clustering of the web. In: WWW, pp. 393–404 (1997)
- Bunke, H., Dickinson, P.J., Kraetzel, M., Wallis, W.D.: A Graphtheoretic Approach to Enterprise Network Dynamics. Birkhäuser, Boston (2007)
- De Domenico, M., Nicosia, V., Arenas, A., Latora, V.: Structural reducibility of multilayer networks. Nat. Commun. **6**, 7864 (2015)
- Johnson, S.C.: Psychometrika **32**, 241 (1967)
- Koutra, D., Ke, T.Y., Kang, U., Chau, D.H., Pao, H.K.K., Faloutsos, C.: Unifying guilt-by-association approaches: theorems and fast algorithms. In: Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML PKDD), Athens, Greece (2011)
- Koutra, D., Vogelstein, J., Faloutsos, C.: Deltacon: a principled massive-graph similarity function. In: Proceedings of the 13th SIAM International Conference on Data Mining (SDM), Texas-Austin, TX (2013)
- Lakhina, A., Crovella, M., Diot, C.: Diagnosing network-wide traffic anomalies. In: Proceedings of the ACM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM’04), pp. 65–76. ACM Press, Portland, Ore, USA, Aug 2004
- Mirkin, B.: Clustering: A Data Recovery Approach, 2nd edn. Chapman and Hall/CRC (2012)
- Papadopoulos, A.N., Manolopoulos, Y.: Structure-based similarity search with graph histograms. In: Proceedings Tenth International Workshop on Database and Expert Systems Applications. DEXA 99, Florence, Italy, pp. 174–178 (1999)
- Papadimitriou, P., Dasdan, A., Garcia-Molina, H.: J. Internet Serv. Appl. **1**, 19 (2010)
- Raymond, J.W., Gardiner, E.J., Willett, P.: RASCAL: calculation of graph similarity using maximum common edge subgraphs. Comput. J. **45**(6), 631–644 (2002)
- United Nations, Department of Economic and Social Affairs, Population Division. International Migration Flows to and from Selected Countries: The 2015 Revision (POP/DB/MIG/Flow/Rev.2015) (2015)
- Wiener, N.: A contribution to the theory of relative position. Proc. Camb. Philos. Soc. **17**, 441–449 (1914)



Delusive PageRank in Incomplete Graphs

Helge Holzmann², Avishek Anand¹, and Megha Khosla²(✉)

¹ L3S Research Center, Leibniz University, Appelstrasse 9a, 30169 Hannover,
Germany

² KBS, Leibniz University, Appelstrasse 4, 30169 Hannover, Germany
khosla@l3s.de

Abstract. Most real-world graphs collected from the Web like Web graphs and social network graphs are *incomplete*. This leads to inaccurate estimates of graph properties based on link analysis such as PAGERANK. In this paper we focus on studying such deviations in ordering/ranking imposed by PAGERANK over incomplete graphs. We first show that deviations in rankings induced by PAGERANK are indeed possible. We measure how much a ranking, induced by PAGERANK, on an input graph could deviate from the original unseen graph. More importantly, we are interested in conceiving a measure that approximates the rank correlation among them without any knowledge of the original graph. To this extent we formulate the HAK measure that is based on computing the impact redistribution of PAGERANK according to the local graph structure. Finally, we perform extensive experiments on both real-world Web and social network graphs with more than 100M vertices and 10B edges as well as synthetic graphs to showcase the utility of HAK.

1 Introduction

Most real-world graphs collected from the Web like Web graphs and social network graphs are *incomplete* or in other words their graph topology is not known in entirety [19], especially if not crawled for a particular purpose or subset, but extracted from existing crawls, such as Web archives. The goal of Web archive crawlers is to capture as much as possible starting from some seed set within some national domain or even broader, given the available but limited resources [6]. Incompleteness is an inherent trade-off already in the design decision of such an archive. Complicating matters further, Web archives are often not constructed in one piece but by merging partial crawls [13]. Additional reasons for the incompleteness in Web archives include the restrictive *politeness* policies (i.e., *robots.txt*) or random timeouts of Web servers. Several studies on this topic have shown that incompleteness is indeed a common issue [15], inevitably affecting the graphs extracted from such crawls as well.

As a result, important graph properties and measures used for link analysis and structural characterization like *authority of vertices* might be inherently flawed or exhibit deviations from their original values. This is commonly observed where users are typically agnostic to the incompleteness of the obtained graph,

hoping that the input graph is a reasonable representative sample of the underlying (unseen) original graph. Some of the well-known measures for computing authority of vertices or relative ordering of vertex authorities based on random walks are PAGERANK [22] and its variants [11, 17].

As an example, consider PAGERANK computed over the .gov Web graph that we will analyze in detail later in this work. Here, the `women.nasa.gov`(*Women@NASA*) page has a high PAGERANK value and is subsequently found within the top 300 pages. However, on a closer examination we observe that most of its PAGERANK is contributed by an in-link from the highly popular NASA homepage (`nasa.gov`). If for some reason this particular in-link is not crawled, e.g., due to a temporary downtime or the decision by *NASA* to exclude their homepage from being crawled, this would cause a large decrease in its PAGERANK and hence a severe rank deviation in the obtained crawl.

One might argue that this is an unlikely case since *important* pages enjoy a high priority and are therefore commonly crawled, but this might not always be the case in reality. To support our claim we performed the following experiment. We ranked pages in a graph constructed from a .de Web archive in 2012¹ based on (1) *inlinks* and (2) PAGERANKS. The above mentioned graph considered only links that emerged in 2012 [12]. We then checked if the top ranked pages in this incomplete graph were indeed archived in that year. Our experiments show that from among the top 1000 pages, ranked according to inlinks, roughly 30% are contained in the archive. According to PAGERANK rankings, less than 20% of the top 1000 pages are contained in the archive. With this small experiment we show that high priority vertices can indeed be missed in real world crawls, which can further cause a rank deviation in the obtained incomplete graph.

We, therefore, study the deviation in orderings/rankings imposed by PAGERANK over incomplete graphs. Vertices in our input crawls are either *completely crawled* (all neighbors are known) or are *uncrawled* (none of their neighbors are known), which we refer to as *ghost vertices*. Based on this, the research questions we ask are the following:

- **RQ I :** *Do incomplete real-world graphs show a deviation in their PAGERANK orderings when compared to full network topology?*
- **RQ II :** *How can we reliably measure the extent of such ranking deviations for incomplete graphs?*

Towards these, we perform extensive experiments on both real-world Web and social network graphs with more than 100 million vertices and 10 billion edges. We first establish empirically that real-world networks indeed show a deviation in their PAGERANK orderings when not crawled completely compared to the complete graph (**RQ I**). We observe ranking correlations (measured by *Kendall's Tau*) dropping down to 0.55 on Web graphs when only 50% of it is crawled. Second, users and applications that use rankings induced by PAGERANK as a feature for downstream ranking and learning tasks would naturally be interested in estimating such a deviation from the (incomplete) input graph at hand as a

¹ The archive has been generously provided to us by the Internet Archive.

measure of confidence. Therefore, as an answer to **RQ II**, we propose a measure called HAK (an acronym of the authors' lastnames) that estimates the ranking deviation of an incomplete input graph when compared to the original graph.

2 Related Work

The authors in [21] analyzed the conditions under which eigenvector methods like PAGERANK and HITS can provide reliable rankings under perturbations to the linkage patterns for a given collection. In particular, when some high ranked page is missed as we discussed in the previous section, the resulting PAGERANK rankings will be highly unstable. Boldi et al. [3] also show the paradoxical effects of PAGERANK computation on Web graphs. They however focus on crawling strategies to preserve page rank computation. In [26] the authors operate on a given subset of vertices and consider the general problem of maintaining multi-scale graph structures by preserving a distance metric based on PAGERANK among all pairs of sampled vertices. Other authors investigated this problem before as well, however, none of them focused on random walk algorithms, such as the widely used PAGERANK, neither explored the effect of missing nodes in real-world Web graphs [23, 24, 27].

The other area of related work comprises of graph sampling approaches which can be broadly classified into two categories: *traversal based* methods [18, 20, 28] and random walk based methods [14, 19]. Graph-traversal based methods employ breadth-first search (BFS) or the depth-first search (DFS) algorithm to sample vertices and are typically shown to exhibit bias towards high-degree vertices [28]. [20] compare various traversal based algorithms and define representativeness of a sample while proposing how to guide the sampling process towards inclusion of desired properties. On the other hand, the random walk based methods are popular for graph sampling because they can produce unbiased samples or generate samples with a known bias [14, 19, 29]. One of the popular sampling algorithms used for Web graphs is the *Forest Fire* algorithm by [18], a generative graph model, in which new edges are added via an iterative “forest fire” burning process where it is shown to produce graphs exhibiting a network community profile plot similar to many real-world graphs. We use this approach in generating synthetic real-world graphs.

3 Preliminaries and Problem

PageRank. As originally conceived, PAGERANK ranks vertices of a directed graph $\mathcal{G} = (V, E)$ where V and E are the vertices and edges respectively, based on the topological structure of the graph using random walks [22]. The problem we are addressing in this paper is attributed to this random walk model behind PAGERANK, representing the *authority* or *importance* of a vertex.

For some fixed probability α , a surfer at vertex $v \in V$ jumps to a random vertex with probability α and goes to a linked vertex with probability $1 - \alpha$. The *authority* of a vertex v is the expected sum of the *importance* of all the vertices

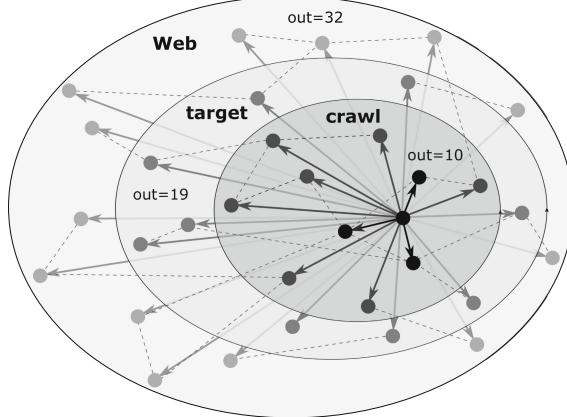


Fig. 1. The neighborhood of a webpage in different subgraphs of the Web.

u that link to v . Consequently, a vertex receives a high PAGERANK value and is ranked at the top by ordering the webpages by *importance* when it is either connected by many incoming edges or reachable from another *important* page.

We first define the notions of *target graph*, *crawl* and *ghost vertices* in the context of incompleteness in graphs due to their collection process:

Definition 1 (Target graph). *The subset of vertices (with the induced edges) of a larger graph (e.g., the Web) that is theoretically reachable by a crawler given its seeds, e.g., a domain, a top-level domain, or all webpages that belong to a certain topic in case of focused crawlers. This graph would be available if every link was followed and every page captured by the crawler, illustrated by the target in Fig. 1.*

Definition 2 (Crawled graph or Crawl). *The (incomplete) graph derived from the set of webpages that have actually been visited by the crawler, discovered/linked yet uncrawled pages are not included. This subset of the target graph is illustrated by the crawl in Fig. 1.*

Definition 3 (Ghost vertex). *Although a hyperlink on a crawled page points to another page that belongs to the target graph, there is a chance the crawler never visited and saved that page, i.e., it is not part the crawl. Such a page or vertex is referred to as ghost vertex, shown by the gray vertices outside the crawl in Fig. 1.*

Ranking Deviations. The deviation among two rankings induced by PAGERANK is a global objective, independent of a specific query. Hence, local or relevance-based measures such as nDCG are not applicable here. The most common metrics to quantify rank correlation are *Spearman's Rho* and *Kendall's Tau*, which are both similar as they are special cases of a more general correlation coefficient and measure relative displacements.

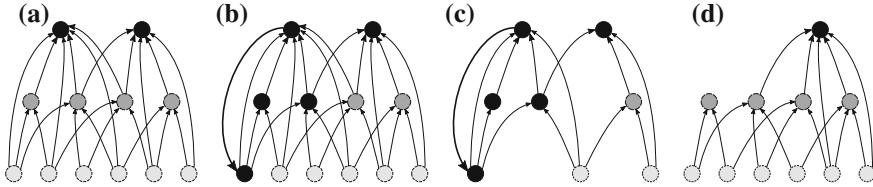


Fig. 2. Example graphs : Darker vertices have a higher *importance* (cp. Sect. 3).

In this work, we use Kendall's Tau [16], ranging from $[-1, 1]$, with 1 corresponding to a perfect rank correlation, 0 corresponding to no correlation and -1 to a perfect inverse correlation, to compare the correlation/deviation of rankings computed on the vertices of a crawl \mathcal{G}_C with respect to that of the target graph \mathcal{G}_T .

In Fig. 2 we provide a few examples of possible graph structures, where partial knowledge of the graph may affect the ranking returned by the PAGERANK values. We remark that in the next sections, we will also provide empirical evidence, supporting the fact that there exists a ranking deviations in crawls of some real-world graphs. In the first subfigure (a), we show the positive case of a DAG where the partial knowledge of the graph will not cause any ranking deviations. As only the topmost vertices shown here receive significantly more links than the others, these are also the most *important* vertices. It is easy to see here that generating a crawl from this structure by removing some vertices will not cause any significant changes in the ranking orderings of the crawl. In the next subfigure (b), a *backlink* has been introduced (left) that feeds back the importance of a top most page to a previously unimportant page and its successors. This importance gets propagated through the cycle which has been created due to the inserted *backlink*. In the next subfigure (c), we illustrate the case of a crawl in which vertices are removed uniformly at random. The chances here are that primarily unimportant vertices are removed, which would still not cause much deviations in the ranking orderings. Finally, if we remove any vertex from the cycle as shown in subfigure (d), its succeeding vertices drastically lose in importance and hence, the ranking among the pages in the crawl changes noticeably.

4 The HAK Measure

With our measure, we estimate quantitatively how reliable a crawl is with respect to the relative ordering of the PAGERANK values on its vertices compared to the corresponding target graph. To this end, we first try to **estimate the size of the target graph**: Given the crawled vertex set and the distinct hyperlinks on the corresponding webpages, some of which are pointing to an uncrawled page (ghost vertex), how big is the target graph or a subgraph that would potentially impact or contribute to the PAGERANK values of the vertices in the crawl? We show that for simple crawling strategies where it can be assumed that each vertex is part of

the crawl independently from all other vertices with some sampling probability p_s , the size of the target graph can be estimated in terms of a very simple property of the crawled vertices, namely, the fraction of its crawled neighbors, referred to as *fidelity*. Secondly, we try to **estimate the impact** exerted by the vertices in the target graph on the crawled vertices, which we in turn use to estimate the number of discordant pairs in the expected rankings, like in Kendall's Tau. Let C denote the set of vertices of the *crawl graph* and let n be the number of vertices in this graph. The main steps in our computation are as follows:

1. Estimate the size of the target graph by using connectivity properties of the crawl. Let T represent the set of vertices in this target graph.
2. Estimate the *impact* (as functions of PAGERANK) of the vertices in C .
3. Assume that the vertices in T exert similar impacts on other vertices.
4. Estimate the number of discordant pairs due to impacts exerted by vertices in $T - C$ on vertices in C .

Estimating the Target Graph. Let \mathcal{N} denote the number of vertices in the target graph. We assume that the crawl is constructed by sampling vertices from the target graph independently and uniformly at random with some probability p_s . We first estimate p_s from the connectivity of the crawl, using a property that we refer to as **fidelity**. Let $d_c(v)$ count the number of vertices $v' \in C$ reachable from v in one step. $d(v)$ denotes the total out-degree of v in the target graph.

Definition 4 (Fidelity). The fidelity of a vertex $v \in T$, $\gamma(v)$, is given by $\gamma(v) = \frac{d_c(v)}{d(v)}$ and the average fidelity of all vertices in C is $\gamma(C) = \frac{\sum_{v \in C} \gamma(v)}{n}$.

We will now show that $\mathbb{E}(\gamma(v)) = p_s(1 - \mathbb{P}(d(v) = 0))$.

Proposition 1. Let for some $0 < p_s < 1$, each vertex in the target graph is sampled independently and uniformly at random with probability p_s . For any $v \in T$, $\mathbb{E}(\gamma(v)) = p_s(1 - \mathbb{P}(d(v) = 0))$.

Proof. The probability that a vertex has fidelity ℓ/k is given by

$$\mathbb{P}\left(\gamma(v) = \frac{\ell}{k}\right) = \mathbb{P}(d_c(v) = \ell | d(v) = k) \cdot \mathbb{P}(d(v) = k) = \binom{k}{\ell} p_s^\ell (1 - p_s)^{k-\ell} \mathbb{P}(d(v) = k).$$

The expected value of fidelity of T can now be computed as

$$\begin{aligned} \mathbb{E}(\gamma(v)) &= \sum_{k \geq 1} \sum_{\ell \leq k} \frac{\ell}{k} \binom{k}{\ell} p_s^\ell (1 - p_s)^{k-\ell} \mathbb{P}(d(v) = k) \\ &= p_s \sum_{k \geq 1} \mathbb{P}(d(v) = k) \sum_{\ell=1}^k \binom{k-1}{\ell-1} p_s^{\ell-1} (1 - p_s)^{k-\ell} = p_s(1 - \mathbb{P}(d(v) = 0)). \end{aligned}$$

With p_s as the sampling probability, $\mathcal{N} \cdot p_s$ gives us the expected number of vertices in the crawl. Using Proposition 1 we obtain $\mathcal{N} = \frac{\mathbb{E}(|C|)}{\mathbb{E}(\gamma(v))}(1 - \mathbb{P}(d(v) = 0))$. We note that for Web graphs $\mathbb{P}(d(v) = 0)$ is the probability that a webpage has no links to other webpages, i.e., there exists a page with pure text and no links. Such a scenario is extremely rare on the Web. Moreover, for synthetic graphs generated using $G_{n,p}$ one can show that $\mathbb{P}(d(v) = 0) = e^{-O(np)}$, which goes to zero for $n \rightarrow \infty$ and constant p . Hence, using the observed average $\gamma(C)$ and the observed size of the crawl, i.e., n , ignoring the multiplicative factor of $1 - \mathbb{P}(d(v) = 0)$ (as $\mathbb{P}(d(v) = 0) = o(1)$ for all practical purposes), we can approximate \mathcal{N} as $\frac{n}{\gamma(C)}$.

PageRank and Impacts. Despite its incompleteness, PAGERANK can be computed on the crawl graph by treating the ghost nodes as dangling nodes. We use the *personalized* variant of PAGERANK for this, starting from the available nodes in C as seeds (s. Sect. 5). Given this, for any vertex v in the crawl C , let $\pi(v)$ denote the value computed by PAGERANK and let $N(v)$ denote the set of immediate neighbors of v . PAGERANK of any vertex u can now be considered as: $\pi(u) = \sum_{v:u \in N(v)} \frac{\pi(v)}{d(v)}$.

We introduce a new property, referred to as **impact**. The impact of a vertex $v \in C$ on one of its neighbors $u \in N(v)$ is defined as: $Im(v, u) = \frac{\pi(v)/d(v)}{\pi(u)}$. Hence, the total impact on any vertex $u \in V$, received from all its incoming edges, is $\frac{1}{\pi(u)} \sum_{v:u \in N(v)} \frac{\pi(v)}{d(v)}$, which is always 1. This implies that any extra impact of x on a vertex will increase its PAGERANK by x times the current PAGERANK. The total impact of a vertex v , $Im(v)$ is then defined as:

$$Im(v) = \sum_{u \in N(v)} Im(v, u) = \sum_{u \in N(v)} \frac{\pi(v)/d(v)}{\pi(u)} = \frac{1}{d(v)} \sum_{u \in N(v)} \frac{\pi(v)}{\pi(u)}.$$

We denote the average of impacts of vertices in C as $m(C) = \frac{\sum_{v \in C} Im(v)}{n}$.

Estimating the Impact of Ghost Vertices. We next compute the impact that could have been exerted by the ghost vertices on the crawled vertices, if the graph was complete and the ghost vertices existed. In a setting like ours, where the PAGERANK is computed from the perspective of the known crawl, the ghost nodes cannot have a bigger impact on the crawl than previously *leaked* to them. Therefore, we build on the assumption that the impact of each vertex in the complete target graph T is on average the same as for the crawl: $Im(C)$. Hence, we approximate the impact exerted by ghost vertices only as follows:

$$\mathcal{I} = |T - C| \cdot Im(C) = n \left(\frac{1}{\gamma(C)} - 1 \right) \cdot Im(C).$$

Some of this extra impact, generated due to ghost vertices, will be acquired by some or all of the vertices in C , changing their PAGERANK values accordingly. This is what eventually will lead to the deviation in rankings. The impact of the ghost vertices can be divided among the vertices of the crawl in several ways. For example, it can happen that the vertex with the lowest PAGERANK receives the total impact, increasing its PAGERANK by a large factor. In this case the number

of discordant pairs is upper bounded by $n - 1$. Moreover, we know from [21] that vertices with low original PAGERANK scores will also have a low PAGERANK value in slightly modified graphs. Therefore, the effect of the loss of information because of incomplete crawls is observed mostly on the PAGERANKS of the nodes higher in the original ranking. We checked experimentally several variants for impact distributions and the best variant, which is affirmative with our tests on real-world and synthetic graphs, is to distribute the total impact \mathcal{I} equally among all vertices. Hence, the **expected number of impacted vertices** that belong to the crawled set will be: $I = \mathcal{I} \cdot \gamma(C)$. In the worst case, each of these impacted vertices will result in forming a discordant pair with each of the unaffected vertex, resulting in a number of discordant pairs of $D = (n - I) \cdot I$. Based on that, HAK is computed with respect to Kendall's Tau as follows:

$$HAK = \frac{\#\text{concordant pairs} - \#\text{discordant pairs}}{\#\text{total pairs}} = \frac{\frac{n(n-1)}{2} - D - D}{\frac{n(n-1)}{2}} = 1 - 4 \cdot \frac{D}{n(n-1)}.$$

5 Experiments

The objective of this evaluation is to assess ranking deviations using Kendall's Tau (cf. Sect. 3) for rankings induced by PAGERANK, computed on a complete target graph vs. an incomplete crawl and compare it against our HAK measure. In contrast to the Kendall's Tau formula used in HAK, for assessing the real rankings, ties were considered by using a variant, also known as *Tau-b*. We **focus only on high-ranked vertices**, as these are typically more interesting in most practical scenarios [21]. We compared the ordering among the top 30%, top 50% and top 70% vertices of the crawl and target graph that appeared in both graphs according to the corresponding PAGERANK values.

The rankings for each of the graphs are computed based on the PAGERANK values. While we employed the regular version PAGERANK on the crawl (with added ghost vertices as sinks), we used the *personalized* variant of PAGERANK for running it on the target graph. The resulting PAGERANK values can be interpreted as their importance with respect to these vertices or the domain represented by the crawl. Both variants of PAGERANK ran for 30 iterations with the damping factor parameter set to the frequently cited value of 0.85.

Experimental Setup. Our experiments require both target graphs and the crawls necessary in order to compute how the rankings on both graphs differ and to evaluate the performance of HAK to estimate this deviation. In reality, neither obtaining the complete target graph is possible nor the actual crawl policy can be determined accurately. To this extent, we consider very large (as complete as possible) real-world graphs under the assumption that those graphs are complete (Table 1). We additionally simulate alternative topological structures by generating synthetic graphs (Table 2). We then simulate crawls on these graphs using different crawling strategies. For all graph and crawl combinations

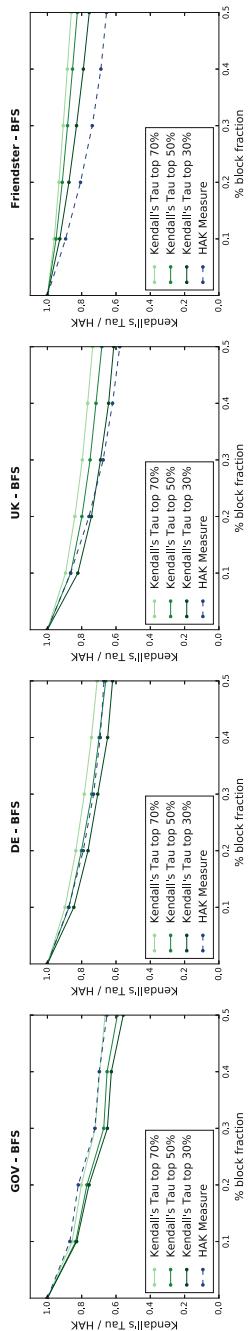


Fig. 3. Ranking deviations measured and estimated for real-world graphs and crawls for different fractions of uncrawled vertices.

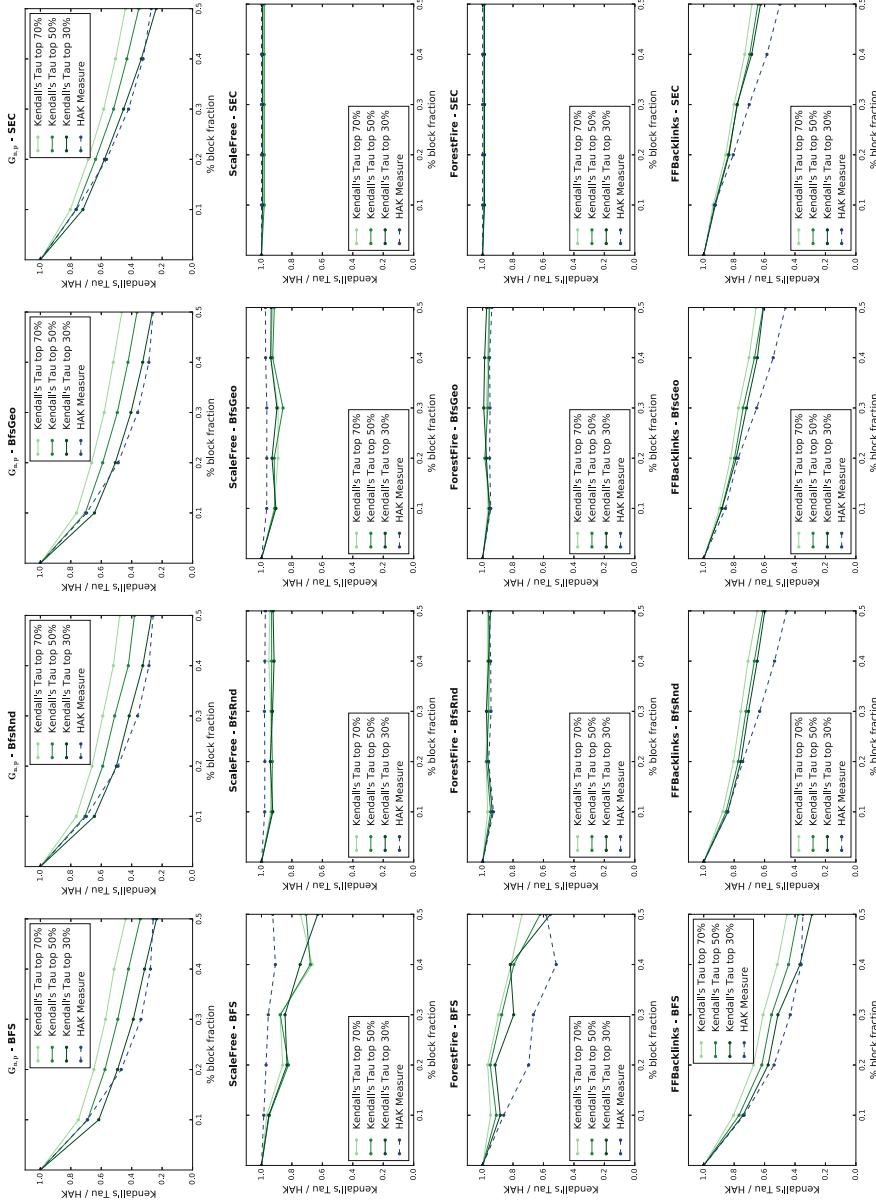


Fig. 4. Ranking deviations measured and estimated with different synthetic graphs and crawls for different fractions of uncrawled vertices (rows) as well as different crawling strategies (columns).

Table 1. Statistics on the studied real-world graphs (# V : original number of vertices, # E : original number of edges, # V_{target} : #target vertices, # E_{target} : #target edges).

	GOV [25]	DE ⁴	UK [4]	Friendster [2]
# V	301,128,778	247,641,473	39,454,746	68,349,466
# V_{target}	5,418,054	133,895,590	38,838,959	61,100,375
# E	2,111,229,433	14,795,732,782	936,364,282	2,586,147,869
# E_{target}	180,657,788	10,085,242,536	928,939,162	2,575,600,737

Table 2. Synthetic graphs (all have 10,000 vertices).

Graph generator	#Edges	Parameters
$G_{n,p}$ [8,9]	299,722	$p \approx 0.0003$ (based on # E in Table 1)
ScaleFree[5]	21,732	$\alpha = 0.41, \beta = 0.54, \gamma = 0.05$ (default)
ForestFire[18]	87,060	$p_f = 0.37, p_b = 0.32$ (most realistic [18])
FFBacklinks	96,262	$p_f = 0.37, p_b = 0.32, p_{\text{backlink}} = 0.0005$

we ran PAGERANK on both crawl and target graphs and compared the rankings using Kendall’s Tau to evaluate HAK.

Crawls and Ranking Deviations in Graphs. We aim to answer **RQ I** and justify the need for estimating ranking deviations before employing PAGERANK for incomplete graphs. We clearly observe that all real-world graphs exhibit a decreasing τ with increasing block fraction (see Fig. 3). Most acutely, τ decreases to 0.55 for the GOV.

Synthetic graphs like $G_{n,p}$ and FFBacklinks (first and last row in Fig. 4) exhibit a similar trend with τ decreasing for increasing block fraction. On the other hand, for the ScaleFree (second row) and ForeFire graphs (third row), we do not witness much change in the ranking orderings, except in the BFS crawls.

A detailed study of the crawls reveals the reasons for such disparate trends for ScaleFree and ForeFire: the crawling strategy combined with the underlying structural properties of the graph sometimes lead to extremely small crawls ($n < 1,000$), much below the desired fraction. First, we observe a scarcity of *backlinks* in ForestFire and ScaleFree. That leads to these graphs to be *DAG-like* without an inadequate number of cycles in the corresponding graphs (cp. Sect. 3). PAGERANK computations over such graphs tend to finish quickly since the lack cycles prohibit the random walk to re-cycle back into the graph. This results in small high-fidelity crawls that do not exhibit large ranking deviations when highly linked vertices are prioritized, explicitly (SEC) or by chance (BfsRnd and BfsGeo). Only the BFS strategy that explicitly blocks random vertices causes a deviation in these crawls, as top vertices may be missed as well (conceivable on the Web for different reasons, e.g., restrictive policies and random failures).

Reinforcing our claim, the addition of backlinks in FFBacklinks resulted in a growing ranking deviation with increasing block fraction. We argue that most

of the real-world graphs will not be DAG-like and will have *backlinks* inducing large cycles. Moreover, the random walk nature of PAGERANK computation increases the importance of these *backlinks* (or feedback loops) towards reaching an equilibrium state. As the core structure of FFBacklinks still resembles the original ForestFire graph, the observed rank deviation is much less severe as compared to $G_{n,p}$.

In addition, we observe that the ranking deviations (in most of the presented cases) increase when we consider a small fraction of the most important vertices. This indicates that most of the low rank vertices in the target graph do not flip their ranks with the more important ones in the crawl, leading to a lower ratio of discordant pairs to the overall total number of pairs. On the other hand, crucial to most applications are the ranking deviations of the *high* PAGERANK vertices, thus making it essential to monitor them.

Finally, we observe that ranking deviation in the Web graphs shown in Fig. 3 are interestingly similar to the random graphs in Fig. 4 and less so with other generative models like ForestFire or ScaleFree graphs. This, we believe, has strong implications in explaining the structure of Web graphs.

Effectiveness of Effectiveness of HAK. We first discuss about the general applicability of the HAK measure and then argue about the supporting experimental evidence reported in Figs. 3 and 4. We recall that the main assumption behind the construction of HAK is that each of the unseen or ghost vertices from the target graph would exert the same fraction of impact (on average) to the crawled set as the actual vertices in the crawl (cp. Sect. 4). We ensure this by constructing the target graph such that each of its vertex has the same fraction of crawled neighbors as the crawled vertices (computed by fidelity). This assumption would not be followed by target graphs, which for example are *DAG-like*, because the ghost vertices there might not have edges back into the crawl. We remark that HAK cannot identify structures in target graph which are not similar to the crawl, yet leading to severe ranking changes in the crawl. For instance, consider a very small crawl with a very high fidelity and low impact. In such a case HAK would always estimate a very low ranking deviation. It could in the worst happen that there exist a few ghost vertices in the target graph with very high PAGERANK, having outgoing edges to only the low rank vertices in the crawl. Our results in Figs. 3 and 4, on the other hand, support the effectiveness of HAK in most of the studied graphs and therefore also validate our assumptions behind HAK.

We first discuss our findings on synthetic graphs. HAK performs fairly well for $G_{n,p}$, for instance with the BFS crawl strategy with 50% block fraction, we record an absolute error of 0.02 (actual: 0.24, estimated: 0.26) for rank correlation of top 30% vertices. The little ranking deviations in ScaleFree and ForestFire can be attributed to the small crawls with high fidelity ($\gamma \in [0.93, 1.0]$). As already discussed, HAK in these cases would always result in a high value, which also explains HAK adapting to the trends. However, we observe a larger deviation for BFS crawls in ScaleFree graphs. Here, HAK underestimates the ranking deviation, which might reflect the existence of the worst case resulting in

a similar estimation as the one described above for very small crawls. However, HAK overestimates the deviation in FFBacklink (see the last 3 plots shown in Fig. 4). We attribute this to the fact that the average impact of the crawl increases in presence of *backlinks* (cp. Sect. 3), which is an overestimation of the actual impact since *Forest Fire* is nevertheless the dominant topology in this graph.

We report more promising results in case of real-world graphs (s. Fig. 3). For instance, for the UK graph we report an almost precise estimation (actual: 0.58, estimated: 0.61). The observed trend in UK is more similar to that seen in $G_{n,p}$ and FFBacklinks, which might also suggest existence of more *backlinks* in this graph, leading to large cycles (cp. Fig. 2). In contrast, the deviation in Friendster is less strong and slightly overestimated by HAK (actual: 0.76, estimated: 0.66) similar to ForestFire.

6 Conclusion

In this paper, we focused on the problem of PAGERANK deviations in Web graphs, typically caused by incomplete crawling. We established that deviations in ranking indeed do occur and can be drastic, as shown in our GOV graph where the correlation among the rankings is only 0.55, measured by Kendall's Tau. To this effect, we proposed the HAK measure, which can reliably estimate such deviations purely on the crawl without any knowledge of the original graph. Our results suggest that incomplete Web graphs behave surprisingly similar to random graph models and quite different from other generative Web models, such as Forest Fire, in terms of PAGERANK deviations. Thus, this study on incompleteness in Web graphs could be important in studying the structure of the Web as well.

Acknowledgements. This work is partially funded by ALEXANDRIA (ERC 339233).

References

1. Ainsworth, S.G., Alsum, A., SalahEldeen, H., Weigle, M.C., Nelson, M.L.: How much of the web is archived? In: Proceeding ACM/IEEE- JCDL 2011
2. Archiveteam. Friendster Social Network Dataset: Friends, : published under, vol. CC0, p. 1.0. Universal (2011)
3. Boldi, P., Santini, M., Vigna, S.: Do your worst to make the best: paradoxical effects in pagerank incremental computations. In: WAW (2004)
4. Boldi, P., Vigna, S.: The WebGraph framework I: Compression techniques. In: Proceedings of the Thirteenth International World Wide Web Conference (WWW 2004), pp. 595–601. ACM Press, USA (2004)
5. Bollobás, B., Borgs, C., Chayes, J., Riordan, O.: Directed scale-free graphs. In: Proceedings of ACM-SIAM Symposium on Discrete Algorithms, SODA 2003 (2003)
6. Costa, M., Gomes, D., Silva, M.J.: The evolution of web archiving. Int. J. Digit. Libr. **18**(3), 191–205 (2016)

7. Dasgupta, A., Kumar, R., Sarlos, T.: On estimating the average degree. In: Proceedings of conference on World wide web, pp. 795–806. ACM (2014)
8. Erdős, P., Rényi, A.: On random graphs. *Publ. Math. Debr.* **6**, 290–297 (1959)
9. Gilbert, E.N.: Random graphs. *Ann. Math. Stat.* **30**(4), 1141–1144 (1959)
10. Hagberg, A.A., Schult, D.A., Swart, P.J.: Exploring network structure, dynamics, and function using NetworkX. In: SciPy2008 (2008)
11. Haveliwala, T.H.: Topic-sensitive pagerank. In: Proceedings of the 11th international conference on WorldWide Web, pp. 517–526. ACM (2002)
12. Holzmann, H., Nejdl, W., Anand, A.: Exploring web archives through temporal anchor texts. In: Proceedings of ACM Web Science Conference - WebSci 2017 (2017)
13. Holzmann, H., Nejdl, W., Anand, A.: The dawn of today’s popular domains: a study of the archived german web over 18 years. In: Digital Libraries (JCDL) (2016)
14. Hübler, C., Kriegel, H.-P., Borgwardt, K., Ghahramani, Z.: Metropolis algorithms for representative subgraph sampling. In: Eighth IEEE International Conference on Data Mining, 2008. ICDM 2008, pp. 283–292. IEEE (2008)
15. Huurdeman, H.C., Ben-David, A., Kamps, J., Samar, T., de Vries, A.P.: Finding pages on the unarchived web. In: IEEE/ACM JCDL (2014)
16. Kendall, Maurice G.: A new measure of rank correlation. *Biometrika* **30**(1/2), 81–93 (1938)
17. Kleinberg, J.M.: Authoritative sources in a hyperlinked environment
18. Leskovec, J., Kleinberg, J., Faloutsos, C.: Graph evolution: densification and shrinking diameters. *ACM Trans. Knowl. Discov. Data* **1**(1) (2007)
19. Li, R.-H., Yu, J.X., Qin, L., Mao, R., Jin, T.: On random walk based graph sampling. In: Data Engineering (ICDE), 2015 (2015)
20. Maiya, A.S., Berger-Wolf, T.Y.: Benefits of bias: towards better characterization of networksampling. In: Proceedings of the 17th ACM SIGKDD International Conferenceon Knowledge Discovery and Data Mining, pp. 105–113. ACM (2011)
21. Ng, A.Y., Zheng, A.X., Jordan, M.I.: Link analysis, eigenvectors and stability. In: Proceedings of International Joint Conference on Artificial Intelligence (2001)
22. Lawrence, P., Brin, S., Motwani, R., Winograd, T.: The pagerank citation ranking: Bringing order to the web. Technical report, Stanford InfoLab (1999)
23. Smith, J.A., Moody, J.: Structural effects of network sampling coverage I: Nodes missing at random. In: Social Networks, vol. 35, pp. 652–668. Elsevier, Amsterdam (2013)
24. Smith, J.A., Moody, J., Morgan, J.H.: Network sampling coverage II: the effect of non-random missing data on network measurement. In: Social Networks, vol. 48, pp. 78–99. Elsevier, Amsterdam (2017)
25. The Internet Archive. The Internet Archive, 1996–2017
26. Vattani, A., Chakrabarti, D., Gurevich, M.: Preserving personalized pagerank in subgraphs. In: Proceedings of ICML (2011)
27. Wang, D.J., Shi, X., McFarland, D.A., Leskovec, J.: Measurement error in network data: a re-classification. In: Social Networks, vol. 34, pp. 396–409. Elsevier, Amsterdam (2012)
28. Wang, T., Chen, Y., Zhang, Z., Sun, P., Deng, B., Li, X.: Unbiased sampling in directed social graph. In: ACM SIGCOMM Computer Communication Review, vol. 40, pp. 401–402. ACM (2010)
29. Zhou, Z., Zhang, N., Gong, Z., Das, G.: Faster random walks by rewiring online social networks on-the-fly. *ACM Trans. Database Syst. (TODS)* **40**(4), 1–36 (2016)



Centrality Maps for Moving Nodes

Clément Bertier^{1,2(✉)}, Farid Benbadis², Marcelo Dias de Amorim¹, and Vania Conan²

¹ LIP6/CNRS – Sorbonne Université, Paris, France

{clement.bertier,marcelo.amorim}@lip6.fr

² Thales SIX GTS, Gennevilliers, France

{farid.benbadis,vania.conan}@thalesgroup.com

Abstract. In dynamic networks, topology changes require frequent updates of centrality measures. Such perpetual calculations range from computationally hungry to unfeasible before the next topological change happens. On top of this, the centrality may seem unstable or even random from the nodal perspective, making them difficult to predict. In this paper, we propose to shift the focus of centrality estimation from the conventional topological perspective to a geographical perspective. We advocate that some centrality metrics are, depending on the situation, inherently related to the geographic locations of the nodes. Our strategy associates a measure of centrality to coordinates and, consequently, to the nodes that occupy those positions. In a vehicular scenario, the one we consider in this paper, geographical centrality is much more stable than node centrality. Hence, nodes do not have to compute their centralities continuously – it is enough to refer to their geographic coordinates and find the centrality by merely consulting a pre-established table. We evaluate our strategy over two large-scale vehicular datasets and show that, whenever we match a centrality to an area, we correctly estimate the centralities up to 80% of the highest-valued nodes.

Keywords: Geographical · Centrality · Maps

1 Introduction

Depending on the dynamics of a graph, maintaining up-to-date centrality values at all times might be unfeasible. For this reason, some authors have proposed approximations to a number of centrality metrics [1, 7]. Unfortunately, non-negligible calculation time is still required, and these approximations may not fit some scenarios. A prime example is a large-scale, highly mobile vehicle-to-vehicle contact graph. We adopt a radically different approach to estimate the importance of a node in dynamic topologies. We leverage the fact that, in many real-world scenarios, the *structure of the contact graph is shaped by the constraints of the underlying geographic area*. Thus, by construction, nodes occupying some areas are likely to have higher centrality than nodes occupying

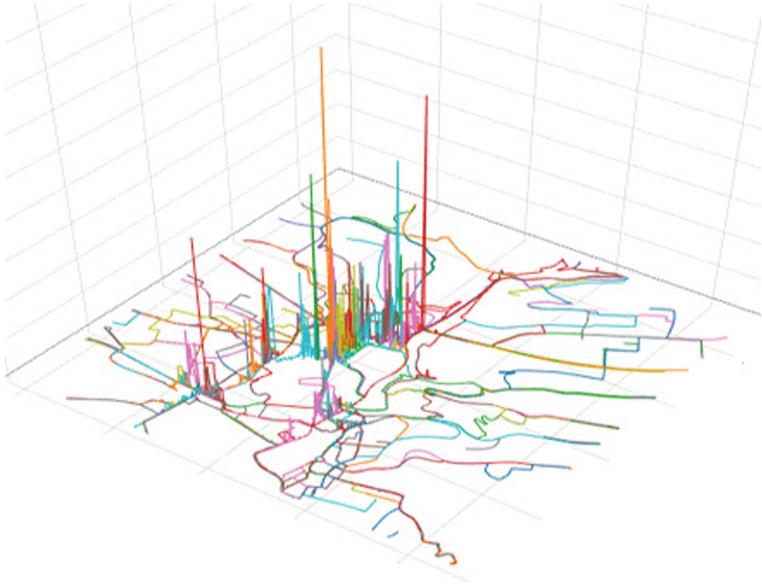


Fig. 1. Betweenness centrality value of a 100 randomly-chosen cars in Luxembourg based on a V2V contact graph. Each color is a different car.

other areas. We advocate that *knowing the position of a node can suffice to know whether a node is central or not*.

Our intuition comes from Fig. 1, where we plot the paths of 100 randomly chosen cars (each car is traced in a different color) in downtown Luxembourg during the morning rush hour (from 6 a.m. to 11 a.m.). In the z-axis, we show their betweenness centrality according to the contact graph formed by vehicles. Note that in the city center (center of the figure) there are surges of betweenness centrality. This central area, holding very high betweenness centrality, corresponds in reality to a very popular boulevard connecting several busy streets. In other words, vehicles in this area are likely to have a high betweenness centrality, whereas cars in the outskirts of the city tend to have a low betweenness centrality. To the best of our knowledge, and in spite of this intuitive observation, no other work uses the geographic positions of nodes as a way to estimate their centrality.

Our idea in this paper involves a two-step process. We first observe the contact graph and compute the nodes' centralities during a learning period. We associate these centralities to the geographic positions where they were measured. We obtain in this way a *centrality map* of the target region which is hopefully stable over time. As we will see later on, this is the case for a significant part of the map, mainly where the centrality is very high or very low (which are centralities of particular interest). Nodes evolving in the target area

do not have to compute their centrality at all times – it suffices to refer to the centrality map to know how central they are.

As a summary, the contributions of this paper are:

- **Centrality maps.** We present a method to create a geographical centrality, by dividing the surface into squares, and utilizing the centrality of nodes located inside the squares as a way to estimate the square's centrality.
- **Application to vehicular networks.** We evaluate our method in two large-scale vehicular networks. We compare the current geographical centrality of a square to the upcoming centrality of nodes in the same square. We observe that these predictions work well on nodes with the lowest and the highest centrality values, often regardless of the type of centrality metric.

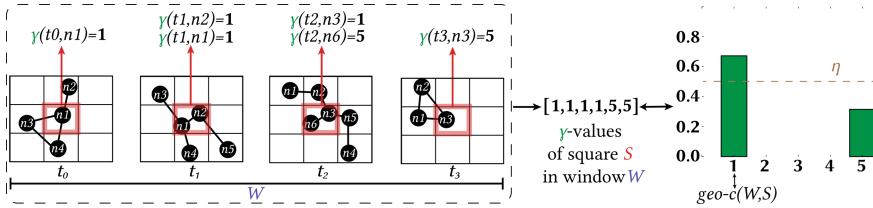


Fig. 2. General methodology to create a dynamic geographical centrality. Here we have 4 different snapshots with different topologies (inside the dashed frame, on the left), and we focus on a single square S (highlighted in red). By analyzing the centrality values of nodes (function γ), we can establish a geographical centrality for S ($geo\text{-}c$).

2 Centrality Maps

Let us rely on Fig. 2 to explain our method to establish a geographical centrality, noted $geo\text{-}c$. Inside the dashed frame (on the left-side of the figure), we show the contact graph at four consecutive time slots. We divide the space in squares of equal size and compute the $geo\text{-}c$ for each square.

We compute the centrality of a square over multiple snapshots by using a sliding window of size W . Consider for example square S in the figure (highlighted in red). We calculate first an instantaneous centrality value, noted as γ , for all nodes and bind these values to the squares they occupy. For instance, at t_0 , node n_1 occupies square S with a $\gamma = 1$, at t_1 , n_1 and n_2 both have a value of 1, and so on.

Once we calculate γ for all nodes inside S during W , we obtain the histogram of γ for S (right-hand side of Fig. 2). Finally, we deduce the $geo\text{-}c$ of S based on the γ with the highest probability.

2.1 Calculation of γ

Our approach is centrality-agnostic (but, for the sake of illustration, we consider the betweenness centrality in Fig. 2). We will evaluate different centrality metrics in the evaluation section.

In dynamic graphs, the notion of high or low values quickly become blurred because of the constant topological changes and fluctuating number of nodes. This makes absolute values of centrality difficult to interpret [6]. A better approach is to rely on the relative centrality of the nodes by ranking them in decreasing order of centralities. Nevertheless, the ranking is still too unstable. In this paper, we make a step forward and consider quantiles of ranks. We refer to this rank as the $\gamma(t, n)$. The steps are as such:

1. We compute the centrality of all nodes in a square at a snapshot (e.g., in Fig. 2, nodes n_1, n_2, n_3 , and n_4 at t_0).
2. We rank all nodes based on their centralities. In Fig. 2, we note that node n_1 at time t_0 has the highest centrality (rank 1) for the betweenness centrality. Two nodes with the same centrality value have the same rank.
3. We segment the ranks in N quantiles, with each quantile holding $\frac{100}{N}$ percent of the ranks: $\gamma : X \mapsto [1, N]$.

Without loss of generality, we fix $N = 5$. Hence, we have $\gamma : X \mapsto [1, 5]$, where the first quantile corresponds to the top 20% of the ranks, the second quantile corresponds to the range 20–40%, and so on. In Fig. 2, if we look at snapshot t_2 , we can see that the γ of nodes n_3 and n_6 are respectively equal to 1 and 5, meaning that n_3 is among the top 20% values and n_6 is among the lowest 20% values.

2.2 Distribution of γ

Now that we have the γ (quantiles) of the nodes within square S in the window, we aggregate these values to obtain the centrality of the square. In Fig. 2, the γ of the nodes inside of S are $[1, 1, 1, 1, 5, 5]$.

The next step is to determine the probability distribution of these values, as represented on the right side of Fig. 2. We note the probability density of a quantile q in square S during window W as $\delta_q(W, S)$; in our example, we have $\delta_1(W, S) = \frac{2}{3}$, $\delta_2(W, S) = 0$, and $\delta_5(W, S) = \frac{1}{3}$, and so on.

We aim to find squares with a representative quantile, i.e., a quantile with much higher probability than the others. To this end, we define a *stability threshold*; a square is considered as *stable* if one of its quantiles above η .

The *geo-c* of a square is defined only if the requirements below are met, otherwise we consider its *geo-c* to be *undefined*:

$$\text{geo-}c(W, S) = q \leftrightarrow \exists!q, \delta_q(W, S) > \eta. \quad (1)$$

3 Applying Centrality Maps to Vehicular Networks

The general idea is that we can guess the present centrality of a node by looking at past $geo\text{-}c$ of the square it falls in. For all time steps t spaced by τ :

1. We compute the geographic centralities $geo\text{-}c(S, W_t)$ of all squares S of the map on the sliding window $W_t =]t - W, t]$.
2. We compute the γ of all nodes at the next time slot $t + \tau$.
3. For each node n within a square S , we compare the true centrality quantile of the nodes $\gamma(t + \tau, n)$ to the one predicted by its geographic position $geo\text{-}c(W, S)$ when it exists. If both values are equal we count the prediction as a success.

3.1 Experimental Datasets

Our evaluation makes use of two vehicular datasets:

- LuST [2] covers 172 km² over the city of Luxembourg, with approximately 300,000 vehicles.
- TapasCologne [9] covers 400 km² over the city of Cologne, with approximately 700,000 vehicles.

For both datasets, we focus on the city center, a surface of approximately 20 km² in both cases. We calculate the centrality of nodes (vehicles) of the contact graphs. Two vehicles form a link if they are within a communication radius Δ of each other. In this work, we take $\Delta = 50$ m. It corresponds to a relatively short distance in the context of vehicular networks, which is a reasonable assumption given our high density urban environments.

These two vehicular datasets depart from one another in several aspects. The road layout of Luxembourg is more star-shaped than Cologne's, whereas Cologne has a shorter and more crowded rush-hour. In terms of graph dynamics, the Luxembourg dataset forms large connected components in the same areas throughout the day, whereas in Cologne they change over the time of day.

We fix the following parameters: window $W = 60$ s, time step $\tau = 10$ s, and stability threshold $\eta = 0.5$.

3.2 Node Centralities

There are numerous centrality metrics in the literature [5], but here we only focus on the most well-known in a networking context.

Degree centrality counts the number of direct links of a node. It provides a value of the local importance of the node in the graph.

Closeness centrality represents the average distance between a node and the rest of the graph. It needs to be computed over the entire connected component.

Noting $d(v, u)$ as the distance (in terms of number of hops) between nodes v and u :

$$clos(v) = \sum_u \frac{n-1}{N-1} \frac{n-1}{d(u, v)}. \quad (2)$$

The $\frac{n-1}{N-1}$ factor aims to weight the metric according to the proportion of reachable nodes in the connected component of v , with n being the size of the connected component and N the total population.

Betweenness centrality gives the proportion of shortest-paths that go through node v . Noting σ_{sd} as the number of shortest-path between a source s and a destination d , and $\sigma_{sd}(v)$ as the number of these shortest-paths that go through node v :

$$bet(v) = \sum_{s \neq v} \sum_{d \neq s \neq v} \frac{\sigma_{sd}(v)}{\sigma_{sd}}. \quad (3)$$

Egobetweenness centrality is a local approximation for the betweenness centrality [4]. It follows the same formula as the betweenness of a node, but limited only to its 1-hop neighbors.

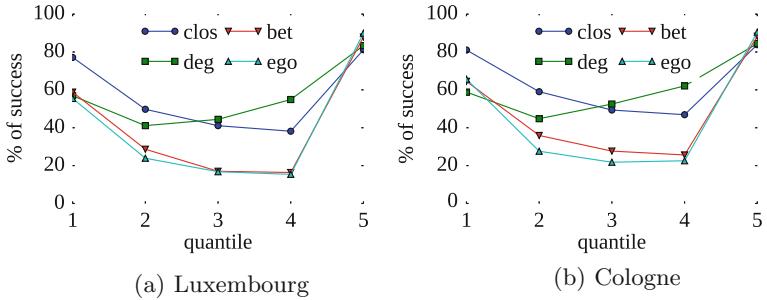


Fig. 3. Success rate of node centrality prediction per quantile for degree (deg), closeness (clos), betweenness (bet) and ego-betweenness (ego) centralities.

4 Results

Figure 3 presents the success rate of predicting the nodes centralities from their positions: γ of the node is compared to the $geo\text{-}c$ value of the square it falls in (see Sect. 3), and results are given per quintile. Evaluations were carried out for both Luxembourg and Cologne datasets over the entire day-time period (from 5.30 am to 10 pm).

The overall pattern for both cities is strikingly similar. Prediction rates are best for the first and fifth quantiles: if the node is within the top 20% values

or lowest 20% values, its node centrality is correctly provided from its position. This is true with 82 to 90% chances if it lies in the fifth quantile and between 60 to 80% chances if it lies in the first quantile.

One other observation is that, for all centralities, prediction success is best for quantiles 1 and 5, but much lower on quantiles 2–3–4, in particular for betweenness and ego-betweenness which are the only ones with prediction success rate lower than 50%. This is due to the great variability in centrality values which often switch from one intermediate quantile to another.

4.1 Contact Density

To understand better the success rate of each centrality, in Fig. 4 we provide number of predictions per quantile (in millions of prediction attempts), for all centralities, along with the proportion of successes (light green) and failures (dark red).

Keep in mind that the number of nodes in each quantile may vary depending on the centrality metric. Indeed different centralities yield different behaviors in terms of value distribution. This is a pure graph property; for instance, the betweenness is known to have few very high valued nodes because of its flow-based nature, but nodes located even one hop away from the highest betweenness-valued node may have a value of zero. In Fig. 4b, around 60% of predictions of betweenness concern nodes of the fifth quantile.

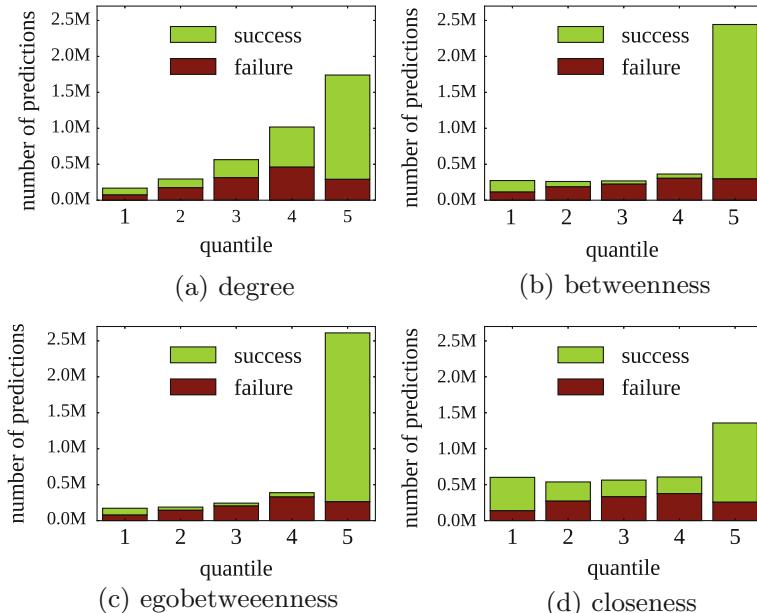


Fig. 4. Number of node centrality predictions per quantile for the four centralities.

Since we only take in account squares which have a defined $geo\text{-}c$; depending on the centrality, the same square at the same time may or may not have a defined $geo\text{-}c$ value. This explains why the four centralities have different behaviors: For the betweenness (and ego-betweenness) most values are found in the 5th quantile, where nodes have low betweenness centrality values. Very few nodes have higher values (quantiles 1 to 4).

However, all the centralities reveal a similar pattern: the 5th quantile holds the largest number of predictable nodes (i.e., cars located in a square with a defined $geo\text{-}c$ value) and this 5th quantile also yields the best prediction results (between 80-90% successes on all four centralities).

These squares with low $geo\text{-}c$ therefore corresponds to areas where cars have limited contact opportunities. The takeaway here is that we can easily predict low centrality values of nodes based on their positions, hence predicting a large proportion of the population.

4.2 Evolution Over Time

In Fig. 5, we show the prediction success ratio evolution over time, during a day-long period. We plot the results for the first quantile (i.e., nodes with centrality values among the top 20%). For all four centralities, the plot shows a pattern corresponding to three periods: the morning rush (from 6 a.m. to 11 a.m.), the noon break (12 a.m. to 3 p.m.), and the evening rush (4 p.m. to 8 p.m.).

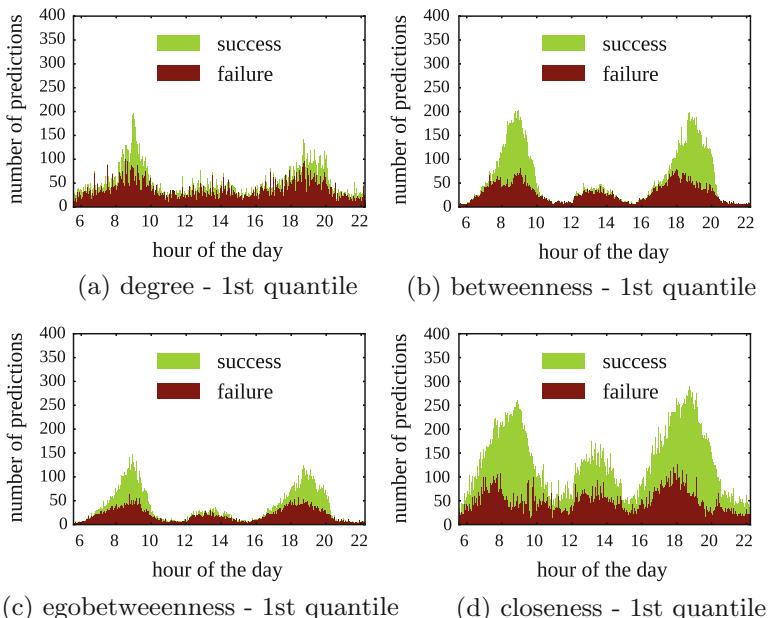


Fig. 5. Evolution of node centrality prediction (success and failure) over time, for the four centralities and the first quantile.

The degree centrality (Fig. 5a) shows a dented pattern. This means that success/failure patterns are not strongly time-correlated in the short term. Predicting degree centrality is very uneasy: there are no large periods of time where success outnumbers failures (but the total number of successes is still larger than failures).

For betweenness centrality (similarly for ego-betweenness) (Fig. 5b, c) better prediction is achieved for the rush hours, both in the morning and in the evening. At noon, it is difficult to obtain good predictions.

The closeness centrality (Fig. 5d) leads to better and more regular prediction successes – it is significantly less sensitive to population density as a majority of the time, no matter the population or hour of the day, there are more successes than failures. Observing these centralities over time helps shedding light on their behaviors. They respond differently to population density, with the closeness being the less sensitive, as well as providing excellent prediction for the morning and evening rush-hour periods.

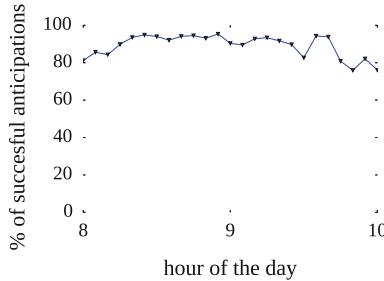


Fig. 6. Successful predictions every 5 min during the morning rush-hour.

On Fig. 6 we focus on the morning peak of the rush hour period; we plot the success rate of the closeness centrality prediction every 5 min from 8 a.m. to 10 a.m. Prediction in this case is very successful: there is at least 80% chance of successfully predicting the correct quantile, with a maximum of approximately 95% of nodes' quantiles correctly predicted. Even if centralities are sensitive to population density, as we have seen above, when we focus only on very dense scenarios we can ensure a very high prediction success rates with our methodology. Such results *strongly encourage that creating a database matching a centrality to a particular time of day is feasible*.

5 Related Work

To the best of our knowledge, very little work has been done on the position of nodes inferring the centrality. For instance, the centrality of streets have been established [3,8], yet they exclusively based their approach on the topography, i.e., “is this street central to the road layout”. There were several efforts to

leverage the geographical (or topographical) position of cars to better disseminate content in vehicular networks. In SADV [12], the dissemination method is based on the awareness of intersections; if no node is found within communication range, and if the current holder of the information is in an intersection, the data is transferred to a static-node assumed to be existing in the middle of the intersection. Later, the node will forward the data to vehicles passing in the intersection. A similar idea is found in Throwboxes [14], where the authors state that knowing contact opportunity is necessary. In DP-IB [13], a data-center tries to disseminate information (accident, traffic-jam) to vehicles within a pre-defined dissemination zone. The data is broadcasted on specific roads, knowing they intersect other roads of the area. To avoid using the data-center too often, they propose to use a *relay and broadcast station*, located at intersections, whose purpose is to copy the information and forward it to other nodes. In all the examples above, it is assumed that intersections are the areas where the most opportunistic forwarding should happen, given the density and transient nature of the vehicles found inside of them. Additionally, numerous VANET algorithms make use of the topography to disseminate the content [10, 11].

6 Conclusion and Future Work

In this paper, we explored the correlation between the centrality of nodes and their geographical position. To do so, we defined a geographical centrality of small areas (squares), based on the centrality of nodes located inside of them. We evaluated this idea over two vehicular datasets, by comparing an estimation of the present centrality of nodes to the passed geographical centrality. Our evaluation showed that up to 80% of the 20% highest and 90% of the 20% lowest centrality values can be successfully estimated, based on their position. By investigating the success rate based on the time of day, we saw that the success rates of different centralities strongly variate according to population. Our finding is the higher the population, the more reliable the centrality estimation can be done from the position, for the highest centrality nodes. Some centralities are less sensitive to this, notably the closeness which yields good results regardless of population or time.

Our future work consists of turning the ideas presented here into a pragmatical solution for position-based centrality estimation. We will first study in greater detail the failures in the predictions by analyzing how far-off the prediction was from the actual centrality, to achieve greater success rate. Then, we aim to implement a database containing the list of areas with their associated centralities according to time of day. It would require to pre-process mobility traces to calculate the centralities beforehand, but once such calculation is done by leveraging the day-to-day redundancy in traffic patterns we can easily estimate the centrality of a user at the sole cost of querying the database, independently from the user's identity or centrality complexity.

References

1. Bader, D.A., Kintali, S., Madduri, K., Mihail, M.: Approximating Betweenness Centrality. In: Algorithms and Models for the Web-Graph, pp. 124–137. Springer, Heidelberg. https://doi.org/10.1007/978-3-540-77004-6_10. http://link.springer.com/10.1007/978-3-540-77004-6_10
2. Codeca, L., Frank, R., Engel, T.: Luxembourg SUMO Traffic (LuST) Scenario: 24 hours of mobility for vehicular networking research. In: 2015 IEEE Vehicular Networking Conference (VNC), vol. 2016-Janua, pp. 1–8. IEEE (2015). <https://doi.org/10.1109/VNC.2015.7385539>. <http://ieeexplore.ieee.org/document/7385539/>
3. Crucitti, P., Latora, V., Porta, S.: Centrality measures in spatial networks of urban streets. Phys. Rev. E-Stat., Nonlinear, Soft Matter Phys. **73**(3), 1–5 (2006). <https://doi.org/10.1103/PhysRevE.73.036125>
4. Daly, E.M., Haahr, M.: Social network analysis for routing in disconnected delay-tolerant MANETs. In: Proceedings of the 8th ACM International Symposium on Mobile Ad Hoc Networking and Computing - MobiHoc 2007, p. 32. ACM Press, New York (2007). <https://doi.org/10.1145/1288107.1288113>. <http://dl.acm.org/citation.cfm?id=1288107.1288113>
5. Freeman, L.C.: A set of measures of centrality based on betweenness. Sociometry **40**(1), 35 (1977). <https://doi.org/10.2307/3033543>. <http://www.jstor.org/stable/3033543?origin=crossref>
6. Kim, H., Anderson, R.: Temporal node centrality in complex networks. Phys. Rev. E **85**(2), 026,107 (2012). <https://doi.org/10.1103/PhysRevE.85.026107>
7. Riondato, M., Kornaropoulos, E.M.: Fast approximation of betweenness centrality through sampling. Data Min. Knowl. Discov. **30**(2), 438–475 (2016). <https://doi.org/10.1007/s10618-015-0423-0>. <https://pdfs.semanticscholar.org/213c/7f8e9a8a046020ebfa2a7aaa194b3c05a87a.pdf>. <http://link.springer.com/10.1007/s10618-015-0423-0>
8. Strano, E., et al.: Street Centrality Versus Commerce and Service Locations in Cities: a Kernel Density Correlation Case Study in Bologna, Italy pp. 1–14 (2007). <http://arxiv.org/abs/physics/0701111>
9. Uppoor, S., Trullols-Cruces, O., Fiore, M., Barcelo-Ordinas, J.M.: Generation and analysis of a large-scale urban vehicular mobility dataset. IEEE Trans. Mob. Comput. **13**(5), 1061–1075 (2014). <https://doi.org/10.1109/TMC.2013.27>. <http://ieeexplore.ieee.org/document/6468040/>
10. Wang, W., Xie, F., Chatterjee, M.: Small-scale and large-scale routing in vehicular ad hoc networks. IEEE Trans. Veh. Technol. **58**(9), 5200–5213 (2009). <https://doi.org/10.1109/TVT.2009.2025652>
11. Wu, H., Fujimoto, R., Guensler, R., Hunter, M.: MDDV. In: Proceedings of the First ACM Workshop on Vehicular Ad Hoc Networks - VANET 2004, p. 47. ACM Press, New York (2004). <https://doi.org/10.1145/1023875.1023884>. <http://dl.acm.org/citation.cfm?id=1023875.1023884>. <http://portal.acm.org/citation.cfm?doid=1023875.1023884>
12. Ding, Y., Xiao, L.: SADV: Static-Node-Assisted Adaptive Data Dissemination in Vehicular Networks. IEEE Trans. Veh. Technol. **59**(5), 2445–2455 (2010). <https://doi.org/10.1109/TVT.2010.2045234>. <http://ieeexplore.ieee.org/document/5431020/>
13. Zhao, J., Zhang, Y., Cao, G.: Data pouring and buffering on the road: A new data dissemination paradigm for vehicular ad hoc networks. IEEE Trans. Veh. Technol. **56**(6 I), 3266–3277 (2007). <https://doi.org/10.1109/TVT.2007.906412>. <http://ieeexplore.ieee.org/document/4358489/>

14. Zhao, W., Chen, Y., Ammar, M., Corner, M., Levine, B., Zegura, E.: Capacity enhancement using throwboxes in DTNs. In: 2006 IEEE International Conference on Mobile Ad Hoc and Sensor Systems, pp. 31–40. IEEE (2006). <https://doi.org/10.1109/MOBHOC.2006.278570>. <http://ieeexplore.ieee.org/document/4053916/>



Core Stratification of Two-Mode Networks

Henry Soldano^{1,2(✉)}, Sophie Bary², Guillaume Santini¹, and Dominique Bouthinon¹

¹ Université Paris 13, Sorbonne Paris Cité, LIPN, UMR 7030, Villetaneuse, France

² Muséum National d'Histoire Naturelle, ISYEB, UMR 7205, Paris, France

henry.soldano@lipn.univ-paris13.fr, sophie.bary@mnhn.fr

Abstract. We propose and experiment a new edge decomposition scheme called stratification. It relies on the notion of core subgraphs and may be used with various core definitions, including two-parameters core definitions when exploring two-mode networks. In the two-parameters case the core subgraphs are partially ordered and stratification relies then in finding a sequence of (totally) ordered subgraphs. The main purpose is to help visualizing a large graph by first extracting its densest part, then removing the corresponding edges and then look again at the densest part of the remaining graph and so on. We present experiments on stratification of a two-mode epistemological network.

Keywords: Edge decomposition · Generalized cores · Two-mode networks

1 Introduction

For some years there is a lot of interest in investigating complex networks, as regulatory networks or co-authoring networks [10]. In particular, many methods have been developed to extract from such networks cohesive subnetworks called communities in which vertices are highly connected to each other and which have few links to other communities [6]. Another way to investigate complex networks consists in extracting a *core subgraph* from the network. First, the k -core definition was proposed in [11] and requires the core subgraph to be the largest subgraph whose nodes all have degree at least k . It was further extended to a class of so-called generalized cores [3] in which nodes have to satisfy other topological constraints. Other core definitions were recently proposed to investigate directed networks [7, 13] and two-mode networks [5]. Core definitions commonly depends on one or several parameters in such a way that, given a core definition, the core subgraphs may be ordered according to the parameter. Obviously, the $(k+1)$ -core subgraph of some graph G is included in the k -core subgraph of G : as all nodes in the $k+1$ core subgraph have degree at least $k+1$ then they have also degree at least k . This ordering led in the k -core case to the k -core (or k -shell) decomposition of a graph [11]: the vertex set V is partitioned

into a sequence $V_\delta, V_{\delta-1}, \dots, V_0$. Here δ stands for the *graph degeneracy*, i.e. the highest k such that that the k -core is not empty. Then, for $0 \leq k < \delta$, V_k contains the vertices that belong to the k -core but not to the $(k+1)$ -core. Let us denote the core subgraph associated to V_k by $G_k = (V_k, E_k)$. The same core subgraph sequence also leads to the edge decomposition $E_\delta, E_{\delta-1} \setminus E_\delta, \dots, E_2 \setminus E_1, E_1$ which is associated to a soft clustering of the vertex set: each edge only belongs to one such edge shell $E_k \setminus E_{k+1}$ but a vertex may belong to several shells. In [1] it was shown that a simple hierarchical clustering of edges, that detects *link communities*, was efficient in revealing functional subgraphs in various biological networks.

We propose a new edge decomposition, called *stratification*. Stratification is not intended to discover link communities, but rather to extract maximal subgraphs with decreasing density levels, i.e. decreasing k -values. The key difference with k -core decomposition is that stratification results in edge subsets that satisfy the core property at decreasing density levels.

The original motivation of this work laid on the exploration of a two-mode network, attributes describing the vertices of both modes. We were primarily interested in the network structure, relating in a second step attributes describing the vertices of both modes to structural results. However, the core definition of two-mode network depends on two parameters [5]: cores are only partially ordered and there is then no straightforward way to decompose the network in a sequence of cores. A similar situation was previously encountered and investigated when exploring collaboration in directed networks [7]. The second contribution of this article is then to apply the above mentioned stratification scheme to partition the edge set when using two parameter core definitions.

We first present cores and core subgraphs in Sect. 2. We then introduce stratification Sect. 3, first in the case of totally ordered cores and then in the case of two-parameters core definitions. Section 4 presents our two-mode network together with its exploration using two parameters stratification.

2 Cores

2.1 Preliminaries

Let $G = (E, V)$ be a graph. we define the *edge function* es by $\text{es}(V') = \{(x, y) \in E \mid x, y \in V' \subseteq V\}$, i.e. $E' = \text{es}(V')$ contains the edges that have both their extremities in V' . The subgraph induced by the vertex subset $V' \subseteq V$ is then defined as $(V', \text{es}(V'))$. In the same way, we define the vertex function vs by $\text{vs}(E') = \{x \in V \mid \exists y \in V \text{ s.t } (x, y) \in E'\}$. The subgraph induced by $E' \subseteq E$ is then $(\text{vs}(E'), E')$.

2.2 Graph Cores

The k -core [11] was first defined as the largest vertex subset inducing a subgraph G' in which any vertex x have degree at least k in G' . This was latter generalized

to a wider class of *vertex cores* according to various structural properties. The core subgraph is defined as the unique maximal subgraph whose vertices all satisfy the structural property.

Let $G = (V, E)$ be a graph. Formally, we define *vertex properties* as $P : V \times 2^V \rightarrow \{\text{true}, \text{false}\}$ mappings where $v \in V'$ and $P(v, V')$ truth value depends on the subgraph $G' = (V', \text{es}(V'))$. The largest such V' is then called the *core* of G according to P and G' is the associated *core subgraph*. To define a core, we need P to be such that there does exist such a largest vertex subset V' . This is true whenever P is *monotone* i.e. for any $X_1 \subseteq X$ we have that $P(x, X_1)$ and $X_2 \supseteq X_1$ implies $P(x, X_2)$ [3, 12]. We further refers to cores as P -cores according to the associated structural property P . The core definitions we further discuss are (i) the k -core, as a single parameter core and (ii) the two parameters h - a HA-core for two mode networks:

1. *k -degree core (or k -core) of a single mode network $G = (V, E)$.* It is a vertex core whose vertex property $P(v, V')$ requires that the degree of v in the core subgraph $G' = (V', \text{es}(V'))$ is at least k .
 2. *h - a HA-core of a two-mode network $G = (V_1, V_2, E)$.* In such a two-mode network, the edge set E is made of vertices linking vertices from V_1 to vertices from V_2 . Let $X_1 \subseteq V_1$, $X_2 \subseteq V_2$ and $X = X_1 \cup X_2$, the subnetwork $(X, \text{es}(X))$ relates then vertices from X_1 to vertices from X_2 . The h - a HA-core is defined through the following vertex property P :
 - $P(v, X)$ holds if and only if $v \in X_1$ and degree of v in $(X, \text{es}(X))$ is at least h or $v \in X_2$ and degree of v in $(X, \text{es}(X))$ is at least a .
- The latter core definition (see an example Fig. 1) is a slight reformulation of the (p, q) -core defined by Cerinsek and Batagelj [5].

2.3 Computing Cores

A generic algorithm for computing cores has been given in [3]. **core(\mathbf{X}, \mathbf{E})** computes the P -core of the subgraph induced by a vertex subset X of some graph $G = (V, E)$.

```

core(  $\mathbf{X}, \mathbf{E}$  )
   $C \leftarrow X$ 
  While  $\exists x \in C$  such that  $P(x, C)$  is false do
     $C \leftarrow C \setminus \{x\}$ 
  Return  $C$ 

```

In [3] the authors stressed that the complexity of a vertex core algorithm depends on the cost of evaluating $P(v, V')$. When evaluating $P(v, V')$ only depends on neighbours $N(v, V')$ of vertex v this cost is linear with $|V|$ and the degree of v . We give hereunder an example of a two-mode network together with its 2-2 HA-core.

Example 1. The vertex sets of the two-mode network (V, E) displayed Fig. 1 are $V_1 = 123$ and $V_2 = 456$ with $V = 123456$. The execution of **core(\mathbf{V}, \mathbf{E})** with

$h = a = 2$ starts with $C = V$ and $\text{es}(C) = E$. We have then the following iterations:

- (1) $C = 12456$: $3 \in V_1$ is removed as it has degree $1 < 2 = h$ in $(C, \text{es}(C))$
- (2) $C = 1245$: $6 \in V_2$ is removed as it has degree $1 < 2 = a$ in $(C, \text{es}(C))$.

Note that as 3 is no more in C , $3 - 6$ is no more in $\text{es}(C)$ and the degree of 6 is now only 1 .

The execution then stops as all vertices x of C have degree at least 2 in $(C, \text{es}(C))$.

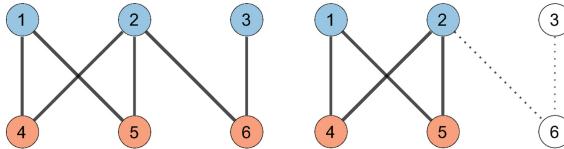


Fig. 1. On the left, the two-mode network G of Example 1 with on the top vertices from V_1 (in blue) and on the bottom vertices from V_2 (in red). On the right, the 2-2 HA-core subgraph of G where vertices outside the core are in white and edges outside of the core subgraph are dashed.

2.4 Ordering Cores

Cores properties can be ordered. When two core properties P and P' are such that for any x and X , we have $P(x, X)$ implies $P'(x, X)$ we write $P \implies P'$ and say that P is stronger than P' . The associated cores are then ordered:

Proposition 1 Let p be the core operator such that for any $X \subseteq V$, $p(X)$ returns the core C of the subgraph $(X, \text{es}(X))$. If two core properties are such that $P \implies P'$, then we have $p(X) \subseteq p'(X)$

As an example, the k -cores as defined in [3], decrease whenever k increases. Regarding the h - a HA two-mode core, the cores are partially ordered according to parameters pairs:

Proposition 2 Let $P_{(h,a)}$ be the vertex property associated to the h - a core definition. Then whenever $h' \geq h$ and $a' \geq a$, $P_{(h',a')}$ is stronger than $P_{(h,a)}$

When a parameter orders the cores by increasing strengths, the *degeneracy* of the graph corresponds to the highest parameter value such that the core is not empty. Let us note P_k the core property with parameter k . A variant **AllCores(X,E)** of the **core(X,E)** algorithm was proposed to compute the cores associated to all parameter values such that the core is not empty [3]: first the nodes that do not satisfy P_1 are removed, thus defining the P_1 -core, then the nodes that do not satisfy P_2 are removed and so on until k is the degeneracy δ .

In the case of two parameters cores ordered as in Proposition 2, a degeneracy may be associated to one parameter by fixing the other parameter value. When considering Example 1 and fixing $h = 2$ we have that the a -degeneracy when $h = 2$ is also 2 as the 2-3 HA core is empty. The variant **AllCores(X,E)** may then be used to compute the a -degeneracy for each value of h . From the computation of all a -degeneracies we may then deduce the *core frontier* i.e. the set of maximal parameter pairs (h, a) resulting in non-empty cores. Core frontiers were first defined in [7].

3 Stratification

3.1 One and Two Parameters Stratification

We first consider a sequence of totally ordered core properties $P_1 \dots P_n$ such that P_k is stronger than P_{k-1} . We suppose that a P_k -core subgraph have no isolated node, and therefore we may simply consider a core subgraph as its edge set E_k . When considering a subgraph $(\text{vs}(E'), E')$ induced by some edge subset E' we will denote by $E_k(E')$ its P_k -core subgraph. The stratification of a graph $G = (V, E)$ is as follows: we start by considering E as our current edge set D from which we extract the strongest non empty core V_δ and define this way the first *strate* $A_\delta = \text{es}(V_\delta) = E_\delta$. Then by removing A_δ from E we obtain a new current edge subset set D and apply then the same scheme to D , extracting the strongest non empty core V_l , stating $A_l = \text{es}(V_l)$, updating D to $D \setminus A_l$ and so on. The process is exemplified below on k-cores:

Example 2 We consider the graph $G = (V, E)$ represented Fig. 2. A k-core subgraph is represented by its edge set E_k . The degeneracy of G is 3. E_3 , the 3-core subgraph of G is the 4-clique induced by 1234. The stratification of G starts with $A_3 = E_3$, then $D = E \setminus A_3$ and A_2 is then the 2-core subgraph $E_2(D \setminus A_3)$, i.e. the triangle {25, 26, 56}. A_1 is then made of the remaining edge {46}.

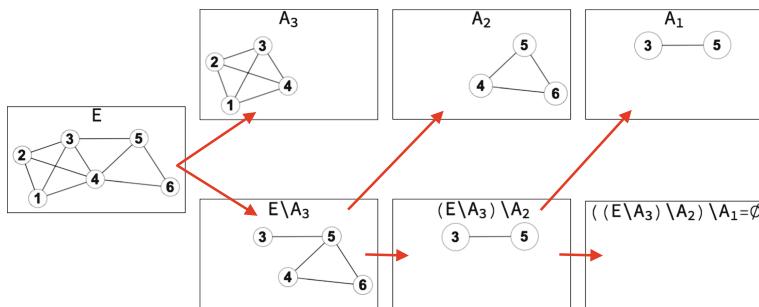


Fig. 2. Stratification of a graph $G = (V, E)$ as a tree. The root G is on the left, the upper elements are the edge subsets A_3, A_2, A_1 , the lower edge subsets represent at each level the remaining subgraph.

We consider now a set of core properties $P_{i,j}$ where we have both $P_{i,j+1}$ and $P_{i+1,j}$ stronger than $P_{i,j}$. When considering a subgraph $(vs(E'), E')$ induced by some edge subset E' we will further denote by $E_{i,j}(E')$ its $P_{i,j}$ -core subgraph. In this two-parameter case, the core properties are only partially ordered while stratification represents a sequence of totally ordered cores. Stratification is then a sequence of edge subsets $\dots A_{h_1,a_1} \dots A_{i_m,j_m}$ where $(i_k, j_k) = (\delta, \delta')$ belongs to the core frontier of the current subgraph D as defined in Sect. 2.4, i.e. (δ, δ') is a maximal pair (i, j) s.t. $E_{i,j}(D) \neq \emptyset$.

The stratification is obtained by applying the following algorithm:

Stratification2(E)

$D \leftarrow E; L \leftarrow \emptyset;$

While $D \neq \emptyset$

 Search for a maximal pair (i, j) s.t. $A_{i,j} \leftarrow E_{i,j}(D) \neq \emptyset$

$L \leftarrow L \cup \{(A_{i,j}, (i, j))\}$

$D \leftarrow D \setminus A_{i,j}$

endWhile

Return L

We need a criterion to choose a pair within the frontier of maximal pairs. In what follows we exemplify the stratification process when choosing a pair (i, j) such that the number of edges in $A_{i,j}$ is maximal.

Example 3 We consider the two-mode network G of Example 1 and the h - a HA-core definition from Sect. 2.2 and also exemplified in Example 1. The frontier of maximal parameter pairs is $\{(2, 2), (3, 1)\}$. For instance, $E_{1,2}$ is non-empty but is included in $E_{2,2}$ while $E_{3,2}$ is empty. $E_{2,2}$ has size 4 while $E_{3,1}$ has size 3 and therefore we select $A_{2,2} = E_{2,2}$. The new current edge set is $D = E \setminus A_{2,2} = \{26, 36\}$. The only maximal pair with non empty core within D is $\{1, 2\}$ and we have $A_{1,2} = \{26, 36\}$ which leads to an empty current edge set D . As a result we have $A = \{A_{2,2}, A_{1,2}\}$. The stratification process is depicted Fig. 3.

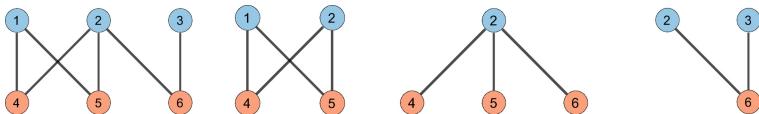


Fig. 3. A two-mode network G and its stratification according to the h - a HA core property. From left to right, the figure represents G , the two maximal core subgraphs $A_{2,2} = E_{2,2}$ and $E_{3,1}$, and $A_{1,2} = E_{1,2}(E \setminus A_{2,2})$.

In our experiments in Sect. 4 we have alternatively maximised the local modularity [9] of a vertex set of a candidate strate $A_{i,j}$ with respect to the current graph D . Local modularity represents the contribution of a given vertex group to the modularity. Contrarily to the number of edges in $A_{i,j}$ the local modularity also refers to external links relating the vertices in $A_{i,j}$ to other vertices in the current subgraph D .

4 A Two-Mode Network of Epistemological Data

We are currently investigating data related to a MNHN-IRD program (first called MUSORSTOM then Tropical Deep-Sea Benthos) of expeditions to explore the deep-sea in the Indo-West Pacific region, since 1976 [2]. The two-mode network investigated here relates a vertex set V_1 of 1480 articles to a vertex set V_2 of 80 campaigns. A directed edge relates an article to a campaign whenever the article mentions the campaign. There are 8043 such edges in this network we refer to as the *ArticleCitesCampaign* network, *ACC* for short. Figure 4 displays the ACC network with a standard force-directed layout [8]. In the periphery of the network we may see a large number of articles that cite very few campaigns together with a dense central subnetwork which displays no clear structure. Each campaign is referred to as ccI where I is some integer.

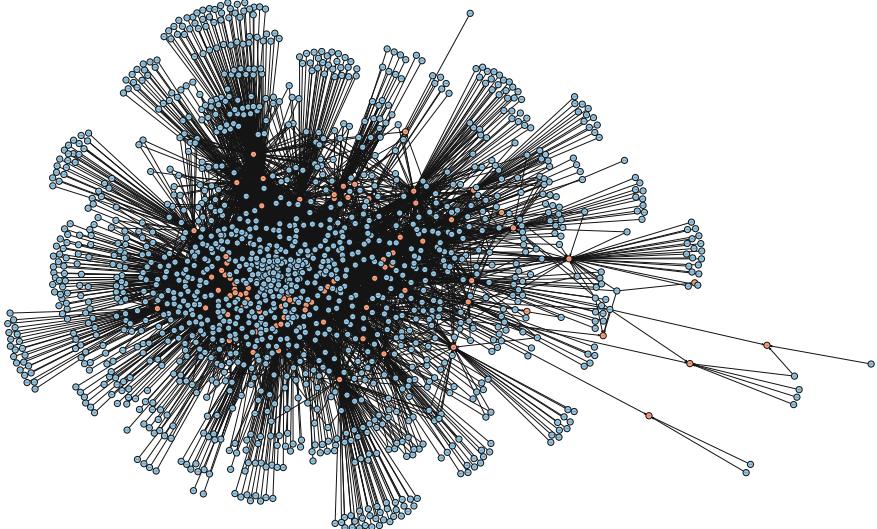


Fig. 4. The two-mode network relating articles (in blue) to campaigns (in red)

As our experiments concern the stratification process, attribute values, when available, are only used to discuss the edge partition we obtain. Regarding the articles, the attribute set contains in particular the following:

- Use of molecular characteristics for taxonomic description or classification purpose (Yes or No)
- Bibliographical categorization
 - Taxonomy with nomenclatural acts as species description and/or taxonomic revision (TA)
 - Taxonomy without nomenclatural purpose (classification) (TS)

- Use of the taxonomy (names and biology of the organism) for inventories and application purposes such as chemistry studies (AT)
- Other fields (e.g. geology, sociology)(Other)

The campaigns have attributes as:

- Geographical region (e.g. New Caledonia, Papua New Guinea,)
- Year
- Objective category with specific purpose as Chemistry, Geology or Halieutic Research or more general purpose as faunistic inventory.

Stratification is applied using h - a HA-cores, with an additional constraint: we exclude from the parameter pairs (h, a) those with $a = 1$ or $h = 1$. For these values the method tends to select stars, i.e. in our case $h = 1$ together with a large value of a , which means that each article in the core cites at least one campaign (a star) cited by at least a articles. However we are more interested in less straightforward structures. Furthermore, to chose among the maximal pairs we experiment two criteria : either maximizing the core subgraph size $|A_{h,a}|$ or maximizing the local modularity of the vertex subset of $A_{h,a}$ with respect to the current graph D . Finally, we limit the stratification depth, i.e we only discuss the first strates $A_{h_1,a_1} \dots A_{h_7,a_7}$. For sake of clarity we refer to A_{h_1,a_1} as $A_1(h_1, a_1)$ or simply A_1 . We first compare in Table 1 the stratification A^N obtained when maximizing the number of edges to the stratification A^M obtained by maximizing the local modularity. The two criteria have different grounds. While the number of edges, i.e. internal links, of the core subgraph $A_{h,a}$ is independent from the current subgraph D , the local modularity of $A_{h,a}$ also depends on the external links to the current subgraph D . Still, the first two steps lead to the same parameter pairs (7, 88) and (4, 52) and we obtain the same subgraphs $A_1^N = A_1^M$ and $A_2^N = A_2^M$ we further refer to as A_1 and A_2 .

Table 1. Stratification with maximization of number of edges (left) or local modularity (right). In col. 3 and 8 ($n_k^1 n_k^2$) stands for the number of articles and campaigns in A_k

k	(h_k, a_k)	(n_k^1, n_k^2)	$ A_{h_k,a_k} $	$ D $	$\text{vs}(D)$	(h_k, a_k)	(n_k^1, n_k^2)	$ A_{h_k,a_k} $	$ D $	$\text{vs}(D)$
Start	(1, 1)	(1480,80)	7673			(1, 1)	(1480,80)	7673		
1	(7, 88)	(212, 20)	2356	5317	1535	(7, 88)	(212, 20)	2356	5317	1535
2	(4, 52)	(229, 20)	1464	3853	1497	(4, 52)	(229, 20)	1464	3853	1497
3	(4,20)	(298, 51)	1655	2198	1345	(3,31)	(46,4)	153	3700	1492
4	(4,2)	(45, 50)	192	2006	1308	(2,44)	(279, 13)	709	2991	1432
5	(2,32)	(32, 2)	64	1942	1297	(2,31)	(174, 12)	438	2553	1394
6	(4,1)	(13, 39)	52	1890	1283	(2,26)	(226, 20)	607	1946	1317
7	(2,20)	(20, 2)	40	1850	1279	(2,16)	(87, 11)	205	1741	1283
Rest	(1,1)	(1200,79)	1850	0	0	(1,1)	(1204, 79)	1741	0	0

Subgraphs A_1 and A_2 are displayed Fig. 5. A first remark is that the force directed layout places campaigns (respectively articles) close one to each other whenever they have common links to articles (respectively campaigns), and as

a result provides a visual and soft co-clustering of these subgraphs. This is very clear regarding the very dense 7-88 core subgraph whose campaigns (in red) are clearly divided into two groups on the left and right part of the central part of the graph. The second 4-52 core subgraph in the stratification process is clearly organised around small and distinct groups of campaigns.

The two groups of campaigns appearing in the A_1 strate are $\{cc5, cc6, cc7, cc8, cc9, cc10, cc13, cc14, cc18, cc19, cc20, cc21\}$ and $\{cc26, cc27, cc28, cc29, cc30, cc32, cc34, cc37\}$, while A_2 contains campaigns $\{cc1, cc2, cc3, cc4, cc7, cc12, cc15, cc25, cc31, cc36, cc41, cc42, cc43, cc46, cc48, cc49, cc53, cc56, cc59, cc60\}$.

In the $A_1(7-88)$ strate all campaigns occurred before year 2000, most of them were located in New Caledonia and almost all the articles lead to nomenclatural acts (TA) and do not use molecular characteristics. The main differences between the two subgroups mentioned above are as follows. The first group (on the left of the subgraph) is made of campaigns which have more specific objectives (Chemistry, Geology and Halieutic Research) and share more articles, i.e. articles that rely on specimens that come from many campaigns. The second group (on the right) is made of campaigns with more general objectives (Faunistic Inventory) and share less articles, these articles dealing then with specimens that come from a smaller number of campaigns.

The $A_2(4-52)$ strate differs from the $A_1(7-88)$ strate when considering the article categorization (see Table 2): the classification without nomenclatural purpose (TS) is overrepresented in A_2 with respect to the A_1 strate. Besides, the use of molecular characteristics in these articles is overrepresented with respect to all the strates. This is in line with the campaign attributes in this strate, as these campaigns have more geographical locations (7 out of the 13 locations) and are more recent (7 out of the 20 campaigns are posterior to year 2000) than in the A_1 strate: recent technical progresses allow a more reliable use of molecular characteristics of the collected specimens [4].

Table 2. Bibliographical categorization in first strates and in the whole graph (in percentages). The last column is the percentage of articles using molecular characteristics

Strate	TA	TS	AT	Other	Mol. Char.
A_1^N 7-88	88,5	3,8	6,2	1,4	5,7
A_2^N 4-52	85,1	9,2	5,3	0,9	18,9
E	69,4	11,7	11,9	7,1	14

We display Fig. 6 the 4-20 core A_3^N of the stratification A^N obtained by maximizing the number of edges and compare it to the 2-44 core A_4^M of the stratification A^M obtained by maximizing the local modularity. We do not discuss here A_3^M which concerns few vertices (see Table 1). The two subgraphs represent different views and emphasize the fact that the two different maximization processes may select subgraphs with different structures. In one hand,

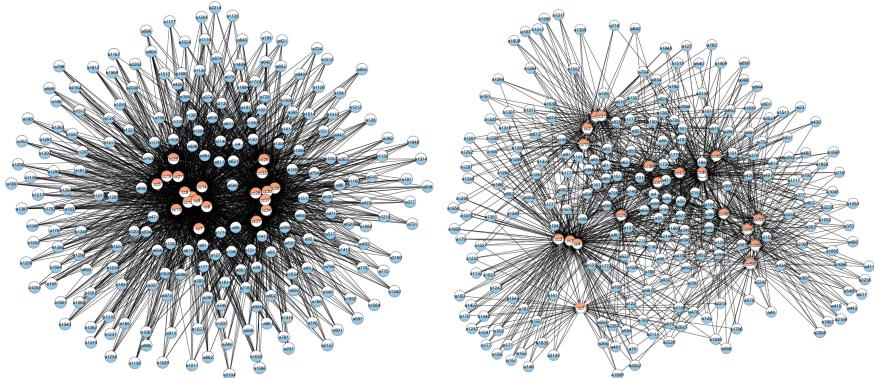


Fig. 5. The two first core subgraphs of the ACC network stratification, common to both maximization criteria. On the left, $A_1 = E_{7,88}$ the 7-88 HA-core subgraph of ACC and on the right the $A_2 = E_{4,52}(E \setminus E_{7,88})$ core subgraph.

the 4–20 core A_3^N contains 55 out of the 80 campaigns and 298 articles citing these campaigns out of the 1480 articles of the original graph. This core focusses on articles citing several campaigns (at least 4) cited by a medium number (at least 20) of articles. On the other hand, the 2–44 core A_4^M contains only 13 out of the 80 campaigns and 279 articles citing these campaigns: it focusses on campaigns cited by a large number (at least 55) of articles. Again, the force-directed layout allows to identify various groups of campaigns and articles citing them.

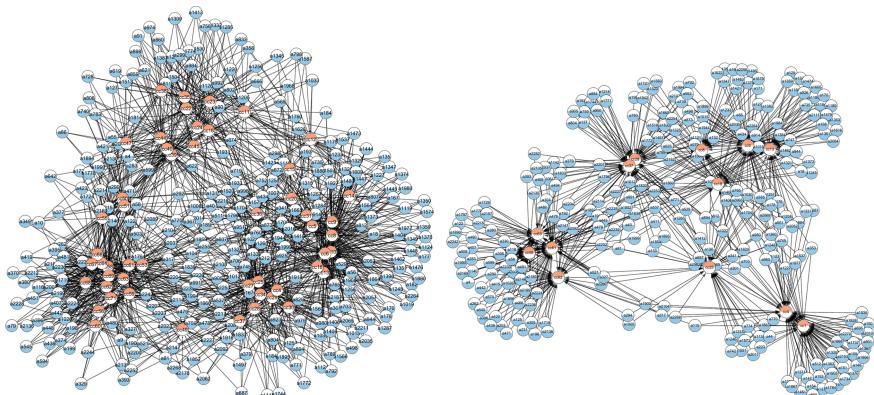


Fig. 6. Two core subgraphs from the ACC network stratifications. On the left, A_3^N the 4-20 core from stratification A^N and on the right A_4^M the 2-44 core from A^M .

We have also experimented the LinkComm method¹ proposed in [1]. LinkComm is a hierarchical clustering of links whose purpose is to discover link communities, i.e. edge subsets. The similarity between two edges ik and jk is obtained as the Jaccard Index of the neighborhoods of nodes i and j while the similarity between ik and jw is null whenever $w \neq k$. In our bipartite setting this means that i and j have to belong to the same vertex set, e.g. given the campaign set, and that ik and jk are stated all the more similar the more i and j are cited by the same articles. A link partition is then obtained by computing an optimum partition density threshold to cut the hierarchy. When applying the method to our network, we obtained 1581 communities from which only 51 contained at least 10 edges. We considered the 5 largest link communities $C_1 \dots C_5$ whose sizes are respectively 2879, 495, 317, 229 and 143 and looked at the corresponding campaign subsets of respective sizes 19, 3, 3, 2, 2 they refer to. Campaign subsets in C_1 and A_1 have Jaccard similarity 0.8: campaigns cc12 and cc36 belong to $C_1 \setminus A_1$ while cc5 and cc27 belong to $A_1 \setminus C_1$. However C_1 contains much more edges (see Table 1) many of which are linked to only one campaign. Interestingly we may also see comparing C_1 represented Fig. 7 to A_1 (see Fig. 5)

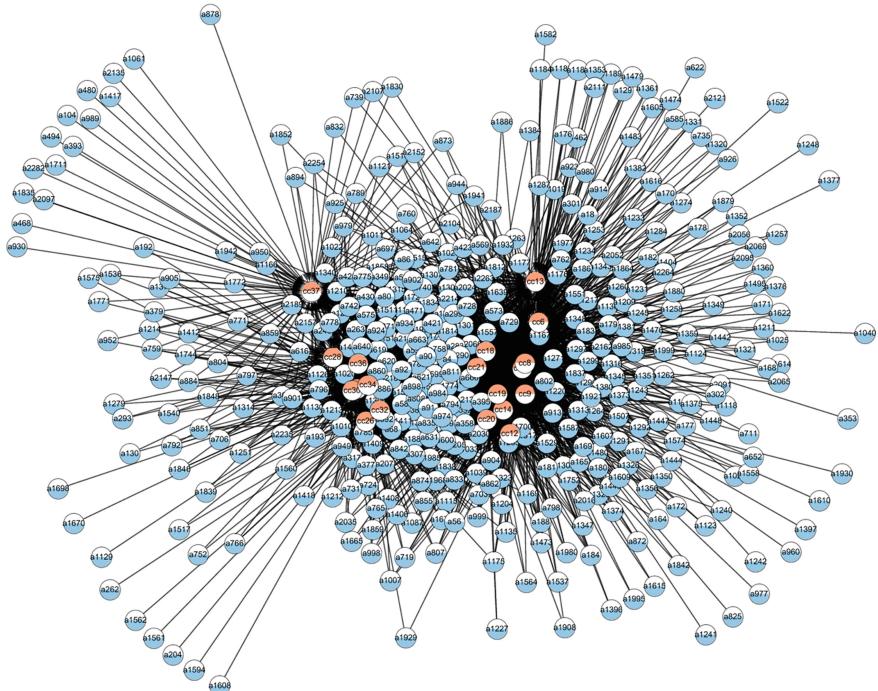


Fig. 7. Graph of the largest community C_1 found in the network ACC using LinkComm

¹ We use the implementation by Jorge C. Valverde-Rebaza to be found at <http://www.labic.icmc.usp.br/jvalverr/>

that the two campaign subsets may also be visually partitioned in two subgroups in the same way. Regarding link communities C_2 to C_5 only few campaigns are represented in each community. However 7 out of the 10 campaigns appearing in these 4 link communities also belong to the set of 20 campaigns appearing in the edge subset A_2 . Overall this means that as expected, stratification differs from link communities detection in that several communities, related by a same density level, appear in the same strata A_i . Still, the two methods reveal related structural information about the network.

5 Conclusion

We have proposed and experimented an edge decomposition method named stratification. It relies on a core definition, depending on parameters in such a way that the resulting edge subsets, called strates, are ordered by decreasing requirements associated to decreasing core parameters. We investigated an epistemological attributed two-mode network and vertex attributes were used a posteriori to discuss the edge decomposition and we did observe that the first strates are associated to particular attribute values. When comparing the stratification results to the results of Linkcomm, whose purpose is to identify link communities, we found that the two methods exhibits different but correlated decompositions: while Linkcomm is designed to find link communities the stratification purpose is to extract subgraphs with decreasing density levels.

References

1. Ahn, Y.Y., Bagrow, J.P., Lehmann, S.: Link communities reveal multiscale complexity in networks. *Nature* **466**(7307), 761–764 (2010)
2. Bary, S.: Scientific representations of biodiversity in the deep-sea: an epistemologic and scientific approach. Ph.D. thesis, Ecole Doctorale 474, Sorbonne Paris Cité (2018)
3. Batagelj, V., Zaversnik, M.: Fast algorithms for determining (generalized) core groups in social networks. *Adv. Data Anal. Classif.* **5**(2), 129–145 (2011)
4. Bouchet, P., Bary, S., Héros, V., Marani, G.: How many species of molluscs are there in the world’s oceans, and who is going to describe them? In: *Tropical Deep-Sea Benthos 29. Mémoires du Muséum national d’Histoire Naturelle* **1993**(208), 9–24 (2016)
5. Cerinsek, M., Batagelj, V.: Generalized two-mode cores. *Soc. Netw.* **42**, 80–87 (2015)
6. Flake, G.W., Lawrence, S., Giles, C.L.: Efficient identification of web communities. In: *KDD*, pp. 150–160. ACM, New York (2000)
7. Giatsidis, C., Thilikos, D.M., Vazirgiannis, M.: D-cores: measuring collaboration of directed graphs based on degeneracy. *Knowl. Inf. Syst.* **35**(2), 311–343 (2013)
8. Kobourov, S.G.: Force-directed drawing algorithms. In: *Handbook of Graph Drawing and Visualization*, pp. 383–408. Chapman and Hall/CRC, United Kingdom (2013)
9. Newman, M.E.J.: Modularity and community structure in networks. *Proc. Natl. Acad. Sci.* **103**(23), 8577–8582 (2006)

10. Palla, G., Derényi, I., Farkas, I., Vicsek, T.: Uncovering the overlapping community structure of complex networks in nature and society. *Nature* **435**(7043), 814–818 (2005)
11. Seidman, S.B.: Network structure and minimum degree. *Soc. Netw.* **5**, 269–287 (1983)
12. Soldano, H., Santini, G.: Graph abstraction for closed pattern mining in attributed networks. In: Schaub, T., Friedrich, G., O’Sullivan, B. (eds.) European Conference on Artificial Intelligence (ECAI). Frontiers in Artificial Intelligence and Applications, vol. 263, pp. 849–854. IOS Press, Amsterdam (2014)
13. Soldano, H., Santini, G., Bouthinon, D., Lazega, E.: Hub-authority cores and attributed directed network mining. In: 29th IEEE International Conference on Tools with Artificial Intelligence, ICTAI 2017, Boston, MA, USA, November 6–8, 2017, pp. 1120–1127. IEEE Computer Society (2017)



OTARIOS: OpTImizing Author Ranking with Insiders/Outsiders Subnetworks

Jorge Silva^(✉), David Aparício, and Fernando Silva

CRACS & INESC-TEC, DCC-FCUP, Universidade do Porto, Porto, Portugal
jmbs@inesctec.pt, {daparicio,fds}@dcc.fc.up.pt

Abstract. Evaluating scientists based on their scientific production is often a controversial topic. Nevertheless, bibliometrics and algorithmic approaches can assist traditional peer review in numerous tasks, such as attributing research grants, deciding scientific committees, or choosing faculty promotions. Traditional bibliometrics focus on individual measures, disregarding the whole data (i.e., the whole network). Here we put forward OTARIOS, a graph-ranking method which combines multiple publication/citation criteria to rank authors. OTARIOS divides the original network in two subnetworks, insiders and outsiders, which is an adequate representation of citation networks with missing information. We evaluate OTARIOS on a set of five real networks, each with publications in distinct areas of Computer Science. When matching a metric's produced ranking with best papers awards received, we observe that OTARIOS is >20% more accurate than traditional bibliometrics. We obtain the best results when OTARIOS considers (i) the author's publication volume and publication recency, (ii) how recently his work is being cited by outsiders, and (iii) how recently his work is being cited by insiders and how individual he is.

1 Introduction

Deciding scientific committees, research grants, or faculty promotions is still done mostly by peer review. Nevertheless, bibliometrics have been proposed to assist the peer review process [13]. Bibliometrics typically rely on the author's productivity (i.e., statistics of author's papers) and the author's impact (i.e., statistics of author's citations) [1], e.g., one of the most widely used bibliometrics is the author's h-index [4], which measures the impact only of his most relevant works.

However, traditional bibliometrics have the drawback of only assigning impact to authors' direct citations, thus ignoring indirect citations. For example, if A cites B , and B cites C , traditional bibliometrics give no merit to C from A 's indirect citation. To address this limitation, graph algorithms have been developed for citation networks [2, 3, 8, 15]. These algorithms are modifications of PageRank applied to citation networks [7]. One of PageRank's major ideas is that not all nodes are equal, i.e., it is good to be referenced by any webpage but it is better to be referenced by important webpages. This idea is general

and applicable to citation networks, i.e., it is important to be cited by important authors. However, traditional PageRank assumes that the full network is known, which is not true for most real citation networks.

Here we propose a novel graph-ranking algorithm for citation networks, named OTARIOS (**O**p**T**imizing **A**uthor **R**anking with **I**nnsiders/**O**utnsiders **S**ubnetworks). OTARIOS divides the citation network in two subnetworks, *insiders* (i.e., authors for which we have all their citations) and *outsiders* (i.e., authors that cite insiders but for whom not all citations are known). Then, only insiders are ranked while outsiders influence the ranks of insiders. OTARIOS efficiently combines different publication/citation attributes in a multi-edge weighted network (instead of a simple weighted network used by traditional PageRank). Furthermore, OTARIOS is a flexible algorithm, allowing users to personalise which publication/citation attributes are used to rank researchers (e.g., value venue prestige highly or lowly).

We evaluate OTARIOS on a set of five networks belonging to different areas of Computer Science. Our ground-truth is the number of best paper awards (i.e., a good method should rank highly authors that have won best papers). Our results show that, for this task, OTARIOS is >30% more accurate than PageRank and >21% more accurate than baseline bibliometrics. OTARIOS obtains the best results when considering (i) the author’s publication volume and recency, (ii) how recently his work is being cited by outsiders, and (iii) how recently his work is being cited by insiders and how individual he is (i.e., publishing a paper with few co-authors).

The paper is organised as follows. Section 2 describes terminology used throughout the work. Section 3 describes OTARIOS and our methodology. Section 4 presents the performance of OTARIOS on a set of five networks. Finally, Sect. 5 presents our main conclusions and gives some directions for future work.

2 Preliminaries

2.1 Terminology

For consistency, we denote sets by calligraphic letters (e.g., \mathcal{S}), elements of those sets (i.e., entities) by the respective capital letter with an index (e.g., $S_i \in \mathcal{S}$), attributes of entities (e.g., year, impact factor) as functions named in lower-case alphabetic or greek letters (e.g., $a(S_i)$ or $\alpha(S_i)$), and constants as sole greek letters (e.g., τ). The cardinality of a given set S is denoted by $|\mathcal{S}|$.

Problem 1. Given a set of papers \mathcal{P} published in a set of venues \mathcal{V} by a set of authors \mathcal{A} , who are the n top-ranked authors?

A paper $P_j \in \mathcal{P}$ is co-authored by authors $\mathcal{A}_{P_j} \subseteq \mathcal{A}$. Likewise, an author $A_i \in \mathcal{A}$ is (one of) the author(s) of papers $\mathcal{P}_{A_i} \subseteq \mathcal{P}$. In paper-level networks, graph $G = \{\mathcal{N}, \mathcal{E}\}$ comprises a set \mathcal{N} of nodes that represent papers and a set \mathcal{E} of edges that represent paper citations, written as $P_{j'} \rightarrow P_j$. In author-level

networks, nodes represent authors and edges represent author citations, written as $A_{i'} \rightarrow A_i$.

Regarding node attributes, papers have meta-data which we use as attributes, namely their publication year, venue prestige, and the number of references, represented by $y(P_j)$, $v(P_j)$ and $r_{out}(P_j)$, respectively. The *recency* of a paper, represented by $\delta(P_j)$, is given by Eq. 1. Similarly, the *recency* of an author, represented by $\delta(A_i)$, is the recency of his most recent paper (Eq. 2). The venue prestige of a paper P_j depends on the venue $V_k \in \mathcal{V}$ where it was published and the year when it was published, represented by $v(P_j) = \lambda(V_k, y(P_j))$. We estimate venue prestige with *CiteScore* [1] (Eq. 3), where $p(V_k, y)$ is the number of papers published in V_k in year y and $c(V_k, y)$ is the number of citations that all papers published in V_k in year y received. Thus, venues with many citations per paper have higher prestige.

Recency of a paper	Recency of an author
$\delta(P_j) = \left(\max_{P_{j'} \in \mathcal{P}} y(P_{j'}) \right) - y(P_j) \quad (1)$	$\delta(A_i) = \min_{P_j \in \mathcal{P}_{A_i}} \delta(P_j) \quad (2)$
Venue prestige	Cited individuality
$\lambda(V_k, y) = \frac{c(V_k, y)}{\sum_{x=1}^3 p(V_k, y-x)} \quad (3)$	$w(A_{i'} \rightarrow A_i, P_j) = \frac{1}{ \mathcal{A}_{P_j} }, A_i \in \mathcal{A}_{P_j} \quad (4)$
Citation recency	Citation prestige
$a(A_{i'} \rightarrow A_i, P_j) = e^{\frac{-\delta(P_j)}{\tau}}, A_{i'} \in \mathcal{A}_{P_j} \quad (5)$	$v(A_{i'} \rightarrow A_i, P_j) = v(P_j), A_{i'} \in \mathcal{A}_{P_j} \quad (6)$

Regarding edges, in paper-level networks edges are traditionally unweighted and simple, i.e., two papers are connected by a single edge with weight equal to 1 [3, 5]. In author-level networks, edges are weighted and multiple, i.e., two authors are connected by multiple edges with different weights. These multiple edges concern different edge attributes that depend on the publication P_j where author $A_{i'}$ cites author A_i . The recency of an edge, represented by $a(A_{i'} \rightarrow A_i, P_j)$, gives more importance to recent citations (Eq. 5). As discussed in the YetRank paper which originally proposes this concept for author ranking algorithms [5], we set the decay factor $\tau = 4$. The venue prestige of an edge, represented by $v(A_{i'} \rightarrow A_i, P_j)$, gives more importance to citations in important venues (Eq. 6). Finally, the individuality of an edge, represented by $w(A_{i'} \rightarrow A_i, P_j)$, gives more importance to citations received in papers that author A_i has few (or no) co-authors (Eq. 4). Thus, $w(A_{i'} \rightarrow A_i, P_j)$, unlike $a(A_{i'} \rightarrow A_i, P_j)$ and $v(A_{i'} \rightarrow A_i, P_j)$, depends on the cited author A_i and not on the citing author $A_{i'}$. The author's attribute total out-edge weight is obtained by summing all of its out-edges, e.g., for citation recency, $a_{out}(A_i) = \sum_{(A_i \rightarrow A_{i'}, P_j)} a(A_i \rightarrow A_{i'}, P_j)$. w_{out} and v_{out} are obtained in the same way.

2.2 Bibliometrics and PageRank

Measuring the scientific impact of institutions, journals, or authors is an important task in the peer review process. Here we focus on measuring the impact of authors, i.e., author ranking. Citations have been widely used by traditional bibliometrics to measure an author's impact. For instance, the widely used h-index [4] measures an author's impact by the number of citations of his most cited papers. However, h-index and similar bibliometrics fail to capture the nature of scientific development since they disregard the fact that a new discovery is not solely due to previous work directly referenced. Graph-based metrics, on the other hand, correctly spread the credit to previous works that paved the way [14].

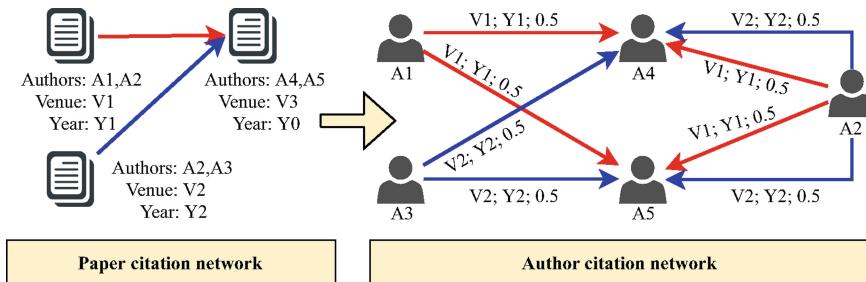


Fig. 1. Transformation of a paper citation network into a author citation network.

Graph-based measures can be computed on paper citation networks or on author citation networks [14]. In our case, we transform the original (unweighted and simple) paper citation network into an (weighted and multi-edge) author citation network (Fig. 1). Numerous graph centrality measures exist and can be used to measure node importance [12]. PageRank [7] is one of the most widely used measures and its rationale is intuitive in author-ranking, i.e., authors cited by important authors are more relevant than authors cited by unimportant authors.

PageRank consists of two steps: score initialisation and score diffusion. The score initialisation step creates a vector R that defines an initial score for every node using *a priori* information. In the simplest case, every node is considered equally important, thus an uniform distribution is used (i.e., $R[A_i] = \frac{1}{|\mathcal{A}|}$) [2, 7, 9]. The score diffusion step updates the node scores taking into consideration network dynamics. Score diffusion is an iterative process which computes three addends: random restart, dangling nodes, and score term. Random restart (RR) evaluates how likely it is to reach a certain node by moving randomly in the network. PageRank defines a value q as the random restart probability, and q is multiplied by the node's initial score R (thus, nodes with higher initialisation receive higher random restart score). Dangling nodes (DN) is a process where the score of nodes that do not have any out-links is split by all other nodes. This is performed to avoid having nodes that do not disseminate their credit. Like

random restart, this division takes into consideration the initialisation vector R , thus nodes initialised with higher values give more credit to other nodes. Score term (ST) updates the score of a node i , according to the score of his in-links. In author citation networks, for instance, scores can be split evenly by co-authors of the cited publication, e.g., if the papers has two authors, the score is divided by the two authors, if it has three authors, the score is divided by the three authors, thus, in the case of three authors, each author receives less credit than in the case with just two authors [2, 8, 15]. PageRank stops when a new iteration produces very low variation in the node scores [7].

One drawback of PageRank is that it assumes that the complete network is known. However, in real-world cases, it is not possible to obtain a complete citation network. Let's assume that we want to rank a set of authors \mathcal{A} . First, we need to expand the network by obtaining all authors $B_i \in \mathcal{B}$ that cite any $A_i \in \mathcal{A}$ such that $B_i \notin \mathcal{A}$. Then, we need to also extract all authors $C_i \in \mathcal{C}$ that cite any $B_i \in \mathcal{B}$ such that $C_i \notin (\mathcal{A} \cup \mathcal{B})$, to correctly determine the scores of all $A_i \in \mathcal{A}$, i.e., C_i does not cite A_i directly but he cites some B_i that cites A_i , thus C_i is actually citing A_i indirectly. Ideally, this should be performed until the complete set of authors (and citations) with seed \mathcal{A} is obtained. Due to memory and time constraints, we can only obtain a sample of the citation network. As a result, traditional PageRank estimates scientific rankings based on incorrect information, i.e., authors in the periphery are not being adequately taken into account since their citations are not in the network. Although there is no ideal solution for this problem, one can be more careful in estimating the rank of nodes in the periphery.

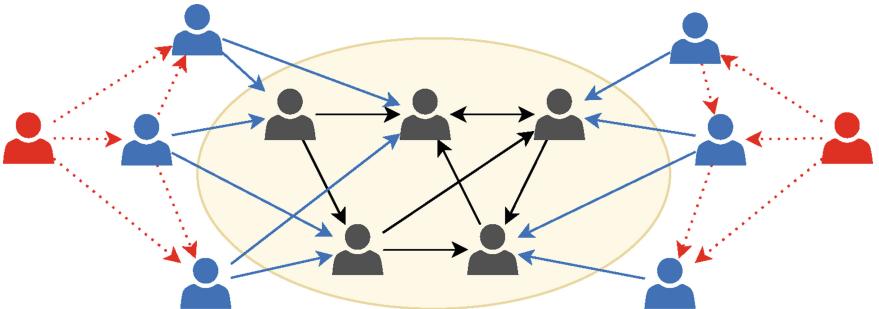


Fig. 2. Example of insiders and outsiders subnetworks. Insiders are nodes/authors coloured in black and outsiders are blue. Note that no links between outsiders exists (dashed red lines). Furthermore, no information exists of outsiders that do not cite any insiders (coloured in red).

3 Methodology

3.1 Problem Description

We propose a new author-level graph-ranking methodology. We start with a set of authors \mathcal{I} that we want to rank (e.g., authors that publish in certain conferences). First, we obtain all citations between all authors $I_i, I_{i'} \in \mathcal{I}$ (i.e., a complete citation network for \mathcal{I}). Second, for each author I_i , we obtain all of his received citations coming from authors $O_i \notin \mathcal{I}$. The process stops here, i.e., we do not obtain all received citations for authors $O_i \in \mathcal{O}$. Doing so iteratively is unfeasible in practice because the number of authors added at each step grows very rapidly. Thus, we divide the citation network into just two groups of nodes: *insiders* (\mathcal{I}) and *outsiders* (\mathcal{O}), i.e., $\mathcal{A} = \{\mathcal{I}, \mathcal{O}\}$ (Fig. 2). Note that no outsider can also be an insider, and vice-versa. Edges connect insiders ($\mathcal{E}_{\mathcal{I}}$) or outsiders to insiders ($\mathcal{E}_{\mathcal{O}}$), but no edges exist from insiders to outsiders nor between outsiders, i.e., $\mathcal{E} = \{\mathcal{E}_{\mathcal{I}}, \mathcal{E}_{\mathcal{O}}\}$.

We aim to estimate the prestige of insiders (i.e., obtain their rank). We do not rank outsiders, instead we use them to increase the accuracy of ranks calculated for insiders. We estimate outsiders' prestige (λ) before insiders rank initialisation. We use the outsiders' history of publications, giving higher prestige to authors with many citations ($c(A_i)$) in few publications ($p(A_i)$) (Eq. 7). Our objective is to increase the initial rank of insiders that are cited by outsiders with high prestige.

$$\text{Outsider prestige} \quad \lambda(A_i) = \frac{c(A_i)}{p(A_i)} \quad (7)$$

3.2 OTARIOS

OTARIOS is a graph-based algorithm for author-level citation networks. Its aim is to rank authors based on their publication and citation history. OTARIOS uses the notion of insider/outsider subnetworks to adequately estimate authors scores in a network with limitation information. Furthermore, OTARIOS is a flexible algorithm that analyses which set of publication/citation attributes lead to better rankings.

On the first step, OTARIOS computes an initial score for each author, represented by $R(A_i)$. OTARIOS calculates $R(A_i)$ by taking into account multiple criteria that favour different author characteristics (Table 1). We divide the criteria into two categories: productivity and outsiders influence. Productivity measures the value of the author's publications, while outsider influence measures the value of the author's citations coming from outsiders. Regarding productivity, OTARIOS takes three factors into account: volume, recency and venues. Regarding outsiders influence, OTARIOS takes another three factors into account: individuality, recency and venues. We compute the author's initial score $R(A_i)$ as the sum of the two products of the factors in each

group (i.e., productivity ($volume \times recency \times venues$) + outsiders influence ($individuality \times recency \times venues$)).

On the second step, OTARIOS improves author scores in an iterative process. Outsiders are removed from the network since their presence degrades the score diffusion step. In each iteration, OTARIOS updates an author's score $S(A_i)$ as $ST(A_i) + RR(A_i) + DN(A_i)$. We compute $RR(A_i)$ and $DN(A_i)$ in function of the initial rank of each author (discussed in Table 1), and compute $ST(A_i)$ in function of the author's citations coming from other insiders. OTARIOS considers three different criteria to assess score term $ST(A_i)$: individuality, recency and venues (Table 2). The $ST(A_i)$ at each iteration is the product of every criteria. (i.e., score term ($individuality \times recency \times venues$)). Like PageRank, OTARIOS stops when it reaches low variation in the node scores.

Here we do not assume that every criteria should be used for author ranking. The criteria's importance depends greatly on the dataset. For instance, venue prestige might be very important to rank some communities (i.e., top authors publish in top conferences of that scientific area, e.g., machine learning) but irrelevant in some other community because we are studying a specific conference (i.e., all authors publish in the same venue, e.g., KDD). OTARIOS is parameterisable, i.e., users can define by which criteria authors are ranked. For example, for a certain application, we may want to rank authors taking into account recent publications and the venue prestige of citations coming from both insiders and outsiders¹.

4 Results

We evaluate OTARIOS on five created networks, each consisting of publications in top-tier Computer Science conferences (Table 3). For each network, we create a ground truth based on peer review using the best paper award information for every conference², i.e., each rewarded paper has a unit of prestige which is equally divided by its authors. Thus, each author has a certain ground-truth prestige that is the sum of the prestige of his awards. As a result, we are assuming that authors that have won more awards with fewer co-authors should be ranked higher.

In our experiments, methods that produce rankings more similar to the ground truth ranking are considered better. For this purpose, for each network and for each method, we compare the method's predicted ranking with the network's ground truth using two commonly used ranking quality measures: Normalized Discounted Cumulative Gain (NDCG) and Mean Reciprocal Rank (MRR). NDCG measures how far the predicted ranking is from the ideal ground-truth ranking (i.e., NDCG closer to 1 is better) [6], while MRR is the mean

¹ Note that we define variants using notation APV+AVW+AVW, where the addends define the criteria used at each group. The first for productivity, the second for outsiders influence and the last for score term. For the example in the text, the variant nomenclature is A+V+V.

² Awards information obtained from: https://jeffhuang.com/best_paper_awards.

Table 1. List of criteria used for OTARIOS’ author rank initialisation: $R(A_i)$. OTARIOS considers both the authors’ productivity and the direct influence of outsiders on the authors. We create different variants of these criteria, e.g., $PV + V$ uses volume (P) and venue prestige (V) to measure author productivity, and uses venue prestige (V) to measure the direct influence of outsiders.

	Criteria	Initialisation: $R(A_i)$	Description
Productivity	Volume (P)	$\frac{1}{(P_j \in^{\#} A_i)^{ ^{\#} P_j }}$	Favours publishing many papers with few co-authors
	Recency (A)	$e^{-\delta(A_i)} / \tau$	Favours publishing recently
	.5 Venues (V)	$(\sum_{(P_j \in^{\#} A_i)} v(P_j)) \times ^{\#} A_i ^{-1}$	Favours publishing in prestigious venues
Outsiders Influence	.5 Individuality (W)	$\sum_{(A_{i'} \rightarrow A_i, P_j)} \frac{\lambda(A_{i'}) \times w(A_{i'} \rightarrow A_i, P_j)}{w_{out}(A_{i'})}, A_{i'} \in \mathcal{O}$	Favours being cited by outsiders that cite few authors
	.5 Recency (A)	$\sum_{\substack{(A_{i'} \rightarrow A_i, P_j) \\ \mathcal{O}}} \frac{\lambda(A_{i'}) \times a(A_{i'} \rightarrow A_i, P_j)}{a_{out}(A_{i'})}, A_{i'} \in \mathcal{O}$	Favours being cited by outsiders more recently
	Venues (V)	$\sum_{(A_{i'} \rightarrow A_i, P_j)} \frac{\lambda(A_{i'}) \times v(A_{i'} \rightarrow A_i, P_j)}{v_{out}(A_{i'})}, A_{i'} \in \mathcal{O}$	Favours being cited by outsiders in prestigious venues

Table 2. List of criteria used for OTARIOS’ author score term calculation: $ST(A_i)$. Combined with author initialisation (Table 1), we create different variants, e.g., $PV+V+A$ combines initialisation $PV+V$ with score term A, i.e., using citation recency. All variants use $RR(N_i) = q \times R(N_i)$ and $DN(N_i) = (1 - q) \times R(N_i)$, thus we omit them from the table.

Criteria	Score term: $ST(A_i)$	Description
Individuality (W)	$\sum_{(A_{i'} \rightarrow A_i, P_j)} \frac{S(A_{i'}) \times w(A_{i'} \rightarrow A_i, P_j)}{w_{out}(A_{i'})}, A_{i'} \in \mathcal{G}$	Favours being cited by insiders that cite few authors
Recency (A)	$\sum_{(A_{i'} \rightarrow A_i, P_j)} \frac{S(A_{i'}) \times a(A_{i'} \rightarrow A_i, P_j)}{a_{out}(A_{i'})}, A_{i'} \in \mathcal{G}$	Favours being cited by insiders more recently
Venues (V)	$\sum_{(A_{i'} \rightarrow A_i, P_j)} \frac{S(A_{i'}) \times v(A_{i'} \rightarrow A_i, P_j)}{v_{out}(A_{i'})}, A_{i'} \in \mathcal{G}$	Favours being cited by insiders in prestigious venues

ground-truth ranking of the predicted ranking (i.e., low MRR is better). Both measures are calculated considering only the top- n authors, referred to as @ n . We compute MRR and NDCG for @5, @10, @20, @50 and @100.

4.1 Performance of OTARIOS Variants

In our analysis we do not assume that combining all publication/citation information necessarily leads to the best results. Instead, we start with simple OTARIOS variants and progressively add new attributes. We illustrate this process for network NET (Table 4). We begin by comparing OTARIOS variants that

Table 3. Set of networks used for experimental evaluation. Data was taken from [10, 11]. The full DBLP dataset contains over 3M publications from 1936 to 2018. Each network contains publications from only a set of conferences, e.g., networks TC contains publications from FOCS, SODA and STOC. For each network we show the number of insider and outsider nodes, $|\mathcal{G}|$ and $|\mathcal{O}|$ respectively, and the number of insider and outsider edges, $|\mathcal{E}_g|$ and $|\mathcal{E}_o|$ respectively.

Network	Conferences	Nodes		Edges	
		$ \mathcal{G} $	$ \mathcal{O} $	$ \mathcal{E}_g $	$ \mathcal{E}_o $
CM	AAAI, IJCAI, ICML, ACL, ICCV, CVPR	35.6k	224.9k	4.6M	4.9M
TC	FOCS, SODA, STOC	5.0k	82.4k	0.5M	0.8M
NET	INFOCOM, NSDI, SIGCOMM, MOBICOM, SIGMETRICS	15.2k	138.8k	2.1M	3.7M
IS	KDD, CIKM, PODS, SIGMOD, VLDB, WWW, SIGIR	282.7k	190.9k	4.0M	5.1M
SE	PLDI, FSE, ICSE, OSDI, SOSP	10.8k	99.9k	1.0M	2.1M

only consider outsiders influence (e.g., $\emptyset + A + \emptyset$). For the best ones, we added productivity criteria (e.g., $AP + A + \emptyset$). In general, we see that results improve when merging outsiders influence with productivity. Finally, we add score term calculation to OTARIOS (e.g., $AP + A + A$). For the NET network, we see that $AP + A + A$ is the variant that obtains the best results, with a mean NDCG of 0.330 and a mean MRR of 606.

Table 4. Comparison of OTARIOS variants on network NET. For each OTARIOS variant, we measure NDCG and MRR for the top- n ranked authors (@n), as well as the metric’s mean value. Bold highlights the highest score for each metric. The best OTARIOS variant is coloured in blue.

OTARIOS variant	NDCG					MRR						
	@5	@10	@20	@50	@100	Mean	@5	@10	@20	@50	@100	Mean
$\emptyset + A + \emptyset$	0.269	0.233	0.207	0.186	0.174	0.214	443	1125	903	1526	2066	1213
$\emptyset + V + \emptyset$	0.269	0.233	0.207	0.186	0.185	0.216	412	1108	916	1522	2096	1211
$\emptyset + AV + \emptyset$	0.269	0.233	0.207	0.186	0.177	0.215	419	1109	902	1511	2074	1203
$AP + A + \emptyset$	0.288	0.246	0.259	0.218	0.241	0.250	350	500	440	1121	1502	783
$AP + V + \emptyset$	0.288	0.246	0.258	0.218	0.239	0.250	344	489	439	1134	1527	787
$AP + AV + \emptyset$	0.288	0.246	0.259	0.218	0.240	0.250	345	494	439	1143	1523	789
$AP + A + A$	0.380	0.297	0.283	0.282	0.280	0.304	385	647	472	1111	1416	806
$AP + A + AV$	0.407	0.345	0.291	0.291	0.274	0.322	242	614	473	1116	1455	780
$AP + A + AW$	0.381	0.369	0.313	0.302	0.288	0.330	219	386	328	879	1219	606

We compare OTARIOS with PageRank [2] and CountRank (CR), a baseline bibliometric. CR counts the total citations received by each author. We use CR because our DBLP dataset does not contain an author’s h-index and collecting this information for all 300k authors is not feasible. We create three CR variants: uniform, individuality and position. For each citation an author receives,

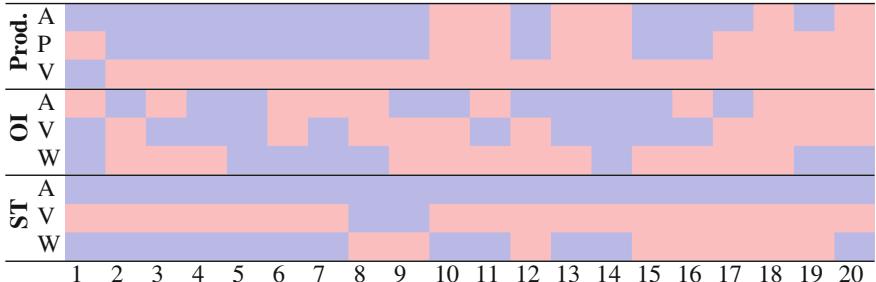
uniform assigns the same merit to each co-author, individuality equally divides the merit by all co-authors, and position gives more credit to authors whose name appears first in the publication (i.e., the first author gets double the merit of the second author, triple of the third author, etc.). Our results show that OTARIOS achieves significantly better results than PageRank (by >30%) and CR (by >20%) (Table 5). We only show results for the five best OTARIOS variants. We should note that 21 OTARIOS variants, of the total 53, obtain better mean MRR and NDCG than the best competitor ($CR_{individuality}$). The best OTARIOS variants obtain a mean NDCG of 0.246 and a mean MRR of 567 over all five networks. Our best OTARIOS variant overall (i.e., considering a combination of NDCG and MRR) is AP+A+AW, which considers (i) the author’s publication volume and publication recency, (ii) how recently his work is being cited by outsiders, and (iii) how recently his work is being cited by insiders and how individual he is.

In order to analyse which criteria are typically more important, we visualise which criteria the top OTARIOS variants are using (Table 6). The best OTARIOS variants (i.e., all top-9 variants) always use a mix of productivity, outsiders influence and score term, showing the importance of considering multiple aspects of publications and citations information. Recency (A) seems to be more relevant to evaluate productivity and insiders score term than outsiders influence. On the other hand, venue prestige (V) seems to be more relevant when measuring outsiders influence than author productivity and insiders score term. This is expected because insiders tend to publish in the same venues, while outsiders might cite insiders in any venues, thus the venue prestige of outsider citations varies greatly.

Table 5. Comparison of OTARIOS against PageRank and simple bibliometrics. The value of each cell is the metric’s mean value for that network (e.g., the mean NDCG and MRR of AP+A+AW for network NET is highlighted in Table 4). Bold highlights the highest score for each metric. The best competitor is colored in red and the best OTARIOS variant is colored in blue. Inside parentheses we show the gain of OTARIOS versus $CR_{individuality}$.

Method	NDGC					Mean	MRR					Mean	
	CM	TC	NET	IS	SE		CM	TC	NET	IS	SE		
$CR_{position}$	0.097	0.049	0.189	0.176	0.261	0.154	1427	463	1009	892	324	823	
$CR_{uniform}$	0.138	0.045	0.278	0.189	0.222	0.174	1659	516	1066	1067	387	939	
PageRank	0.180	0.032	0.231	0.176	0.338	0.191	1203	508	817	720	356	721	
$CR_{individuality}$	0.129	0.043	0.247	0.211	0.372	0.200	1171	438	878	744	289	704	
OTARIOS	0 + AVW + AW	0.143	0.081	0.323	0.213	0.315	0.215	1161	324	664	707	289	629
	0 + V + AW	0.148	0.080	0.321	0.214	0.314	0.215	1169	325	671	709	294	634
	AP + VW + AW	0.150	0.087	0.330	0.268	0.383	0.244	1070	273	604	680	207	567
	AV + VW + AW	0.143	0.085	0.356	0.264	0.383	0.246	1333	285	618	676	215	626
	AP + A + AW	0.152	0.087	0.330	0.273	0.383	0.245	(+22%)	1079	272	606	688	207

Table 6. Criteria considered by the 20 best OTARIOS variants as evaluated by the mean NDCG metric. The rows represent different criteria (related to productivity (Prod.), outsiders influence (OI) and score term (ST)) and the columns the OTARIOS variants ranked at position n . Blue indicates that the criteria is considered by the variant, while the red indicates its absence.



5 Conclusions

Here we put forward OTARIOS, a new graph-ranking algorithm to measure authors' scientific impact. OTARIOS is capable of combining publication and citation information and deal with networks with limited information. We obtained the best results when OTARIOS considers (i) the author's publication volume and publication recency, (ii) how recently his work is being cited by outsiders, and (iii) how recently his work is being cited by insiders and how individual his work is (i.e., publishing with few authors is better). This evaluation was performed on a set of five networks where the ground-truth was the number of best awards in the conferences belonging to the specific network. Our tests showed that OTARIOS is $\approx 20\%$ more efficient than bibliometric approaches and $\approx 30\%$ more efficient than PageRank.

Regarding future work, we plan to test OTARIOS on paper-level citations and verify that we are also capable of improving that approach from the state of the art. Furthermore, we plan to develop a method to automatically identify outsiders (e.g., insiders with low density in the citation network, or insiders with low co-authorship ratio to other insiders) and analyse if this strategy improves author-ranking.

Acknowledgments. This work is partially funded by the ERDF through the COMPETE 2020 Programme within project POCI-01-0145-FEDER-006961, and by National Funds through the FCT as part of project UID/EIA/50014/2013. Jorge Silva is supported by a FCT/MAP-i PhD grant (PD/BD/128157/2016). David Aparício is also supported by a FCT/MAP-i PhD grant (PD/BD/105801/2014).

References

1. Elvesier, B.V.: Research Metrics Guidebook. Elvesier, Amsterdam (2018)
2. Ding, Y.: Applying weighted pagerank to author citation networks. *J. Am. Soc. Inf. Sci. Technol.* **62**(2), 236–245 (2009). <https://doi.org/10.1002/asi.21452>. <https://onlinelibrary.wiley.com/doi/abs/10.1002/asi.21452>
3. Dunaiski, M., Visscher, W.: Comparing paper ranking algorithms. In: Proceedings of the South African Institute for Computer Scientists and Information Technologists Conference, pp. 21–30. ACM (2012)
4. Hirsch, J.E.: An index to quantify an individual's scientific research output. *Proc. Natl. Acad. Sci.* **102**(46), 16569–16572 (2005)
5. Hwang, W.S., Chae, S.M., Kim, S.W., Woo, G.: Yet another paper ranking algorithm advocating recent publications. In: Proceedings of the 19th International Conference on World Wide Web, pp. 1117–1118. ACM (2010)
6. Järvelin, K., Kekäläinen, J.: Cumulated gain-based evaluation of ir techniques. *ACM Trans. Inf. Syst. (TOIS)* **20**(4), 422–446 (2002)
7. Page, L., Brin, S., Motwani, R., Winograd, T., et al.: The Pagerank Citation Ranking: Bringing Order to the Web (1998)
8. Radicchi, F., Fortunato, S., Markines, B., Vespignani, A.: Diffusion of scientific credits and the ranking of scientists. *Phys. Rev. E* **80**(5), 056,103 (2009)
9. Sidiropoulos, A., Manolopoulos, Y.: Generalized comparison of graph-based ranking algorithms for publications and authors. *J. Syst. Softw.* **79**(12), 1679–1700 (2006)
10. Tang, J., Zhang, J., Yao, L., Li, J., Zhang, L., Su, Z.: Arnetminer: Extraction and Mining of Academic Social Networks. In: KDD 2008, pp. 990–998 (2008)
11. Tang, J., Zhang, J., Yao, L., Li, J., Zhang, L., Su, Z.: Citation Network Dataset. <https://aminer.org/citation> (2017). Accessed 14 Sep 2018
12. Valente, T.W., Coronges, K., Lakon, C., Costenbader, E.: How correlated are network centrality measures? *Connections (Toronto, Ont.)* **28**(1), 16 (2008)
13. Vieira, E.S., Cabral, J.A., Gomes, J.A.: How good is a model based on bibliometric indicators in predicting the final decisions made by peers? *J. Inf.* **8**(2), 390–405 (2014)
14. Wang, H., Shen, H.W., Cheng, X.Q.: Scientific credit diffusion: researcher level or paper level? *Scientometrics* **109**(2), 827–837 (2016)
15. West, J.D., Jensen, M.C., Dandrea, R.J., Gordon, G.J., Bergstrom, C.T.: Author-level eigenfactor metrics: Evaluating the influence of authors, institutions, and countries within the social science research network community. *J. Am. Soc. Inf. Sci. Technol.* **64**(4), 787–801 (2013)



Cascading Effects of Targeted Attacks on the Power Grid

Rounak Meyur^{1(✉)}, Anil Vullikanti², Madhav V. Marathe², Anamitra Pal³,
Mina Youssef⁴, and Virgilio Centeno¹

¹ Department of Electrical and Computer Engineering, Virginia Tech, Blacksburg,
VA, USA
{rounakm8,virgilio}@vt.edu

² Biocomplexity Institute and Department of Computer Science,
University of Virginia, Charlottesville, VA, USA
{asv9v,mvm7hz}@virginia.edu

³ School of Electrical, Computer, and Energy Engineering, Arizona State University,
Tempe, AZ, USA
anamitra.pal@asu.edu

⁴ Eaton Power Center, Omaha, Nebraska, USA
minayoussef82@gmail.com

Abstract. We study the resilience of real world power grids to targeted adversarial attacks. Prior blackouts have shown that failures in the power grid can cascade, starting from a single failure, leading to a large number of failed nodes. In this paper, we study the problem of identifying a set of k critical nodes, whose failure/attack leads to the maximum number of tripped nodes. There has been a lot of work on this problem, but it has been mainly restricted to simple networks and failure models with either steady state analysis or DC power flow. In this paper, we perform AC power flow based transient analysis on a detailed power grid model. We find that a simple greedy approach yields node sets with higher criticality than a degree based approach, which has been suggested in many prior works. Furthermore, we observe that the cascades exhibit a non-monotonic behavior as a function of k .

1 Introduction

The power grid is an important societal infrastructure; its failure can have a significant impact on national and economic security of a country [2, 4, 15, 32]. An important characteristic of the power grid is that failures can cascade as a result of single node(bus) failure. For example, the well known north-east US blackout of 1965 was triggered by the tripping of a heavily loaded 230 kV transmission line due to faulty setting of a relay, resulting in a series of cascading events [31]. The blackout of 1977, on the other hand, was initiated by a lightning strike; however, equipment failures and operator errors escalated disturbances in the grid resulting in a widespread blackout [27]. Based on the NERC reports, the US

© This is a U.S. government work and not under copyright protection in the U.S.; foreign copyright protection may apply 2019

L. M. Aiello et al. (Eds.): COMPLEX NETWORKS 2018, SCI 812, pp. 155–167, 2019.
https://doi.org/10.1007/978-3-030-05411-3_13

transmission system has experienced more than 400 blackouts in between 1984 and 1999 [7]. Recently, failures in the protective systems have been instrumental in causing large blackouts like the Western Electricity Coordinating Council (WECC) blackouts in 1996 and north-east blackout (2003) [22, 30]. In 2012, India suffered the largest power outage in history initiated by a relay misoperation in one of the heavily loaded lines [23].

A fundamental problem that arises in this context is to identify *critical nodes* of the power network. The criticality of a node is the impact on the power grid when it is targeted by an adversary. *Given a parameter k , the optimal critical node problem is to find a set of k nodes, whose failure will have the largest possible impact on the power grid.* There are many metrics to quantify the impact; in this paper, we will focus on the total number of tripped nodes (buses) and the occurrence of a widespread blackout due to an unstable power swing. There has been substantial work on analyzing cascading failures in the power grid, as we discuss in Sect. 2. However, due to the complexity of power flow, most prior works simplify the problem along one of the three important dimensions: (i) assume that the underlying network is a random graph with certain structural properties, e.g., [5], (ii) simplify the power flow dynamics, e.g., either ignoring it completely (as in [5, 13, 18]), or using DC power flow or steady state analysis [1, 3, 6, 9, 26], or (iii) assume a simple failure propagation model, e.g., ignoring cascading effects [5], using sandpile models [13, 18] or simple redistribution models [33].

Such simplifications often make the analysis computationally tractable. However, the conclusions pertaining to network resilience and critical nodes might not hold for real systems. For instance, modeling failures in terms of simple node removals or sandpile type of models tends to pick nodes of the maximum (weighted) degree as the most critical nodes. However, our results show that this is not always the case: in some cases, the highest degree node is not the most critical. Similarly, using only DC power flow or steady state analysis also misses transient aspects, which we find can have a significant impact. Furthermore, the accurate representation of protection and control systems is necessary to model the failure of nodes/edges in the power system. This makes the problem very challenging.

Our Contributions. In this paper, we take a comprehensive approach to study cascading failures in power grids and identify critical nodes in the system. We combine a realistic representation of the power grid, along with a model of location level loads for Washington DC. We perform AC power flow with transient analysis on the system in order to identify critical nodes, whose failure (or attack) leads to the maximum damage. Our main results are summarized below.

1. Analysis of the criticality of high degree nodes. Prior studies have generally suggested that high degree nodes are most critical. When there are multiple nodes with the highest degree, we find through our analysis, that not all k size subsets of these highest degree nodes are equivalent in terms of criticality. We find that none of the high degree based attacks introduced instability in the system and created a widespread blackout.

2. Greedy strategy for finding a critical set. We examine an alternative approach for finding a good critical set by picking a node that causes the maximum additional number of tripped nodes, at each step. We find that this strategy leads to more failed nodes compared to high degree based attack. Also, we find there is a significant probability of instability in the power grid.

3. Non-monotonicity of critical sets. Our results yield a potentially interesting and counter-intuitive observation, namely that the criticality of a set can *decrease* if more nodes are added to it. Flow based models and structural models of the power grid yield monotonic behavior of critical sets.

Organization. We describe related work in Sect. 2 and our methodology in Sect. 3. The experimental design is discussed in Sect. 4 followed by discussion of some interesting results in Sect. 5. Due to space restrictions, results are presented for $k = 1, 2, 3$ in detail. Additional results for high values of k are discussed in a technical report that can be downloaded [24]. We note that in general, in practice k is usually small.

2 Related Work

A number of authors [1, 3, 6–9, 17, 26, 33] have studied cascading failures in power grids using quasi steady state analysis with DC power flow. Informally, in these papers, a transmission line fails when the power flow exceeds the capacity of the line. The power is redistributed either to the neighboring lines as in the case of sandpile like models or is redistributed by recomputing the steady state DC power flow. With reactive power component being ignored and the assumption of a flat voltage profile, the DC power flow analysis may produce good approximations under some circumstances. However, since voltage instability is an important cause for cascading events in the power grid, it is not a suitable tool to simulate such events. Here, we use AC power flow model to accurately simulate the actual operating point in the power system.

Historically relay misoperations has been instrumental in causing cascading events in power system [7]. The papers referenced above do not consider relay misoperations. A failed protection system which remains dormant in normal operating condition and becomes exposed during a disturbance causes significant cascading events resulting in blackouts [28, 29]. In this paper, we use a stochastic model of hidden failures in the transmission line relays.

Stability of power system subjected to cascading events is evaluated either from the network structure point of view (evaluating the degree distribution of nodes) [4, 5, 13, 32] or from the convergence of steady state power flow solution [1, 6, 7, 9]. However, such measures do not necessarily cover all possibilities of grid instability. The non-linear mechanisms like the rotor angle stability or voltage collapse are not accurately captured in these methods [11]. In this paper, we have used dynamic transient analysis to assess stability of the power system. The simultaneous modeling of power system dynamics and protection functions like over-current, under-voltage, over-voltage trips etc ensures that the actual operation of the power system is exactly replicated for cascading event studies.

Another important aspect of studying cascading events in the power system network is the impact of different initiating events. In [4, 5, 33], the authors simulated cascading failures in power system which have been initiated by the failure of a random node in the network. In [32], four different types of initiating events are simulated in the Chinese transportation network to replicate different types of targeted attack scenarios. Such models replicate a geographically co-related attack scenario or a targeted attack on random nodes. However, given the complex non-linear nature of the power system, it would be interesting to study the impact of choosing the attack nodes in a greedy fashion to maximize the impact. In this paper, the greedy choice is compared with the extent of damage caused by selecting highly connected nodes as the points of targeted attack.

Cascading failures in interdependent power and communication networks was studied in [5]. However the paper is based on a simplified Boolean model of system failure. In [13, 18], the authors consider a probabilistic Boolean model of cascading failure. These and the follow on papers are useful in that one can often either obtain analytical results or carry out large number of simulations to get a detailed understanding of cascade dynamics. However, the model simplification comes at a cost; the recent blackout reports [22, 30] suggest that cascades need not propagate locally due to complex non-linear nature of the power grid. In [4], the authors studied the effect of connectivity between layered networks on the cascade probability in the network. The authors used the Bak-Tang-Wiesenfeld sandpile dynamics to represent the cascade of loads in the power grids. However it fails to replicate the actual system conditions in a power grid where a node (or bus) trips due to under-voltage or under-frequency and not due to overload.

3 Preliminaries and Problem Formulation

We first describe our AC power flow simulation tool for the Washington DC power grid. This involves many components—development of a synthetic power grid of the region in and around Washington DC, including protection devices, and the models for AC power flow and transient analysis. Finally, we define the problem we study.

Synthetic Power Network Data. Accurate data on the power grid is not available publicly, because of its sensitive and proprietary nature. We have constructed the transmission and subtransmission system of the region around Washington DC based on scattered information found in: books, Internet, old maps, manual exploration on Google Earth, and open source information from utilities [10, 19–21]. These are all integrated to estimate the geographical locations of generators, substations and transmission line routes. The parameters of the system are approximately estimated based on the relative distance between the buses and expert knowledge. We give complete details of our methodology in the full version of the paper [24].

Power Flow Model. The standard power system model is basically a set of non-linear differential algebraic equations. The algebraic network power balance equations and the differential equations are solved using a partitioned approach

where the solution of the algebraic equations alternates with the solution of the differential equations [25]. The combined equations are given below.

$$\begin{aligned}\dot{\mathbf{x}} &= \mathbf{f}(\mathbf{x}, \mathbf{y}, \mathbf{p}) \\ 0 &= \mathbf{g}(\mathbf{x}, \mathbf{y}, \mathbf{p}),\end{aligned}\quad (1)$$

where $\mathbf{x} \in \mathbb{R}^n$ are the state variables, $\mathbf{y} \in \mathbb{R}^m$ are the algebraic variables; $\mathbf{p} \in \mathbb{R}^l$ are the independent variables, $\mathbf{f} : \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^l \mapsto \mathbb{R}^n$ are the differential equations, and $\mathbf{g} : \mathbb{R}^m \times \mathbb{R}^m \times \mathbb{R}^l \mapsto \mathbb{R}^m$ are the algebraic equations—these are obtained from the active and reactive power balance equations at all buses.

We consider the power system network as an undirected graph $G(V, E)$ where V denotes the set of nodes (buses) and E denotes the set of edges (transformers and transmission lines). Let $E_v \subset E$ be the set of edges connected to v . Let L_v and G_v respectively be the set of loads and generators connected to node v . The power flow equations consist of the following equalities and inequalities as shown in Eq. 2.

$$\begin{aligned}\sum_{k \in E_v} P_k + \sum_{g \in G_v} P_g - \sum_{l \in L_v} P_l &= 0, \forall v \in V \\ \sum_{k \in E_v} Q_k + \sum_{g \in G_v} Q_g - \sum_{l \in L_v} Q_l &= 0, \forall v \in V \\ Q_g^{min} \leq Q_g \leq Q_g^{max}, \forall g \in G_v, \forall v \in V \\ P_g^{min} \leq P_g \leq P_g^{max}, \forall g \in G_v, \forall v \in V \\ U_v^{min} \leq U_v \leq U_v^{max}, \forall v \in V, f_k \leq f_k^{thermal}, \forall k \in E,\end{aligned}\quad (2)$$

where P_g, Q_g are the real and reactive generation, P_l, Q_l are the real and reactive load and U_v is the voltage at node v . P_k, Q_k are the real and reactive power flow along edge k . $[Q_g^{min}, Q_g^{max}]$ and $[P_g^{min}, P_g^{max}]$ denote the reactive and real power operational limits of the g^{th} generator at the node v , $[U_v^{min}, U_v^{max}]$ is the pre-specified limit inside which all voltages must lie, and $f_k^{thermal}$ is the thermal limit of the k^{th} transmission line/transformer. Equation 2 is solved at every time step of the simulation which then accounts for any change that might have happened in the network conditions/topology due to the dynamics of the system. For this study, the Newton-Raphson (NR) method was used for solving the power flow equations and determining the voltage magnitude and angle at every bus in the network.

Transient Stability Analysis. In response to a rapid loss of load (or generation), the power system frequency will increase (or decrease). However, the generator controls respond to this change by changing the power output to meet the electric load demand based on Eq. 3.

$$P_m - P_e(\delta) = M\ddot{\delta} + D\dot{\delta} \quad (3)$$

P_m is the mechanical power input, $P_e(\delta)$ is the electrical power output as a function of the electrical angle (δ), M is a function of the machine's inertia, and D

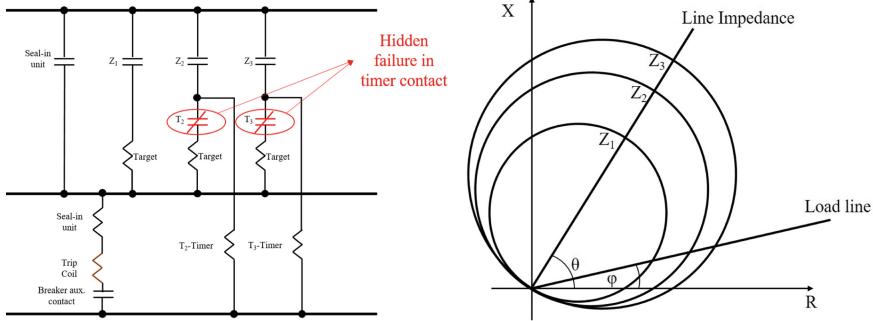


Fig. 1. Schematic diagram of a mho distance relay with three zone protection (left); R-X diagram showing operating characteristics of a mho distance relay with three zone protection (right); see [12] for details.

corresponds to the damping coefficient. Equation 3 is the governing equation for a generic transient stability analysis [14, 25]. In the present study we have followed a numerical integration method to solve the differential equations. Three states of the system are considered: (i) Pre-disturbance state: the stable equilibrium point where the system lies before the attack occurs; (ii) The disturbance state: a sudden three phase fault occurs at the targeted buses along with three phase line faults in the lines connected to them. The system stays in this state momentarily before the protective devices comes into operation; (iii) Post-disturbance state: post-attack operating point of the power system where it may return to a stable equilibrium point or might collapse altogether. If a set of generator rotor angles differ from the rotor angles of another set by more than 180 degrees, the system is considered to collapse due to instability causing widespread blackout [14].

Modeling Protection Systems. The distance protection schemes in power system play an important role in response to a disturbance. A typical mho relay distance protection consists of three zones: Zone 1 is for primary protection designed for instantaneous operation while Zone 2 and Zone 3 are back-up with inherent time delay [12]. Figure 1 represents the schematic diagram of a mho distance relay with three zone protection scheme. The inherent time delays in the operation of zones 2 and 3 is implemented through the timer contacts T_2 and T_3 and their corresponding timer coils. This means that if the apparent impedance (the impedance measured by the relay) encroaches a particular zone, the contact Z closes instantaneously; but, the trip coil is not energized if the timer contact T remains open. The timer contact T closes only after the inherent time delay.

If there is a hidden failure in the timer contacts of zones 2 or 3 (contacts are closed permanently), the trip coil is not energized until the apparent impedance encroaches any of these zones (since contact Z_2 and Z_3 are open). The failure in the timer contact remains hidden until the instant when the apparent impedance encroaches zone 2 or zone 3. This leads to an instantaneous trip in zones 2 and 3 which are unnecessary trips (since they are supposed to issue a trip signal with

an inherent delay) [28]. In this paper, the hidden failure in the transmission line relays are modeled using a probabilistic approach, wherein a set of relays with hidden failure are randomly sampled from a probability distribution.

Problem Formulation: Critical Set. We define the criticality of a set of target nodes S as the “impact” on power system, when they are attacked together; the impact can be measured in terms of many metrics, e.g., the number of nodes lost and the empirical probability of the occurrence of an unstable power swing. The empirical probability is the ratio of number of simulations which resulted in unstable power swing to the total number of simulations. The set of nodes with the maximum criticality (over all subsets of the same size) is referred to as an optimum critical set. Since finding an optimal critical set is a challenging non-linear optimization problem, we study various heuristics to find sets with high criticality.

4 Experimental Design

We will use “targeted” or “attacked” synonymously. We consider two heuristics to select the nodes to attack, and study the following: (i) The variation of criticality for highly connected nodes. (ii) The comparison of criticality of highly connected target nodes and target nodes selected through greedy strategy. (iii) The variation of criticality with cardinality of the set of target nodes.

1. High degree. We pick the top k nodes with the highest degree (also referred to as having maximal connectivity). This has been suggested as the best choice in many prior works, e.g., as discussed in [11]. The motivation behind such a choice is that if a highly connected bus experiences a fault due to a blast, all the branches connected to it are affected by the fault. In the synthetic grid of the area in and around Washington DC, there are 5 highly connected nodes in the network with each of them having a degree of 16. These are identified as nodes with Target IDs A,B,C,D and E. The voltage level of these nodes are 138, 230, 345 kV.

2. Greedy heuristic. We pick k nodes iteratively from the power system. In each iteration, the node which leads to the maximum additional number of tripped nodes is added to the target set. Due to the large number of nodes in the power grid, we have focused on the set of 500 kV high voltage substation buses to build the target set of k nodes through k iterations. This strategy is motivated by greedy algorithms, which has been shown to be very effective in some problems (see [16]). There are 31 500 kV substation buses in the power grid which are identified as nodes with Target IDs 1, 2, …, 31.

For each set of target nodes for either attack strategy, the cascading event simulation is run 10 times to account for the probabilistic occurrence of hidden failures in the transmission line relays. The mean number of nodes tripped in the 10 simulations is considered as a measure of the impact caused by the attack. The other impact can be considered as the number of times the targeted attack has resulted in an unstable power system and thereby leading to a widespread

blackout. We consider the empirical probability of blackout which denotes the fraction of simulations for which the system collapses due to instability. The system is considered to collapse when the generator rotor angles differ by more than 180 degrees.

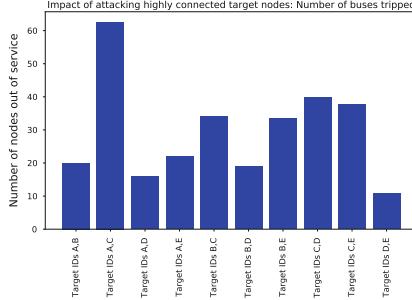


Fig. 2. Comparing criticality (number of nodes tripped) of target node sets ($k = 2$) comprising of highly connected nodes

5 Results and Discussion

1. Analysis of criticality of high degree nodes. We consider $k = 2$ in our experiments, because of the computational overheads in solving power flows. There are five nodes of the maximum degree (16), so we consider the 10 possible combinations of picking subsets of size 2 from them. Figure 2 shows the criticality of the 10 possible target node sets. The number of nodes (buses) tripped after the attack is considered as the measure of criticality. We observe that not all pairs are equivalent, and there is a wide variation in the number of out-of-service (tripped) nodes for each set of target nodes. The maximum number of node outages (62)

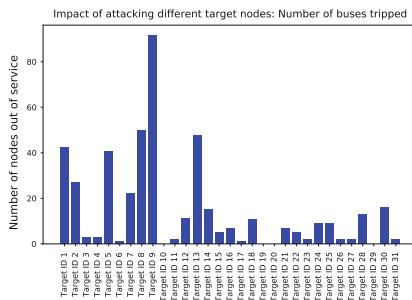


Fig. 3. Comparing criticality (number of nodes tripped) of target node sets ($k = 1$) comprising of 500 kV nodes selected through greedy strategy

occurs when nodes A and C are attacked. We also observed that none of these attacks introduced instability in the system and created a blackout. Therefore, the comparison of this criticality measure is not shown.

2. Analysis of the greedy heuristic. In this section, we consider choice of target nodes based on the greedy strategy. Figure 3 shows the comparison of criticality of the 500 kV node sets when a single node is attacked ($k = 1$). We observe that when node 18 is selected as the target of attack, it results in the maximum number of node outages (110). We also observe that none of these target sets results in an instability in the power system. Therefore, this measure of criticality is not shown here.

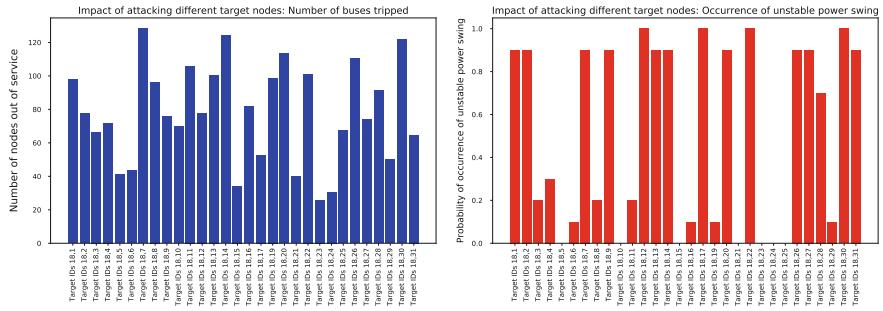


Fig. 4. Comparing criticality of target node sets ($k = 2$) comprising of 500 kV nodes selected through greedy strategy

We now consider the second iteration of the greedy strategy where we identify the 500 kV node which results in the maximum additional criticality. Node 18 is selected as one of the elements in the target sets for $k = 2$ since it has the maximum criticality when attacked singularly. Therefore, the combination of node 18 with remaining 30 nodes are considered as the possible target sets for $k = 2$. Figure 4(left) compares the number of tripped nodes as the measure of criticality for the target node sets when two nodes are attacked simultaneously ($k = 2$). In this case, multiple simulations resulted in an unstable power swing leading to blackouts. Figure 4(right) compares the empirical probability of the occurrence of a blackout due to instability as another measure of criticality of the target node sets. Comparing criticality of target sets in Figs. 2, 4, we note that target node sets chosen through greedy strategy have higher criticality than target sets of highly connected nodes.

From Fig. 4 we observe that the target set comprising of nodes 18 and 30 has the maximum criticality. Therefore, for the third iteration we choose these nodes as the common elements for the target sets. We analyze the criticality of combining these nodes with each of the other target nodes in Fig. 5. More detailed results of criticality of target node sets are currently studied and updated in [24].

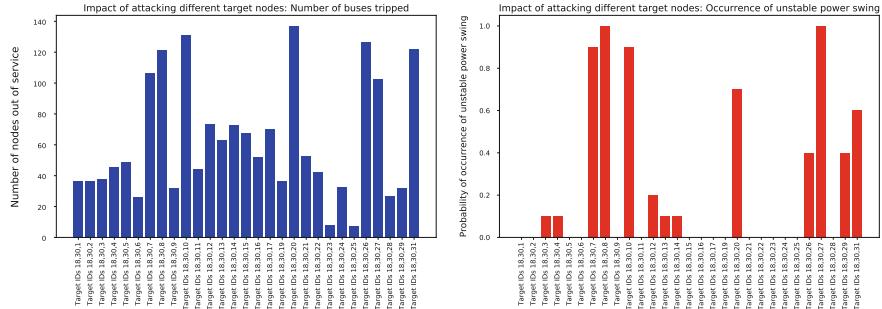


Fig. 5. Comparing criticality of target node sets ($k = 3$) comprising of 500 kV nodes selected through greedy strategy

3. Non-monotonicity of criticality. From the statistics in Figs. 3 and 4, a number of interesting observations can be made. Though node 18, when targeted singularly has resulted 110 node outages, it caused fewer outages when combined with the nodes 5, 15, 21 or 24. The main reason for such counter-intuitive behavior is the transient stability analysis which has been considered to depict the operation of a realistic power system. A power grid is said to be transient stable if it is able to maintain synchronism when subjected to a disturbance [14]. Essentially, this is achieved if the real and reactive power generation in the power system matches the real and reactive power load demands. If the target nodes are selected in such a way that results in tripping of equal amounts of load and generation, the remnant power system might reach a condition where the real/reactive power generation matches the load. For example, we consider three results of cascading events in the synthetic power grid of Washington DC in Fig. 6. In the first case only target node 18 is attacked, while in the second case target node 23 is attacked alongside 18 and in the third case target nodes 18 and 30 are attacked together. These are the black nodes in the network since they are the nodes which fail at the earliest. The color of the nodes and edges indicate their mean time of failure over 10 simulations. In the first case, when a single node (target node 18) is attacked, it results in a number of trips on the neighboring lines. However, the buses with heavy load remains connected to the grid causing overload on the in-service transmission lines in the network. Some of the relays on these lines may trip due to presence of hidden failures or because of the apparent impedance encroaching Zone-1 of the mho characteristics. Particularly, the tripping of edges in the yellow ellipse is shortly followed by a number of generator trips in the red ellipses. This is because a loss of important transmission lines causes a voltage collapse in the system which results the generators to deliver reactive power above its capacity, thereby leading to over-excitation protection trips. In other cases, the voltage collapse causes under-voltage protection system of generators to trip.

However, in the second case when target nodes 18 and 23 are simultaneously attacked, a number of heavy loads (in the vicinity of target node 23) are removed

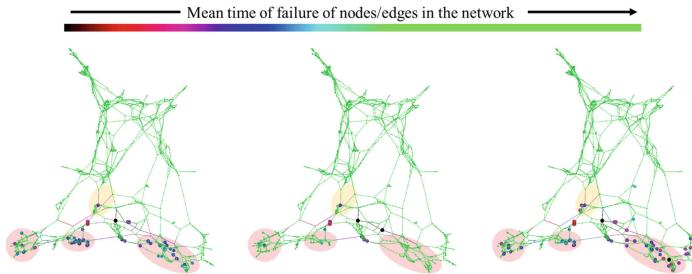


Fig. 6. Comparing the effect of three different targeted attacks on the power system. The left figure shows widespread failure of nodes when only target node 18 is attacked. The middle figure shows a stable power grid when target nodes 18 and 23 are attacked simultaneously. The right figure shows an unstable power grid when target nodes 18 and 30 are attacked together.

from the system which does not overload the lines in the yellow ellipse. Therefore, the system is saved from a voltage collapse and generators do not trip in the red ellipses. Within a few seconds, the system is stabilized. In the third case, the selected targets results in removal of sufficient transmission lines but heavy loads remain connected. This creates a similar situation as in the first case and leads to tripping of lines in the yellow ellipse. This is shortly followed by the generator trips in the red ellipse and system becomes unstable due to loss of large number of generators.

6 Summary and Future Work

We studied the resilience of the power grid to targeted attacks. In contrast to earlier work, a realistic representation of the power grid and its constituent elements was used to study the problem. Two simple strategies were used: a node degrees based selection and a greedy strategy resulting in maximum criticality.

We found that a targeted attack on the highly connected nodes of the power grid causes a large number of node outages, yet the system is stabilized eventually. On the other hand, a targeted attack on certain selected high voltage nodes can cause blackouts in the grid due to instability. Furthermore, it is observed that increasing the number of targets does not necessarily increase the number of node outages. Such conclusions indicate that cascading events in the power grid cannot be simply modeled using standard failure models like the sandpile dynamics model. The role of protective elements and the generator control systems need to be effectively modeled to analyze the cascading events accurately.

The non-monotonicity of criticality indicates that the greedy strategy does not necessarily ensure the optimal set of k target nodes to maximize the attack. We feel that an adaptive greedy strategy of choosing nodes might generate the optimal set of target nodes; a detailed study of such a strategy is one direction for future work. The impact of a hybrid choice comprising of the greedy and

degree based strategies can be studied. Another important aspect of cascading events in the power system is the order of transition as discussed in Pahwa et al. [17]. We leave this as a topic for future work.

Acknowledgements. We thank the reviewers for their constructive comments. We also thank members of the Network Dynamics and Simulation Science Laboratory at Virginia Tech and Professor Arun Phadke and late Professor James Thorp for their collaboration and their insightful suggestions on topics related to the paper. This work has been partially supported by DTRA CNIMS (Contract HDTRA1-11-D-0016-0001), NSF DIBBS Grant ACI-1443054, NSF BIG DATA Grant IIS-1633028, NSF EAGER Grant CMMI-1745207 and DOE Grant DE-EE0007660.

References

1. Bae, K., Thorp, J.S.: A stochastic study of hidden failures in power system protection. *J. Decis. Support. Syst.* **24**(3), 259–268 (1999)
2. Barrett, C.L., Eubank, S., Evrenosoglu, C.Y., Marathe, A., Marathe, M.V., Phadke, A., Thorp, J., Vullikanti, A.: Effects of hypothetical improvised nuclear detonation on the electrical infrastructure. In: International ETG-Congress 2013; Symposium 1: Security in Critical Infrastructures Today, vol. 1, pp. 1–7 (2013)
3. Bernstein, A., Bienstock, D., Hay, D., Uzunoglu, M., Zussman, G.: Sensitivity analysis of the power grid vulnerability to large-scale cascading failures. *SIGMETRICS Perform. Eval. Rev.* **40**(3), 33–37 (2012)
4. Brummitt, C.D., D’Souza, R.M., Leicht, E.A.: Suppressing cascades of load in interdependent networks. *Proc. Natl. Acad. Sci.* **109**(12), E680–E689 (2012)
5. Buldyrev, S.V., Parshani, R., Paul, G., Stanley, H.E., Havlin, S.: Catastrophic cascade of failures in interdependent networks. *Nature* **464**(1), 1025–1028 (2010)
6. Carreras, B.A., Lynch, V.E., Dobson, I., Newman, D.E.: Dynamics, criticality and self-organization in a model for blackouts in power transmission systems. In: Proceedings of the 35th Annual Hawaii International Conference on System Sciences (2002)
7. Chen, J., Thorp, J.S., Dobson, I.: Cascading dynamics and mitigation assessment in power system disturbances via a hidden failure model. *Int. J. Electr. Power Energy Syst.* **27**(4), 318–326 (2005)
8. Chertkov, M., Pan, F., Stepanov, M.G.: Predicting failures in power grids: the case of static overloads. *IEEE Trans. Smart Grid* **2**(1), 162–172 (2011)
9. Dobson, I., Chen, J., Thorp, J.S., Carreras, B.A., Newman, D.E.: Examining criticality of blackouts in power system models with cascading events. In: Proceedings of the 35th Annual Hawaii International Conference on System Sciences (2002)
10. Dominion Virginia Power: Arlington Area Upgrade Fact Sheet (2012)
11. Hines, P., Cotilla-Sanchez, E., Blumsack, S.: Do topological models provide good information about electricity infrastructure vulnerability? *Chaos: Interdisciplinary J. Nonlinear Sci.* **20**(3), 033,122 1–5 (2010)
12. Horowitz, S.H., Phadke, A.G.: Power System Relaying, 2nd edn. Research Studies Press, Taunton, UK (1995)
13. Huang, X., Gao, J., Buldyrev, S.V., Havlin, S., Stanley, H.E.: Robustness of inter-dependent networks under targeted attack. *Phys. Rev. E* **83**(6), 065,101 1–4 (2011)
14. Kundur, P.S.: Power System Stability and Control. McGraw Hill Education, New York (1994)

15. National Academies of Sciences: Engineering, and Medicine.: Analytic Research Foundations for the Next-Generation Electric Grid. The National Academies Press, Washington, DC (2016)
16. Nemhauser, G.L., Wolsey, L.A., Fisher, M.L.: An analysis of approximations for maximizing submodular set functions. *Math. Program.* **14**(1), 265–294 (1978)
17. Pahwa, S., Scoglio, C., Scala, A.: Abruptness of cascade failures in power grids. *Sci. Rep.* **4**(1), 3694 1–9 (2014)
18. Parshani, R., Buldyrev, S.V., Havlin, S.: Interdependent networks: Reducing the coupling strength leads to a change from a first to second order percolation transition. *Phys. Rev. Lett.* **105**(4), 048,701 1–4 (2010)
19. PEPCO: Comprehensive Reliability Plan for District of Columbia (2010)
20. PJM: PJM Manual 03: Transmission Operations (2011)
21. PJM: PJM Transmission Providers Facilities List (2012)
22. Pourbeik, P., Kundur, P.S., Taylor, C.W.: The anatomy of a power grid blackout - root causes and dynamics of recent major blackouts. *IEEE Power Energy Mag.* **4**(5), 22–29 (2006)
23. Romero, J.J.: Blackouts illuminate India’s power problems. *IEEE Spectr.* **49**(10), 11–12 (2012)
24. Meyur, R.: Cascading effects of targeted attacks on the power grid. www.ece.vt.edu/~rounak/cascade.pdf (2018)
25. Sauer, P.W., Pai, M.A.: Power System Dynamics and Stability. Prentice Hall, Upper Saddle River (1998)
26. Soltan, S., Mazauric, D., Zussman, G.: Cascading failures in power grids: analysis and algorithms. In: Proceedings of the 5th International Conference on Future Energy Systems, e-Energy 2014, pp. 195–206. ACM, New York (2014)
27. Sugarman, R.: Power/energy: New York City’s blackout. *IEEE Spectr.* **15**(11), 44–46 (1978)
28. Tamronglak, S., Horowitz, S.H., Phadke, A.G., Thorp, J.S.: Anatomy of power system blackouts: preventive relaying strategies. *IEEE Trans. Power Deliv.* **11**(2), 708–715 (1996)
29. Thorp, J.S., Phadke, A.G., Horowitz, S.H., Tamronglak, S.: Anatomy of power system disturbances: importance sampling. *Int. J. Electr. Power & Energy Syst.* **20**(2), 147–152 (1998)
30. US-Canada Power System Outage Task Force: Final Report on the August 14, 2003 Blackout in the United States and Canada: Causes and Recommendations (2004)
31. Vassell, G.S.: Northeast blackout of 1965. *IEEE Power Eng. Rev.* **11**(1), 4–8 (1991)
32. Wang, W., Yang, S., Hu, F., Stanley, H.E., He, S., Shi, M.: An approach for cascading effects within critical infrastructure systems. *Phys. A Stat. Mech. Its Appl.* **510**(1), 164–177 (2018)
33. Zhang, Y., Yağan, O.: Optimizing the robustness of electrical power systems against cascading failures. *Sci. Rep.* **6**, 27,635 1–15 (2016)

Community Structure



A Memory-Based Label Propagation Algorithm for Community Detection

Antonio Maria Fiscarelli^{1,2(✉)}, Matthias R. Brust², Grégoire Danoy³,
and Pascal Bouvry^{2,3}

¹ C2DH, University of Luxembourg, 11 Porte des Sciences, Esch-sur-Alzette,
Luxembourg

{antonio.fiscarelli,matthias.brust}@uni.lu

² SnT, University of Luxembourg, 6 avenue de la Fonte, Esch-sur-Alzette,
Luxembourg

³ FSTC-CSC-ILIAS, University of Luxembourg, 6 avenue de la Fonte,
Esch-sur-Alzette, Luxembourg

{gregoire.danoy,pascal.bouvry}@uni.lu

Abstract. The objective of a community detection algorithm is to group similar nodes in a network into communities, while increasing the dissimilarity between them. Several methods have been proposed but many of them are not suitable for large-scale networks because they have high complexity and use global knowledge. The Label Propagation Algorithm (LPA) assigns a unique label to every node and propagates the labels locally, while applying the majority rule to reach a consensus. Nodes which share the same label are then grouped into communities. Although LPA excels with near linear execution time, it gets easily stuck in local optima and often returns a single giant community. To overcome these problems we propose MemLPA, a novel LPA where each node implements memory and the decision rule takes past states of the network into account. We demonstrate through extensive experiments on the Lancichinetti-Fortunato-Radicchi benchmark and a set of real-world networks that MemLPA outperforms most of state-of-the-art community detection algorithms.

Keywords: Network analysis · Graph theory · Community detection
Label propagation

1 Introduction

Real-world networks often exhibit a community structure where nodes in a community are highly connected and few links connect the different communities. The objective of a community detection algorithm is to partition similar nodes of the network in groups, while increasing the dissimilarity between the groups. This problem is NP-hard [4], therefore it is important that community detection algorithms maintain a low complexity while possessing high scalability. Due to

growing interest, community detection has attracted many researchers from different areas such as computer science [1], natural sciences [11] and social sciences [23], making it a notably active research field.

Several community detection methods have been proposed in the literature such as greedy algorithms that incorporate modularity optimization [3, 5, 9, 16], spectral methods based on modularity matrix [15] and random walk-based methods [18, 22]. In particular, the Label Propagation Algorithm (LPA) assigns a unique label to every node and propagates the labels locally, while applying the majority rule to reach a consensus. Nodes sharing the same label are then grouped into communities. This method runs in near linear time, is scalable and requires only local information, thus it is especially suitable for large networks. On the other hand, it gets easily stuck in local optima and is thus outperformed by more sophisticated algorithms.

The method that we propose is called MemLPA, a variation of the classic LPA where each node implements memory and the decision rule takes past states of the network into account. We also adapt some of the improvements proposed in the literature to our method such as neighborhood preference and termination criterion based on active nodes. The use of memory improves performance and prevents a single label from flooding the network. We conducted extensive experiments on the Lancichinetti-Fortunato-Radicchi benchmark and a set of real-world networks. The algorithm is tested against state-of-the-art community detection algorithms and it outperforms most of them for values between 0.5 and 0.8 of the mixing parameter.

The remainder of this article is organized as follows. Section 2 presents a state-of-the-art analysis on community detection algorithms and label propagation algorithms. Our contribution, MemLPA, is introduced in Sect. 3 and its performance is analyzed and compared to other community detection algorithms on artificial and real-world networks in Sect. 4. Finally, Sect. 5 provides our conclusions.

2 Related Work

In this Section we provide an overview of community detection algorithms that do not require a priori information about the network, such as the number of communities. In particular, we discuss LPA and several variations that have been proposed in the literature.

Girvan and Newman [16] proposed a divisive hierarchical algorithm that removes edges with high betweenness to enhance the separation of communities. This algorithm runs in $O(nm^2)$. A faster version was proposed [5] that iteratively merges nodes into communities to optimize modularity. The complexity for this method is $O(md \log n)$, where d is the depth of the dendrogram. Blondel, Guillaume, and Lambiotte [3] developed a similar method called Louvain. It repeatedly merges nodes into communities that achieve the highest modularity improvement and builds a new network where nodes represent the communities found. This method runs in $O(n \log n)$. Walktrap [18] defines a similarity

between nodes according to the transition probability of random walkers. In fact, random walkers are more likely to stay within the same community. It runs in $O(n^2m)$ or $O(n^2 \log n)$ on sparse networks. Infomap [22], similarly, models flow patterns in networks using the transition probability of random walkers and runs in $O(m)$. Newman [15] proposed a spectral method based on the Eigen-spectrum of the modularity matrix that runs in $O(n(m+n))$ or $O(n^2)$ on sparse networks. Finally, Reichardt and Bornhol [21] interpreted community detection as the minimization of the energy function of a spin model. It runs in $O(n^{3.2})$ on sparse networks.

Many of the algorithms proposed in the literature have downfalls that make them inefficient on large-scale networks: they have high complexity and require global information of the network. To overcome this problem, Raghavan [19] proposed the Label Propagation Algorithm (LPA): it runs in near linear time, is scalable and uses the network's local information only, without the need of optimizing any objective function. It assigns a unique label to every node and propagates the labels locally, while applying the majority rule to reach a consensus. Nodes sharing the same label are then grouped into communities. On the other hand, LPA gets easily stuck in local optima and is thus outperformed by more recent and sophisticated algorithms. Also, a certain label may "flood" the network and create a single giant community. Several variations have been proposed. Clark [2] developed a variation that includes modularity optimization to improve performance. This method was further enhanced by Liu and Murata [14] with a greedy method that allows it to escape from local optima. Leung [13] introduced a decision rule based on node preference, such as node degree, to improve performance and hop attenuation to prevent a label from flooding the network. The algorithm is scalable and still runs in near linear time. Xie and Szymanski [27] proposed another node preference, based on neighborhood similarity, that is related to the clustering coefficient. Šubelj and Bajec [24] elaborated two particular strategies, defensive preservation and offensive expansion, that adapt node preference to focus on core nodes and border nodes of communities. They also found that the network structure affects the effectiveness of node preference and hop attenuation. The algorithm runs in $O(m^{1.19})$ and is highly scalable. Xie and Szymanski [26] also developed LabelRank, a variation of the classic LPA that takes inspiration from the MCL algorithm [8]. Each node maintains a list of label distributions that are propagated through the network. An inflation and a cutoff operator are applied to shorten these lists.

3 MemLPA: A Memory-Based Label Propagation Algorithm

The classic LPA updates nodes' labels according to the current state of the network. During a certain iteration, each node collects its neighbors' labels and selects the most chosen one. At each iteration, all these labels are discarded and new ones are collected. This mechanism makes the algorithm memory-less, since it does not consider past states. In this Section we introduce MemLPA, a

variation of the classic LPA where nodes implement memory: the use of memory increases performance with limited increase in complexity and without affecting scalability.

3.1 Algorithm Description

When using memory, labels are not discarded but updated at each iteration. Each node maintains a list of labels where every element of the list contains a counter associated to that label. Initially, each node is assigned a unique label (line 3 of the pseudocode) and their label lists are empty (line 4). At each iteration, each node collects its neighbors' labels (line 8) and updates its label list according to weight (for weighted networks) and node preference (line 9). If a new label appears, a new entry is created in the list, otherwise the counter for the corresponding label is incremented. Each node then selects a label from the list using a decision rule that takes into account the labels' counters, the maximum element in this case (line 11). This mechanism can be applied to directed or undirected as well as to weighted or unweighted graphs. Figure 1 shows how MemLPA works.

In order to keep MemLPA scalable, we propose a synchronous update rule: each node independently updates its state according to the network's state during the previous iteration. It has been shown that a synchronous update can cause LPA to oscillate between two different configurations and an asynchronous update can lead to better results, but in Sect. 4 we show how the two different update rules affect the convergence of MemLPA. As node preference, we use a heuristic based on neighborhood similarity: we compute the percentage of neighbors that a node shares with another. In Sect. 4 we show the impact of this heuristic on performance. To speed up the algorithm, we define a cutoff operator to prune label lists (line 10). At each iteration, all labels below a certain threshold are deleted, keeping only the most relevant ones. Regarding the termination criterion, several options have been proposed in the literature that are based on convergence, modularity improvement, active nodes and scarcity of updates. Modularity is based on global information of the network, therefore we decided to use a termination criterion based on the active node list: a node is considered active if, in an attempt to update its label, it chooses a different label. The active node list initially contains all nodes (line 1) and at each iteration a node is removed if it is no longer active or it is added if any of its neighbors becomes active again (line 13). This keeps the algorithm decentralised and speeds up the algorithm significantly. In Sect. 4 we show that using an active node list allows the algorithm to stop right after NMI or modularity has reached an optimal value.

To our knowledge, the only method that explicitly refers to memory in LPA is the Speaker-Lister Label Propagation Algorithm (SLPA) [28]. We would like to point out how our method is different: in SLPA, each node selects only one label from the collection of labels received from its neighbors, while nodes in MemLPA update their label lists with all the labels received. SLPA uses an asynchronous

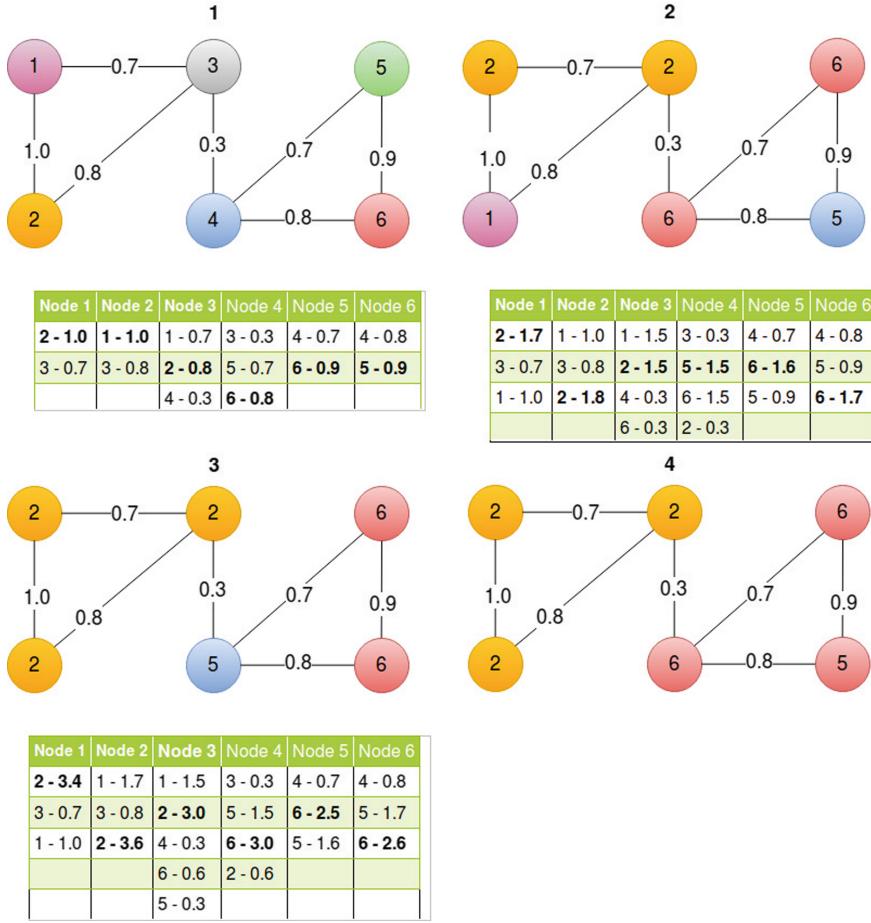


Fig. 1. Iterations of MemLPA on a weighted undirected graph. Color and node number represent labels. Columns in a table represents label lists maintained by a node.

update rule, while our method is synchronous. SLPA uses a thresholding procedure that is performed on label distributions once the algorithm converges, while MemLPA applies a cutoff operator during each iteration. Finally, SLPA terminates after a fixed number of iterations, while MemLPA's termination criterion is based on active nodes. Some work on consensus dynamics also refers to memory. For example, a non-deterministic version of the Naming Game [20, 25], which is similar in some aspects to LPA, extends the agents with local memory but also uses a shared memory, making it not decentralized, which is the main difference to our approach.

3.2 Complexity

The complexity of MemLPA on a certain node, where k is the average degree and h is the average label list length, can be assessed this way:

- Computing the neighborhood intersection for node preference with k neighbors has complexity $O(k * k)$. Notice that it only needs to be computed once and nodes can store this information.
- Updating the label list with k new values has complexity $O(k)$.
- Using the cutoff operator has complexity $O(h)$.
- Choosing a new label has complexity $O(h)$.

Algorithm 1: MemLPA

```

Input : Graph G(N, E)
Output: Communities C
1  $AL \leftarrow N$  //Initialize active list
2 for  $n \in N$  do
3    $c_n \leftarrow l_n$  //Assign unique label to nodes
4    $L_n \leftarrow \emptyset$  //Initialize label lists
5 end
6 while  $AL \neq \emptyset$  do
7   for  $n \in AL$  do
8      $C_n \leftarrow CollectLabels(Neigh(n))$  ;
9      $L_n \leftarrow UpdateLabelList(C_n)$  ;
10     $L_n \leftarrow \{l_n^m \in L_n, m \in N \mid |mean(L_n) - sd(L_n)| \leq l_n^m\}$ 
11     $c_n \leftarrow ApplyRule(L_n)$  ;
12  end
13   $AL \leftarrow UpdateActiveList(AL)$ 
14 end

```

In Sect. 4 we show how the cutoff operator keeps the average list length constant and significantly lower than the average degree. Iterating on all nodes, the overall complexity is $O(k^2 * n)$ or $O(k * m)$, instead of $O(m)$ of the classic LPA. This means that the complexity of MemLPA is still near linear w.r.t. network size.

4 Performance Study

We implemented MemLPA and assessed the use of memory and some of the variations proposed in the literature. We then compared it to other state-of-the-art community detection algorithms to show how it outperforms most of them. We also ran MemLPA to study some of its characteristics that are important for convergence. For the analysis we ran all algorithms on the Lancichinetti-Fortunato-Radicchi (LFR) benchmark [12], an established benchmark in the literature for community detection that allows to generate networks with properties similar to

real-world networks. As performance metrics, we used the Normalized Mutual Information (NMI) [7, 29] and Adjusted Rand Index (ARI) [10]. We also applied these algorithms on a set of real-world networks of different nature and used the modularity measure to evaluate the quality of the solutions.

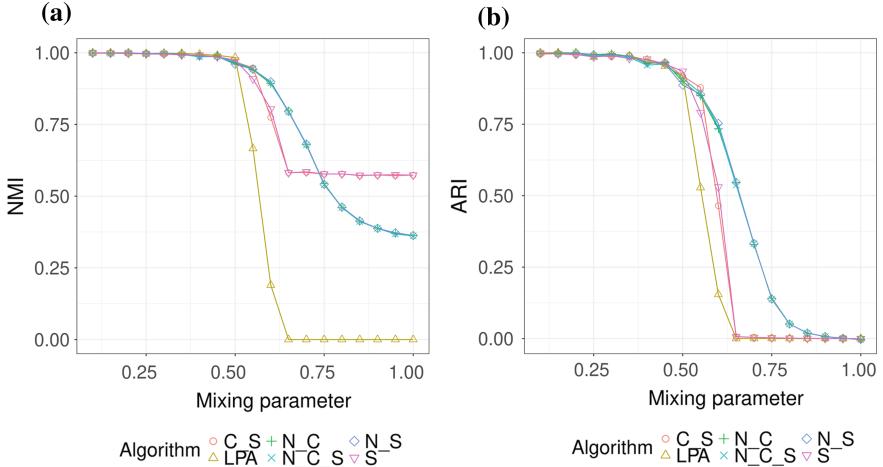


Fig. 2. Experiments on the LFR benchmark. All variations, except the basic LPA, implement memory. N: node preference, C: cutoff operator, S: synchronous update. Experiments are run 20 times and results averaged.

4.1 Artificial Networks

The first set of experiments was conducted on the Lancichinetti-Fortunato-Radicchi (LFR) benchmark to investigate the advantages of the LPA variations chosen and the use of memory. A mixing parameter μ controls the portion of intra-community and inter-community edges. Node degree and community size are not fixed but drawn from a power-law distribution. Benchmark graphs were generated with number of nodes $N = 1000$, minimum community size $C.\min = 10$, maximum community size $C.\max = 50$, average degree $K.\avg = 20$, maximum degree $K.\max = 50$, degree exponent $K.\exp = 2$ and community size exponent $C.\exp = 1$, while μ was dynamically changed. We compared the classic LPA to different variations of MemLPA that use synchronous (S) and asynchronous update rule, with and without node preference (N), with and without cutoff operator (C). Figure 2 shows that using a synchronous or asynchronous update does not make a significant change in performance (N_C_S vs N_C). Using the cutoff operator does not degrade performance either (C_S vs S and N_C_S vs N_S). This shows that MemLPA can be kept decentralized and scalable without compromising performance. For low values of μ all variations obtained optimal results. The classic version of LPA, the only one not using memory, was the first algorithm to drop in performance for $\mu \geq 0.5$ because, for these networks, a

single label overpropagated and created a single giant community. This confirms that the use of memory improves performance and prevents a label from flooding the network. For $\mu \in [0.5, 0.7]$ the variations that use node preference (N_S, N_C and N_C_S) obtained the best results, but it is not the case for higher values. In fact, the variations that did not use node preference (S and C_S) obtained higher constant values of NMI for $\mu \in [0.7, 1]$. We must consider that the NMI is dependent on network size and number of communities. Therefore we decided to use the Adjusted Rand Index to have a more accurate comparison. As a result, we can see that node preference was dominant for any $\mu \geq 0.5$.

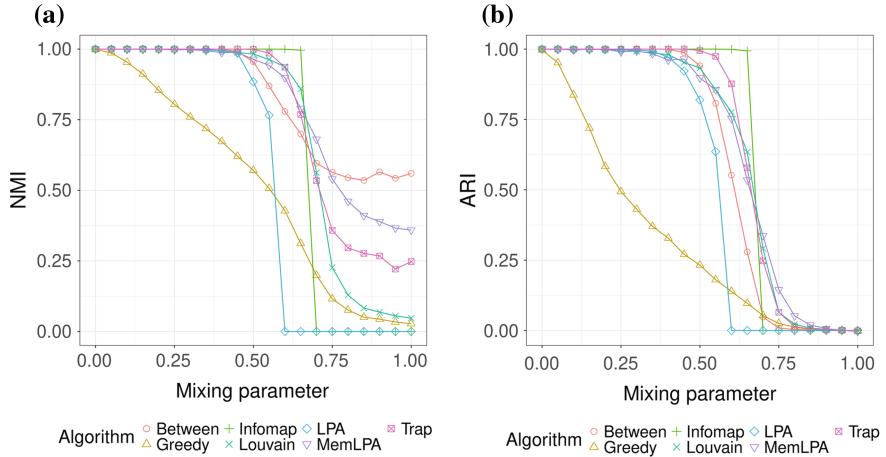


Fig. 3. Experiments on the LFR benchmark. Experiments are run 20 times and results averaged.

The same experiment was performed to compare MemLPA to other state-of-the-art community detection algorithms. We chose the algorithms described in Sect. 2 and they are all available in the igraph package [6]. Notice that Eigen was not used because it did not converge on some networks, while Spinglass cannot work with unconnected graphs.

Figure 3 shows that, for low values of μ , most algorithms obtained optimal results, while Greedy gradually decreased in performance. For $\mu \in [0.5, 0.7]$ most of the algorithms started degrading in performance, especially LPA and Between. MemLPA, in this range, was only outperformed by Infomap and Trap. For $\mu \geq 0.7$ MemLPA was the best algorithm after Between but, considering ARI, MemLPA performed slightly better until all algorithms' performance dropped to zero.

We also conducted two experiments to analyze some of the characteristics of MemLPA at runtime. We used $\mu = 0.1$ to generate networks where communities are very well defined and $\mu = 0.6$ for loose communities. As performance measures we recorded NMI, modularity and the ratio between number of communities found by MemLPA and real communities. The information that we recorded is the percentage of runs that terminated, the number of active nodes

and the average ratio between label list length and node degree. Figure 4 shows that, for $\mu = 0.1$, MemLPA increased in performance quickly, being able to find the correct number of communities. The percentage of active nodes dropped significantly right after the best performance was reached, causing most of the runs to terminate. The length of label lists, w.r.t. node degree, dropped significantly during the first iterations and then stabilized. For $\mu = 0.6$, as expected, there was a similar behavior but the algorithm converged slower. Surprisingly, the average list length is lower for $\mu = 0.6$. A possible explanation is that nodes in well defined communities hold very strong labels in their label lists, while for loose communities labels are weaker and more likely to be removed by the cutoff operator.

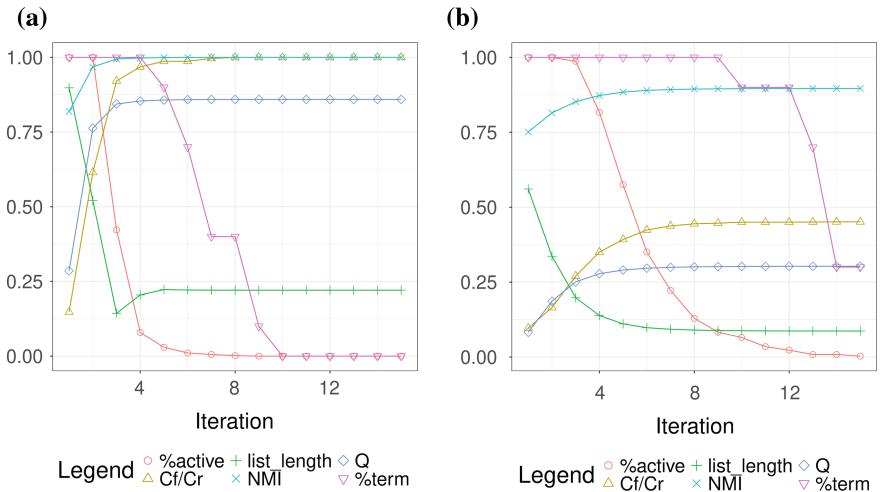


Fig. 4. Experiments on the LFR benchmark. $\mu = 0.1$ and $\mu = 0.6$ has been used for the two experiments. Both experiments have been run 50 times and results averaged.

4.2 Real-World Networks

We conducted similar experiments on a set of real-world networks of different nature. An overview of all the networks is available in Table 1.

Table 1. Real-world networks characteristics

	#nodes	#edges	Directed	Weighted
Karate	34	78	No	Yes
UKfaculty	81	817	Yes	Yes
Mail	184	2116	Yes	No
Dolphins	62	159	Yes	No
Jazz	198	2742	Yes	No
USAirports	755	23473	Yes	Yes

In the first experiment, like Sect. 4.1 for artificial networks, we investigated the advantages of memory and the LPA variations chosen. Figure 5 shows that, as previously seen on artificial networks, using a synchronous or asynchronous update did not make a significant change (N_C_S vs N_C) and the cutoff operator did not degrade performance (C_S vs S and N_C_S vs N_S). This allows MemLPA to be scalable, fast and performing. Node preference did not affect performance on unweighted networks significantly, while performance mostly degraded for the weighted ones (N_S vs S and N_C_S vs C_S). A possible explanation is that weight is a more significant factor than neighborhood overlapping when it comes to measuring the similarity or the level of nodes' interaction. Additionally, other types of heuristics might be more effective, such as node degree. Implementing memory was beneficial on all networks when compared to the memory-less LPA. In particular, it prevented labels from overpropagating on the *Mail* network where the classic LPA finds a single huge community. In the second experiment we compared MemLPA to state-of-the-art community detection algorithms. MemLPA was among the most performing algorithms on all networks. In particular, it obtained the best result for *Karate* and *UKFaculty*.

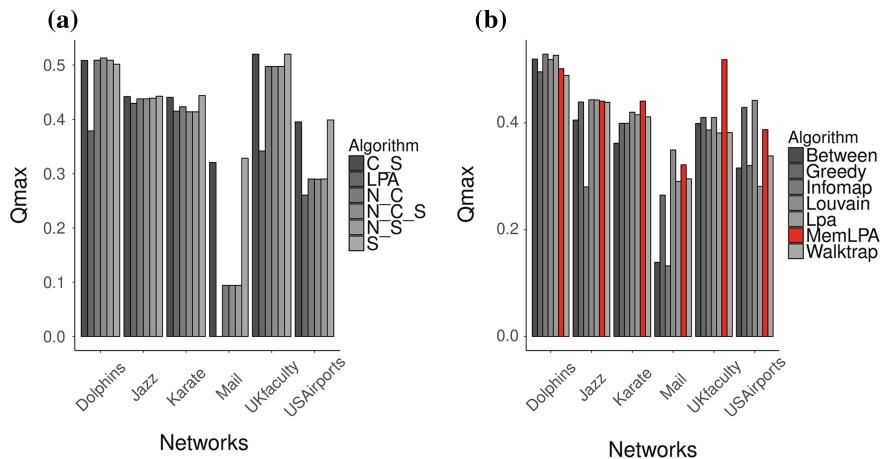


Fig. 5. Experiments on real-world networks. Each barplot represents the results obtained for all algorithms on a specific network. Experiments have been run 100 times.

4.3 Discussion

We conducted extensive experiments on the LFR benchmark, investigated the advantages of MemLPA and found that memory improves performance and prevents labels from overpropagating. We compared MemLPA to several state-of-the-art community detection algorithms: it outperforms most of them for values of the mixing parameter between 0.5 and 0.8, while still running in near linear time and using only local information. Then, we conducted two experiments

to analyze some of its characteristics at runtime, such as number of iterations before termination, number of active nodes and label list length, using NMI, modularity and number of communities found as performance metrics. We also ran MemLPA on a set of real-world networks of different nature and compared the quality of the solutions found to the ones of all the algorithms considered. With these findings, we can conclude that the use of memory is beneficial and the algorithm is competitive.

5 Conclusions and Future Work

In this paper we proposed MemLPA, a variation of LPA where nodes are seen as agents that interact with their neighbors, implement memory and use a decision rule based on past states of the network. It runs in near linear time, only uses local information of the network and is scalable. We gave an overview on community detection algorithms, LPA and the variations proposed in the literature. We investigated the advantages of memory and we found that its usage prevents labels from overpropagating and increases performance. We conducted extensive experiments on the Lancichinetti-Fortunato-Radicchi benchmark and used Normalized Mutual Information and Adjusted Rand Index as performance metrics. We compared MemLPA to state-of-the-art community detection algorithms to show how it outperforms most of them, especially for values of the mixing parameter between 0.5 and 0.8. We also conducted experiments on a set of real-world networks of different nature and evaluated the quality of the solutions found using the modularity measure. For future work, we plan on implementing other LPA variations and compare them in order to improve MemLPA. We also want to investigate the correlation between network structure and the use of memory. Finally, as suggested in [17], we want to combine topological properties with the clustering metrics already used for a better comparison of the different variations and understanding of community structure.

Acknowledgement. This work is partially funded by the joint research programme UL/SnT-ILNAS on Digital Trust for Smart-ICT.

References

1. Albert, R., Jeong, H., Barabási, A.L.: Internet: Diameter of the world-wide web. *Nature* **401**(6749), 130–131 (1999)
2. Barber, M.J., Clark, J.W.: Detecting network communities by propagating labels under constraints. *Phys. Rev. E* **80**(2) (2009)
3. Blondel, V.D., Guillaume, J.L., Lambiotte, R., Lefebvre, E.: Fast unfolding of communities in large networks. *J. Stat. Mech.: Theory Exp.* **2008**(10) (2008)
4. Brandes, U., et al.: On modularity clustering. *IEEE Trans. Knowl. Data Eng.* **20**(2), 172–188 (2008)
5. Clauset, A., Newman, M.E., Moore, C.: Finding community structure in very large networks. *Phys. Rev. E* **70**(6) (2004)

6. Csardi, G., Nepusz, T.: The igraph software package for complex network research. *InterJournal Complex Syst.* (2006). <http://igraph.org>
7. Danon, L., Diaz-Guilera, A., Duch, J., Arenas, A.: Comparing community structure identification. *J. Stat. Mech.: Theory Exp.* **2005**(09), P09008 (2005)
8. Dongen, S.: A Cluster Algorithm for Graphs (2000)
9. Girvan, M., Newman, M.E.: Community structure in social and biological networks. *Proc. Natl. Acad. Sci.* **99**(12), 7821–7826 (2002)
10. Hubert, L., Arabie, P.: Comparing partitions. *J. Classif.* **2**(1), 193–218 (1985)
11. Jeong, H., Tombor, B., Albert, R., Oltvai, Z.N., Barabási, A.L.: The large-scale organization of metabolic networks. *Nature* **407**(6804), 651–654 (2000)
12. Lancichinetti, A., Fortunato, S., Radicchi, F.: Benchmark graphs for testing community detection algorithms. *Phys. Rev. E* **78**(4), 046110 (2008)
13. Leung, I.X., Hui, P., Lio, P., Crowcroft, J.: Towards real-time community detection in large networks. *Phys. Rev. E* **79**(6), 066107 (2009)
14. Liu, X., Murata, T.: Advanced modularity-specialized label propagation algorithm for detecting communities in networks. *Phys. A Stat. Mech.* **389**(7), 1493–1500 (2010)
15. Newman, M.E.: Finding community structure in networks using the eigenvectors of matrices. *Phys. Rev. E* **74**(3), 036104 (2006)
16. Newman, M.E., Girvan, M.: Finding and evaluating community structure in networks. *Phys. Rev. E* **69**(2), 026113 (2004)
17. Orman, G.K., Labatut, V., Cherifi, H.: Comparative evaluation of community detection algorithms: a topological approach. *J. Stat. Mech. Theory Exp.* **2012**(08), P08001 (2012)
18. Pons, P., Latapy, M.: Computing communities in large networks using random walks. In: *ISCIS*, vol. 3733, pp. 284–293 (2005)
19. Raghavan, U.N., Albert, R., Kumara, S.: Near linear time algorithm to detect community structures in large-scale networks. *Phys. Rev. E* **76**(3), 036106 (2007)
20. Filho, R.J., Brust, M.R., Ribeiro, C.H.: Consensus dynamics in a non-deterministic naming game with shared memory. arXiv preprint <arXiv:0912.4553> (2009)
21. Reichardt, J., Bornholdt, S.: Statistical mechanics of community detection. *Phys. Rev. E* **74**(1), 016110 (2006)
22. Rosvall, M., Bergstrom, C.T.: Maps of random walks on complex networks reveal community structure. *Proc. Natl. Acad. Sci.* **105**(4), 1118–1123 (2008)
23. Scott, J.: *Social Network Analysis*. Sage, California (2017)
24. Šubelj, L., Bajec, M.: Unfolding communities in large complex networks: combining defensive and offensive label propagation for core extraction. *Phys. Rev. E* **83**(3) (2011)
25. Uzun, T.G., Da Silva-Filho, R.J., Brust, M.R., Ribeiro, C.H.: Influence of shared memory and network topology in the consensus dynamics of a naming game. http://dimap.ufrn.br/csbc2011/anais/eventos/contents/SEMISH/Semish_Sessao_1_Artigo_3_Uzun.pdf
26. Xie, J., Szymanski, B.K.: Labelrank: A stabilized label propagation algorithm for community detection in networks. In: *Network Science Workshop (NSW)*. IEEE (2013)
27. Xie, J., Szymanski, B.K.: Community detection using a neighborhood strength driven label propagation algorithm. arXiv preprint <arXiv:1105.3264> (2011)
28. Xie, J., Szymanski, B.K., Liu, X.: Slpa: Uncovering overlapping communities in social networks via a speaker-listener interaction dynamic process. In: *Data Mining Workshops (ICDMW)*. IEEE (2011)
29. Yang, Z., Algesheimer, R., Tessone, C.J.: A comparative analysis of community detection algorithms on artificial networks. *Sci. Rep.* **6**, 30750 (2016)



Estimating the Similarity of Community Detection Methods Based on Cluster Size Distribution

Vinh-Loc Dao^(✉), Cécile Bothorel, and Philippe Lenca

IMT Atlantique - Lab-STICC CNRS, UMR 6285 F-29238 Brest, France
{vinh.dao,cecile.bothorel,philippe.lenca}@imt-atlantique.fr

Abstract. Detecting community structure discloses tremendous information about complex networks and unlock promising applied perspectives. Accordingly, a numerous number of community detection methods have been proposed in the last two decades with many rewarding discoveries. Notwithstanding, it is still very challenging to determine a suitable method in order to get more insights into the mesoscopic structure of a network given an expected quality, especially on large scale networks. Many recent efforts have also been devoted to investigating various qualities of community structure associated with detection methods, but the answer to this question is still very far from being straightforward. In this paper, we propose a novel approach to estimate the similarity between community detection methods using the size density distributions of communities that they detect. We verify our solution on a very large corpus of networks consisting in more than a hundred networks of five different categories and deliver pairwise similarities of 16 state-of-the-art and well-known methods. Interestingly, our result shows that there is a very clear distinction between the partitioning strategies of different community detection methods. This distinction plays an important role in assisting network analysts to identify their rule-of-thumb solutions.

Keywords: Community detection · Similarity metric
Community size · Comparative analysis

1 Introduction

Community detection discloses interesting information about the heterogeneous structure of complex networks and opens promising perspective in many theoretical as well as applied domains [8, 23, 24]. Although showing a high similarity with traditional unsupervised data clustering, community detection techniques have just been becoming prosperous in the last two decades remarked by the invention of modularity [14] and the availability of a large volume of networks from small scale to very large scale thanks to the development of Internet and notably social platforms. Since then, a numerous number of detection techniques with various approaches have been proposed [3, 5] to solve this network decomposition

problem. Even though communities are widely assumed to be sub-graphs where nodes are more densely connected relatively to the rest of the network, there is no commonly accepted standard process to evaluate the accuracy of detection methods. Indeed, the notion of goodness varies according to contextual objective and also the assumption about the underlying network model. By consequence, there is normally a confusion when one needs to find the most suitable method among available ones that is presumed to satisfy some specific requirements in outcome quality.

Meanwhile, the stated issue leaves behind rooms for developing theoretical and empirical techniques for comparing community detection algorithms. Actually, new methods are usually introduced in accompany with quality evaluation based on many variants of Mutual Information [27] or modularity. These two approaches work well to validate the functionality of proposed methods in ad-hoc networks but are not directly interpretable in a comparative evaluation of clustering quality. Actually, the former ones do not provide structural information of detected communities and the latter ones are dependent on hypotheses about null models. In other words, equivalent scores do not directly ensure an equivalence of partition quality.

In this paper, we are interested in estimating pairwise similarity of community detection methods based on expected community size. These estimates also reveal information about the closeness in terms of number of communities - a very important and intuitive characteristic of clustering algorithms - which is considered as an essential perspective in community detection literature [5] and recently addressed by many in-depth researches [7, 19]. Specifically, we conduct an empirical experiment to inspect a large number of state-of-the-art and widely used community detection methods and estimate their similarity using size distribution of communities that they discover on a large dataset of networks across several domains. The result of our analysis implicates that community detection methods can be classified in three well discernible groups exhibiting three essential strategies of node partition. These strategies produce a great impact on the outcome of community detection methods, making them very distinctive. We will show that this taxonomy exposes very useful information for proposing appropriate methods according to expected analysis strategy.

2 Estimating the Similarity of Community Detection Methods

We present a novel approach to determine the similarity of community detection methods using the size distribution of communities that they discover. Certainly, this is only one among interesting quality aspects that differentiate one method from the others. Nonetheless, it allows to get more insight into the difference in terms of partitioning strategy.

Specifically, a very naive but efficient approach to evaluate the similarity of two methods is to inquire into the “*closeness*” of the two corresponding community size distributions. As such, two methods could be supposed to be similar

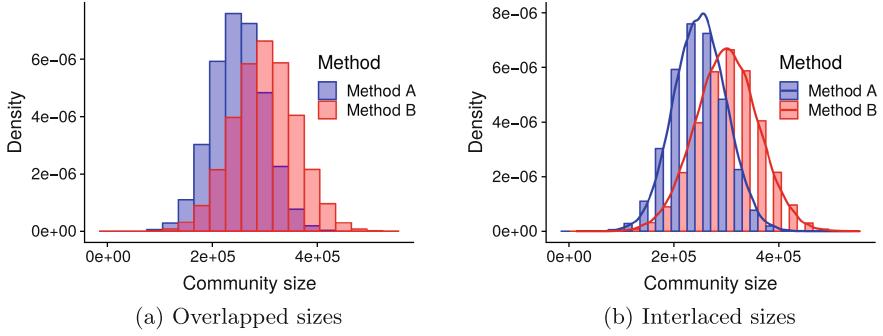


Fig. 1. The size distributions of communities detected by two different methods.

if their corresponding density distributions expose a large intersection area as shown in Fig. 1(a). From this notice, we can define our new similarity function as follows.

First, we denote two 2-tuples (\mathcal{A}, n^a) and (\mathcal{B}, n^b) being the multisets representing all communities detected on a set of networks $\mathcal{G} = \{G\}$ by method A and method B respectively, where $\mathcal{A} = \{x_1^a, x_2^a, \dots, x_r^a\}$ and $\mathcal{B} = \{x_1^b, x_2^b, \dots, x_s^b\}$ being the ascending ordered sets of sizes of communities: $1 \leq x_1^a < x_2^a < \dots < x_r^a$ and $1 \leq x_1^b < x_2^b < \dots < x_s^b$. The multiplicity functions $n^a : \mathcal{A} \rightarrow \mathbb{N}_{\geq 1}$ and $n^b : \mathcal{B} \rightarrow \mathbb{N}_{\geq 1}$ measure the number of communities of sizes x_i^a and x_i^b respectively. Let $N^a = \sum_{i=1}^r n^a(x_i^a)$ and $N^b = \sum_{i=1}^s n^b(x_i^b)$ being the total number of communities of all sizes detected by each method, we define a similarity function describing the closeness of A and B on \mathcal{G} as:

$$S_{\mathcal{G}}(A, B) = \frac{1}{2} \sum_{i=1}^r \sum_{j=1}^s \min \left\{ \frac{n^a(x_i^a)}{N^a}, \frac{n^b(x_j^b)}{N^b} \right\} \delta(x_i^a, x_j^b), \quad (1)$$

where $\delta(x_i^a, x_j^b) = 1$ if $x_i^a = x_j^b$ and 0 otherwise. Equation (1) is simply the common fraction of same-size communities detected on \mathcal{G} by both A and B : $0 \leq S_{\mathcal{G}}(A, B) \leq 1$. This definition seems to be intuitive but does not work well in practice. As illustrated in Fig. 1(b), when the sizes interlace each other, a low score will be produced although the similarity in this case is as much as that of the case of Fig. 1(a). Choosing an appropriate binning interval would mitigate the problem. This solution is, however very inflexible, sensible to the characteristic of data as well as to the functionality of the methods in use. A straightforward alternative can be envisioned by using a kernel density estimator to uncover the probability density functions as shown by the solid lines in Fig. 1(b). In this way, we approximate the common fraction of same-size communities of Equation (1) by the overlapping area of two corresponding continuous distributions. The premise behind this estimation is that two similar methods must not compulsorily produce a large portion of exactly same-size communities, but a large portion of comparable-size ones. Hence, we consider the following estimator to

take in local information of community size x_0 :

$$\hat{f}(x_0) = \frac{1}{hn} \sum_i K\left(\frac{x_i - x_0}{h}\right), \quad (2)$$

where h is the bandwidth controlling the neighborhood interval around x_0 and K is the kernel function controlling the weight given to the observations $\{x_i\}$ chosen as Gaussian in our analysis. Using this estimator, we rewrite the similarity function defined in Eq. (1) as follows:

$$S_{\mathcal{G}}(A, B) = \int \min\{\hat{f}^{(a)}(x), \hat{f}^{(b)}(x)\} dx, \quad (3)$$

where

$$\hat{f}^{(u)}(x) = \frac{1}{hN^u} \sum_i^{N^u} \left[n^u(x_i^u) K\left(\frac{x_i^u - x}{h}\right) \right], \quad (4)$$

with $u \in \{a, b\}$. In the estimations of this paper, the bandwidth h is selected based on the normal reference rule [26] to minimize the mean integrated squared error. The only exception is the cases illustrated Fig. 3 where a higher value has been chosen to get a higher smoothing quality for a better illustration.

Using Eqs. (3) and (4) to estimate the similarity between pairs of detection methods on a large dataset will help us discovering different behaviors of community detection methods. Since the accuracy of the estimator depends on the networks of the dataset that we analyze, the result will obviously relativized. However, a large and representative corpus would help to reduce the dependency impact.

3 Community Detection Methods

A community is roughly described as a group of nodes in a graph where there must be many edges connecting them together than edges connecting the community with the rest of the graph [5]. However, in practice, this concept is mathematically or algorithmically formulated in different ways engendering various discovery approaches. In this paper, we select a representative set of state-of-the-art and widely studied detection methods whose approaches spread out over the most commonly used in the literature. These methods are summarized in Table 1 with corresponding information. Their approaches could be briefly summarized as follows:

- **Edge removal:** In this approach, inter-community edges in a network are gradually removed in order to disconnect densely connected groups. The problem of community detection is translated to identifying candidates for inter-community edges based on their topological positions. Popular techniques include using edge betweenness centrality (GN in Table 1) or edge clustering coefficient, which could be based on triangular (RCCLP-3) or quadrangular (RCCLP-4) patterns.

Table 1. Community detection methods and associated implementations involved in our analyses grouped by different methodological approach. The label column denotes the corresponding abbreviations used in our paper.

Approach	Method	Label	Source
Edge removal	Girvan-Newman [14]	GN	igraph ¹
	Radicchi et al. [16]	RCCLP-3 (for $g = 3$)	Authors ²
	Radicchi et al. [16]	RCCLP-4 (for $g = 4$)	Authors ²
Modularity optimization	Clauset et al. [2]	CNM	igraph ¹
	Blondel et al. [1]	Louvain	Authors ³
	Newman [13]	SN	igraph ¹
Dynamic process	Pons et al. [15]	Walktrap	igraph ¹
	Rosvall et al. (2007) [22]	Infomod	Authors ⁴
	Rosvall et al. (2009) [21]	Infomap	Authors ⁵
Statistical inference	Lancichinetti et al. [10]	Oslom	Authors ⁶
	Riolo et al. [19]	(DC)SBM	Authors ⁷
Other methods	Reichardt et al. [18]	RB	igraph ¹
	Raghavan et al. [17]	LPA	igraph ¹
	Xie-Szymanski [28]	SLPA	Authors ⁸
	Demeo et al. [12]	Conclude	Authors ⁹

¹<http://igraph.org/> (Available in R, Python and C/C++)

²<http://homes.sice.indiana.edu/filiradi/resources.html>

³<https://sourceforge.net/projects/louvain/>

⁴<http://www.tpu.se/~rosvall/code.html>

⁵<http://www.mapequation.org/>

⁶<http://www.oslom.org/>

⁷<http://www-personal.umich.edu/~mejn/>

⁸<https://sites.google.com/site/communitydetectionslpa/>

⁹<http://www.emilio.ferrara.name/code/conclude/>

- **Modularity optimization:** Methods in this approach use a common objective function called *modularity* [14], but have different optimization strategies. The modularity function measures the quality of a partition by calculating the difference between the fraction of intra-community edges of the partition with the expected of such fraction in the associated partition whose edges are redistributed randomly following a null model.
- **Dynamic process:** Methods in this group do not use directly topological information in order to deduce densely connected subgraphs. Instead, they exploit stochastic information from various dynamic models regulated by network structure in order to deduce community structure.
- **Statistical inference:** This approach takes into consideration the statistical significance of community structure based on different theoretical network models. Such methods usually optimize likelihood functions to find the

Table 2. A summary of network used in this analysis where *Size* is the number of networks in each category, *Nodes* and *Edges* indicates the average number of nodes and edges of networks respectively. This dataset is collected from [9, 11, 20].

Category	Size	Nodes	Edges	Notable networks
Biological	7	1860	10763	Yeast, brain, protein-protein interactions
Communication	9	39595	195032	Email, forums, message exchanges
Information	25	38358	159812	Amazon, DBLP, citation & education webs
Social	37	6888	49666	Facebook, Youtube, Google plus networks
Technological	19	18431	48494	Internet, AS Caida, Gnutella P2P networks
Miscellaneous	11	4298	49033	Ecology, power-grid, synthetic networks
Total size *	108	1.99M	9.08M	

*The total number of networks, nodes and edges in the whole dataset respectively.

best configuration fitting hypothetical assumptions using different searching strategies.

- **Other methods:** Some approaches define implicitly or explicitly requirements about community structure or mix different traditional approaches to take the advantages of each one. In many cases, they can be classified into one theoretical family or another. To simplify the theoretical taxonomy, we present them in a common group.

To maintain the controllability of the experiment and to ensure the reproducibility of the analysis, all of the above presented methods are studied with the default parameters determined by the authors. The following section will be dedicated to an introduction of the network dataset that will be used in our experiment.

4 Network Dataset

Our experiment requires a large number of networks in order to reduce the impact of the irregularity which could be presented in a small set of ad-hoc networks. Hence, presuming that networks in different domains possess various structural particularities [4], we collect networks spanning of a variety of categories which are widely studied in the research community. To ensure that distribution of community size is allowed to be spread in a wide range, it is very essential to gather networks from small scale to large scale. We cover networks from around 30 vertices up to 320000 vertices and a million edges. Table 2 encapsulates the principle information of networks involved in our experiment.

Fig. 2 exhibits the distribution of network size in each category. As we can see, the marginal distributions on top imply that inside each category, networks also span in a relatively wide range of size with some slight differences from one category to another. Additionally, the networks in this dataset are quite sparse. The majority of networks has an average degree of approximately from 10 to 20

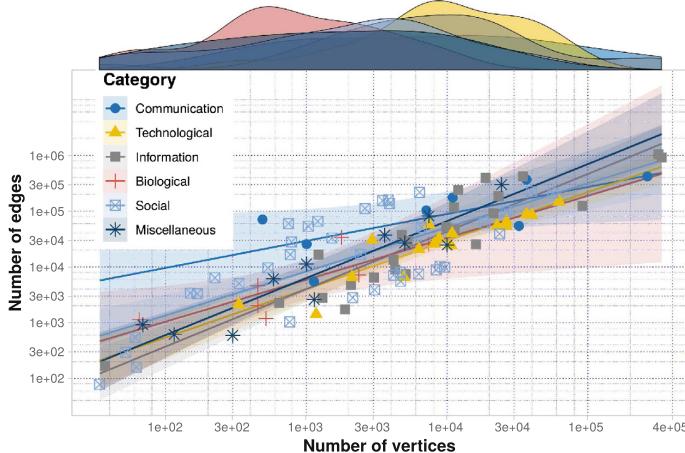


Fig. 2. Structural information of network employed in the experiment. The solid lines represent estimated relations between number of nodes and number of edges in each network category using a linear regression model. Accordingly, the translucent color backgrounds represent the corresponding 95% confidence intervals.

connections. Also, the number of edges increase linearly by the number of nodes with equivalent rates among categories as can be deduced from the gradients of the linear estimates.

5 Experimental Results

The experimental procedure is simple. We gradually employ each method presented in Section 3 to discover community structures on the whole set of networks summarized in Table 2. When the whole set of communities is identified, we proceed to measure the volumes of communities detected by each method to identify the elements of the corresponding 2-tuples. Finally, we use the similarity function defined by Equation (3) to estimate the closeness between each pair of methods.

Due to the huge number of necessary experiments, only processes having a reasonable theoretical estimated time and memory consumption are maintained (less than a few days and require at most 30 to 40 GBytes of memory). The outcome distributions are illustrated in Fig. 3.

As we can see, there is a clear difference in the densities of community size, showing that these methods have various partitioning strategies. Knowing that methods belonging to the same theoretical group (as shown in Table 1) are placed next to each other, we can notice some agreements between the theoretical families with practical outcomes as follows:

Edge removal: GN and RCCLP-3 have very similar distributions where a large number of communities are very small. This is due to the fact that in

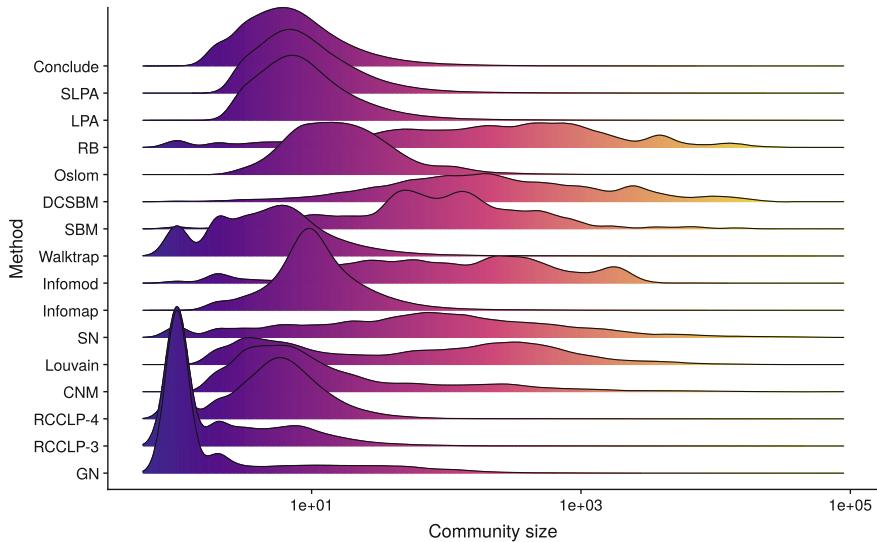


Fig. 3. The distribution of community size contained in the partitions detected on the networks of the dataset. The distribution are smoothed using a Gaussian kernel estimator. The illustrative gradient color is for the ease of view purpose.

some highly local centralized networks having star-like structures, they have a tendency to remove edges connecting hub and peripheral nodes, hence create singletons (single node community). This phenomenon is less distinguishable on RCCLP-4 since there are much less quadrangular than triangular connections in networks.

Modularity optimization: Modularity is known to suffer from resolution limit phenomenon [6], which often aggregates small communities in large scale networks. We can see from Fig. 3 that Louvain and SN found very large communities as predicted. In the meanwhile, there are also a comparable number of small communities found on small graphs. The behavior is a little bit different on CNM method.

Dynamic process: Methods in this family show very discernible distributions. Although all based on dynamic processes, they have different assumptions about community structure and searching mechanisms. Therefore, the closeness in the concept does not lead to a similarity in practical results.

Statistical inference: The Bayesian SBM and DCSBM uses Monte Carlo sampling process which is very time demanding in order to sweep the solution space. This makes the method unfeasible if the maximum number of clusters is not limited. Indeed, in the default version, the maximum number of communities is limited at 25 making (DC)SBM methods find very large communities in general. On the other hand, Oslom method use a different discovery mechanism and identify globally smaller communities.

Other methods: In this group, LPA, SPLA (both based on label propagation) and Conclude display nearly identical distributions. RB methods, being

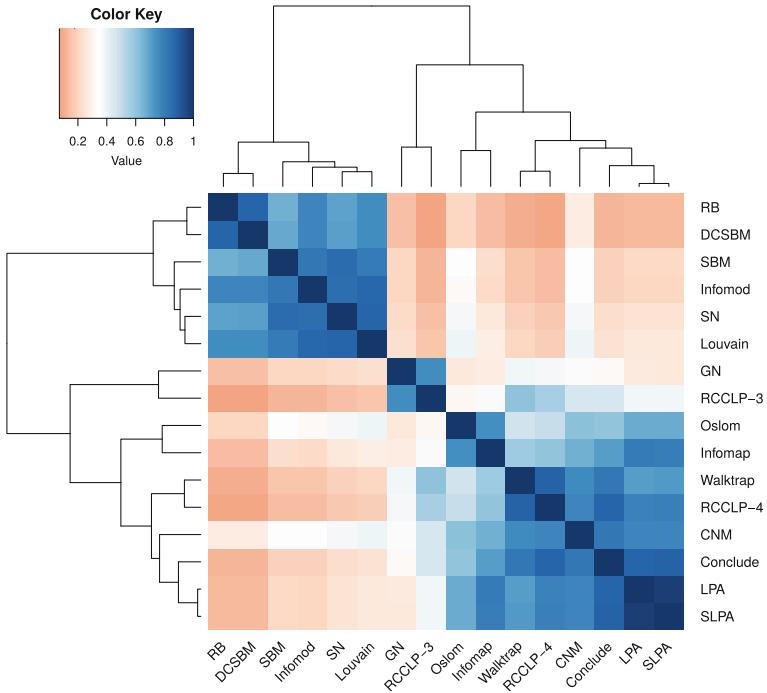


Fig. 4. The estimated proximity between detection methods. Similar methods share a large fraction of same-size communities. Methods are ordered using hierarchical clustering. The dendrogram proposes a hierarchical structure of the fitting closeness. Blue colors mean high similarity.

based on a very close concept with modularity (with a tuning parameter), exhibits a similarity with modularity optimization based methods.

Quantitatively, applying the estimator presented in Eq. (4) to compute pairwise similarities between the methods leads us to the results demonstrated in Fig. 4. As we can see, according to the community size criterion, these methods can be classified into different classes of partitioning strategy. The separations are very shaped showing that the distinction is very clear between groups. Therefore, we choose to characterize these methods by 3 (possibly 4) principle groups as follows:

Group 1: RB, DCSBM, SBM, Infomod, SN, Louvain: Methods in this group discover communities whose size vary in a very large spectrum, from very small to very large communities. The characterized community size distribution is quite flat, meaning all sizes are nearly equally considered on the dataset.

Group 2: GN and RCCLP-3: These two methods identify a huge number of very small communities including singletons regardless of network size. As a consequence, there are few variations in community volume.

Group 3: the others: These methods produce communities whose sizes approach bell-shaped distribution. The strategy can be translated as: not left not right, i.e. not too small and not too big communities.

As we can see, the strategies that these methods partition networks are very discernible. In fact, some other alternatives have been used to parametrize the estimator such as regulating the bandwidth h by using cross validation or pilot estimation of derivatives [25]; adapting other kernel functions K . However, the only major difference is noticed at GN and RCCLP-3 methods, which are no more highly similar. This variation is explainable since both GN and RCCLP-3 create a large number of very small communities, there are high spurious peaks in the distributions making the estimation unfeasible. In the meanwhile, using the original function of Equation (1) helps us to validate the closeness between GN and RCCLP-3.

Also, it worth noting that we could not yet conclude whether two methods are similar in a wide sense if their distributions are close; on the contrary we are able to deduce that they are not similar if their distributions are too different.

6 Discussion and Conclusion

Our experimental taxonomy discloses a new source of information that some traditional evaluation methods could not directly expose. For example, we demonstrate the similarity between partitions detected by these methods in Fig. 5 using Normalized Mutual Information (NMI) metric [27]. As we can see, one would hardly identify and get insight into the differences between detection methods in an intuitive way. In the meanwhile, using only community size distribution to deduce the similarity of methods could lead to unexpected results. Specifically, two methods could produce exactly the same sequence of community size, but the way that nodes are distributed into the two partitions are totally different. In this case, combining the two evaluation paradigms discloses complement information that helps network practitioners to characterize and deduce appropriate community detection methods. For instance, taking SBM and DCSBM, the average NMI score is approximately 0.5, which means that the two partitions are quite different. However, Fig. 1 divulges that SBM and DCSBM produce communities having quite similar sizes. In the case of LPA and SLPA on the other hand, the two methods produce quite identical outcomes as there is a consistency in the average NMI and our similarity index, which are all nearly 1.

Finally, the variation of the similarity according to the changing of input dataset has not been investigated comprehensively to validate the consistency of our result. In fact, it seems that the distributions illustrated in Fig. 3 have a tendency to move to the left hand side if more small scale networks are involved and inversely, to the right hand side if large scale networks are more implicated. Also, the (in)consistency of each method to the variation of data would not be at the same magnitude, i.e. some methods could be more “*robust*” in controlling community size than others. A further investigation is deemed necessary and promising as perspective.

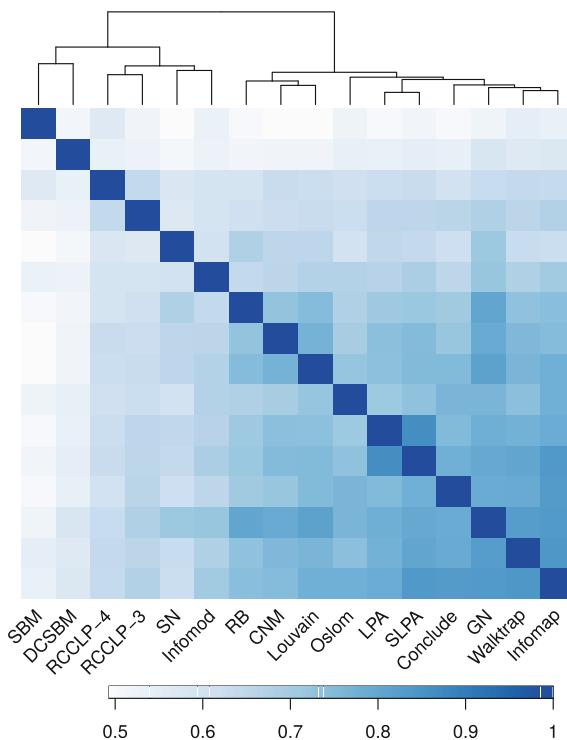


Fig. 5. The similarity of partitions average NMI.

References

- Blondel, V.D., Guillaume, J.L., Lambiotte, R., Lefebvre, E.: Fast unfolding of communities in large networks. *J. Stat. Mech.: Theory Exp.* **2008**(10), P10,008 (2008)
- Clauset, A., Newman, M.E.J., Moore, C.: Finding community structure in very large networks. *Phys. Rev. E* **70**(6) (2004)
- Coscia, M., Giannotti, F., Pedreschi, D.: A classification for community discovery methods in complex networks. *Stat. Anal. Data Min.* **4**(5), 512–546 (2011)
- Dao, V.L., Bothorel, C., Lenca, P.: An empirical characterization of community structures in complex networks using a bivariate map of quality metrics. ArXiv e-prints (2018)
- Fortunato, S.: Community detection in graphs. *Phys. Rep.* **486**(3–5), 75–174 (2010)
- Fortunato, S., Barthelemy, M.: Resolution limit in community detection. *Proc. Natl. Acad. Sci.* **104**(1), 36–41 (2006). <https://doi.org/10.1073/pnas.0605965104>
- Ghasemian, A., Hosseini Mardi, H., Clauset, A.: Evaluating overfit and underfit in models of network community structure. ArXiv e-prints (2018)
- Hizanidis, J., Kouvaris, N.E., Zamora-López, G., Diaz-Guilera, A., Antonopoulos, C.G.: Chimera-like states in modular neural networks. *Sci. Rep.* (19845) (2016)
- Jerome, K.: The koblenz network collection. In: Proceedings Conference on World Wide Web Companion, pp. 1343–1350 (2013). <http://konect.uni-koblenz.de>

10. Lancichinetti, A., Radicchi, F., Ramasco, J.J., Fortunato, S.: Finding statistically significant communities in networks. *PLoS ONE* **6**(4), e18,961 (2011)
11. Leskovec, J., Krevl, A.: SNAP Datasets: Stanford Large Network Dataset Collection (2014). <http://snap.stanford.edu/data>
12. Meo, P.D., Ferrara, E., Fiumara, G., Provetti, A.: Mixing local and global information for community detection in large networks. *J. Comput. Syst. Sci.* **80**(1), 72–87 (2014)
13. Newman, M.E.J.: Modularity and community structure in networks. *Proc. Natl. Acad. Sci.* **103**(23), 8577–8582 (2006). <https://doi.org/10.1073/pnas.0601602103>
14. Newman, M.E.J., Girvan, M.: Finding and evaluating community structure in networks. *Phys. Rev. E* **69**(2) (2004)
15. Pons, P., Latapy, M.: Computing communities in large networks using random walks. In: Yolum, p., Güngör, T., Gürgen, F., Özturan, C. (eds.) Computer and Information Sciences - ISCIS 2005, pp. 284–293. Springer, Berlin (2005)
16. Radicchi, F., Castellano, C., Cecconi, F., Loreto, V., Parisi, D.: Defining and identifying communities in networks. *Proc. Natl. Acad. Sci.* **101**(9), 2658–2663 (2004)
17. Raghavan, U.N., Albert, R., Kumara, S.: Near linear time algorithm to detect community structures in large-scale networks. *Phys. Rev. E* **76**(3) (2007)
18. Reichardt, J., Bornholdt, S.: Statistical mechanics of community detection. *Phys. Rev. E* **74**(1) (2006)
19. Riolo, M.A., Cantwell, G.T., Reinert, G., Newman, M.E.J.: Efficient method for estimating the number of communities in a network. *Phys. Rev. E* **96**(3) (2017)
20. Rossi, R.A., Ahmed, N.K.: The network data repository with interactive graph analytics and visualization. In: Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence (2015). <http://networkrepository.com>
21. Rosvall, M., Axelsson, D., Bergstrom, C.T.: The map equation. *Eur. Phys. J. Spec. Top.* **178**, 13–23 (2009). <https://doi.org/10.1140/epjst/e2010-01179-1>
22. Rosvall, M., Bergstrom, C.T.: An information-theoretic framework for resolving community structure in complex networks. *Proc. Natl. Acad. Sci.* **104**(18), 7327–7331 (2007)
23. Schaub, M.T., Delvenne, J.C., Rosvall, M., Lambiotte, R.: The many facets of community detection in complex networks. *Appl. Netw. Sci.* **2**(1) (2017)
24. Sekara, V., Stopczynski, A., Lehmann, S.: Fundamental structures of dynamic social networks. *Proc. Natl. Acad. Sci.* **113**(36), 9977–9982 (2016)
25. Sheather, S.J., Jones, M.C.: A reliable data-based bandwidth selection method for kernel density estimation. *J. R. Stat. Soc.* **53**(3), 683–690 (1991)
26. Silverman, B.W.: Density Estimation for Statistics and Data Analysis. Chapman and Hall, London (1986)
27. Vinh, N.X., Epps, J., Bailey, J.: Information theoretic measures for clusterings comparison: variants, properties, normalization and correction for chance. *J. Mach. Learn. Res.* **11**, 2837–2854 (2010)
28. Xie, J., Szymanski, B.K.: Towards linear time overlapping community detection in social networks. In: Advances in Knowledge Discovery and Data Mining, pp. 25–36. Springer, Berlin (2012)



Links in Context: Detecting and Describing the Nested Structure of Communities in Node-Attributed Networks

Tobias Hecking^(✉) and H. Ulrich Hoppe

University of Duisburg-Essen, Duisburg, Germany
hecking@collide.info, hoppe@collide.info

Abstract. This paper describes Links in Context as a novel approach for detecting and characterising the community structure in networks when further information on the properties of nodes is available. The general idea is straightforward and extends the well-known Link Communities framework introduced by Ahn et al. [1] by additionally taking node attributes into account. The basic assumption is that each edge in a social network emerges in a certain context, which is constituted by the node attributes shared by its two endpoints. In this regard, our approach focuses on subspaces of attributes that are relevant for explaining the emergence of particular edges. The proposed method allows for detecting highly overlapping community structures where nodes can be part of many groups emerging in different social contexts.

Keywords: Overlapping community detection · Attributed networks
Link Communities

1 Introduction

In addition to structural information on how nodes are connected in social networks, sometimes more information about the properties of the nodes or edges are available, for example, the strength of a connection or personal features of the actors (nodes). When it comes to the task of identifying densely connected groups of actors, it has been argued that focusing on structure or attributes alone may not be sufficient to adequately map the community structure of social networks [9]. In this regard, the attributes of the actors play a crucial role in the emergence of densely connected substructures (or communities). It can safely be assumed that many social ties do not occur at random but emerge in a certain social context. This context is, to some extent, represented by the properties of the actors. For example, taking a network of employees in a company and their connections based on email communication, knowing which project they are currently working on, their department, or whether or not they hold a membership to the company's gym would be useful for detecting social groups within the

network because of the tendency of actors to form social ties with others with whom they have certain properties in common, which is known as homophily [10, 13].

This consideration leads to the approach for community detection in node-attributed networks presented in this paper – in the following referred to as Links in Context, which differs from existing methods by a strong focus on the links instead of the nodes. The basic idea is to identify for every edge the subset of attributes its two endpoints have in common (context), and thus, can explain the emergence of the social connection. For example, Fig. 1 shows an ego node (center) that participates in different groups where the links in each group can be described by characteristic subsets of attributes.

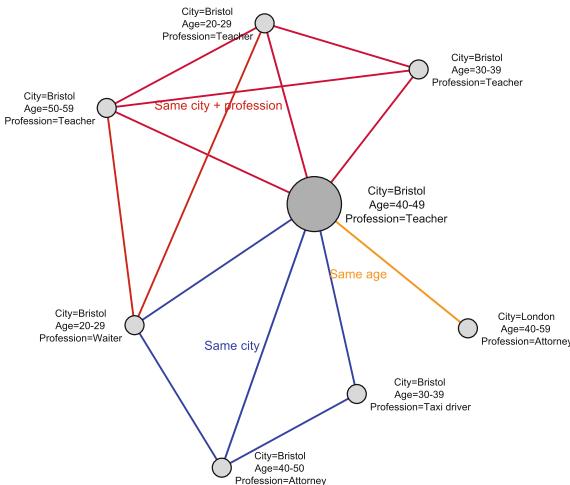


Fig. 1. Example of an ego participating in different groups in different contexts.

Many existing community detection methods partition the set of nodes into separated units, and thus, cannot model this kind of nested organisation of communities. However, similar to Ahn et al. [1] one can define a community as an agglomeration of links instead of nodes. This allows for an interpretation of communities as nested structures induced by nodes creating connections in multiple overlapping social contexts.

The proposed way of identifying and describing the community structure in social networks has the potential to advance applications in different areas. Recently attributed community detection has been successfully applied in the area of recommender systems [6]. Recommendations may be further contextualised by considering different communities a node participates in and their corresponding attributes. Another application is supporting access control decisions in social networks similar to [14] and [5]. To this end, a user specifies a short list of users as prototypes of an audience who should be excluded from seeing a post, for example, work colleagues should not see private pictures. Using the

network structure and node attributes, Links in Context can be used to find all social contexts matching the users on the exclusion list to augment the excluded audience.

The paper is structured as follows: After this introduction, Sect. 2 provides more context and discusses related work. The Links in Context approach as an extension of the Link Communities method [1] is described in Sect. 3. The evaluation of the method in comparison with other approaches on real-world networks is presented in Sect. 4. Section 5 concludes the findings.

2 Community Detection with Node Attributes

According to Fortunato [8] two main concepts are prominent in community detection: A proper subcommunity is marked by the compactness of the group in comparison to the data in general and by its separation from other groups. Moreover, if node attributes are known, they can be used for interpreting and explaining different subcommunities. While classical community detection algorithms only make use of the structural information to partition the nodes of a network, heterogeneous clustering techniques on attributed graphs additionally take the compositional domain, i.e. node attributes, into account.

In the case of perfect homophily, i.e. nodes with the same features always connect to each other, feature-based and network-based clustering algorithms will yield the same groups of nodes, and consequently, they would be interchangeable. Since this is usually not the case, the problem of community detection in node-attributed networks is not trivial on the methodological and especially on the conceptual level. For example, there are different interpretations of the example network given in Fig. 2 even on such a small scale.

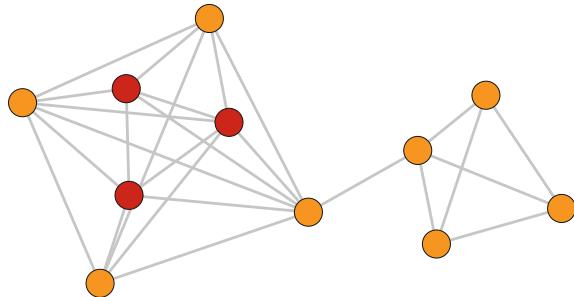


Fig. 2. Example of a network with node attributes and three node clusters

By considering the network structure only, one would partition the set of nodes into two groups with only one edge between them. This structure is easily detectable by well-known community detection methods such as Blondel's modularity optimisation (Louvain algorithm) [2]. Taking the node colours as node attributes there would be two different clusters (yellow nodes and red nodes) that

are discriminated by the colour attribute. Clustering algorithms that operate on attribute spaces, e.g. k-means, will produce such a partition. Both interpretations, either exclusively focusing on attributes or links, do not show the complete picture. The 7 nodes on the left can be interpreted either as a single group of nodes that are heterogeneous in their attributes, two groups of same coloured nodes with heavy interaction, or an attribute-heterogeneous group that is composed of two subgroups of same coloured nodes. Which interpretation is most appropriate depends on the application context, and thus, community detection algorithms for node-attributed networks should be flexible enough to allow different explanations of the results. To combine both aspects, node similarities in the attribute space can be represented as edge weights of a network that is used as input to network clustering algorithms. Vise versa the structural separation of nodes in the network can be transformed into a distance metric such that it can be combined with attribute-based distances. For an extensive review of specific implementations of both approaches we refer to Bothorel et al. [3].

Another challenge is that often only a subset of the whole attribute space is important for explaining a subcommunity (the attribute gender is important for the formation of ties in sports clubs but not necessarily in corporates). This is, for example, explicitly taken into account as ‘point of views’ by Cruz Gomez et al. [4].

In addition to the requirements of structural separation of node clusters and their description by concise subsets of attributes, it is often reasonable to allow nodes to belong to multiple subcommunities as described in the examples above. While many algorithms aim at partitioning the set of nodes into non-overlapping clusters, the CESNA algorithm [15] models the interdependence of the network structure and the node attributes in a probabilistic framework. Node clusters are derived by assigning nodes to overlapping groups that best explain the emergence of the observed network structure and the node attributes. In experiments it was shown that CESNA outperforms other algorithms in the task of re-discovering ground truth communities in node-attributed networks.

Similarly the Links in Context approach proposed in this paper, also allows for overlapping clusters that can be described by a subset of locally relevant node attributes. The difference is that it focuses on clustering edges instead of nodes (similar to [1]) based on the assumption that different social contexts can be better described as an agglomeration of similar links than similar nodes. In this regard a node can be part of many different groups leading to a highly nested community structure. A comparison of both approaches will be shown later in Sect. 4.

3 The Links in Context Approach

3.1 Data Representation

In this paper a node-attributed network $G = (N, E, Q)$ is represented by a set of nodes N , a set of undirected edges $E \subseteq [N]^2$, and a set of binary attributes Q . Furthermore, every attribute $q \in Q$ is of a certain type $t(q) \in T$. For example,

if there are the attribute types “gender” and “city” nodes can be described by different attribute combinations of these two types ($gender = female, city = Paris$), ($gender = male, city = Barcelona$), The method described in the following requires that every attribute type is categorical such that it can be represented as binary variables as in the examples above. Thus, attribute types like “age” have to be preprocessed and transformed into appropriate categories, for example, age range.

3.2 Extension of Link Communities

As mentioned before, the fundamental assumption behind our approach is that every link between two actors in a social network evolves in a social context that is constituted by certain attributes shared by its endpoints. Such contexts can, for example, be workplace, interest in motorcycling, age range, etc.

The idea that social context is rather captured in the links has inspired this work to build upon the Link Communities framework introduced by Ahn et al. [1]. The Link Communities algorithm differs from most other community detection algorithm in that it groups links instead of nodes. First, the similarity of edges $e_{i,j}$ and $e_{i,k}$ that are incident to a common node i is computed as the Jaccard similarity of the neighbourhood (set of adjacent nodes) denoted as $n(j)$ and $n(k)$ of the two other nodes j and k (Eq. 1). The assumption is that a community is better characterised as an agglomeration of links rather than nodes and nodes are typically part of many overlapping communities.

$$s(e_{i,j}, e_{i,k}) = \frac{|n(j) \cap n(k)|}{|n(j) \cup n(k)|} \quad (1)$$

Consequently, links are considered to be similar if they are embedded in the same densely connected parts of the network. It is important to note that s is symmetric but does not impose a strict similarity metric. Since only edges with a common incident node are compared, the triangle inequality does not hold for all triplets of edges. However, if the known similarities (or distances) between edges are taken as input for single-linkage hierarchical clustering, on the lower levels of the resulting cluster hierarchy the edges of a network are partitioned into clusters that are more likely to be located within densely connected regions than between those regions. Clusters of links can easily be turned into communities of nodes. Since every node i can possibly be part of $d(i)$ clusters, where $d(i)$ is the degree of node i , the resulting subcommunities are highly interleaved and can even be fully included in larger ones.

Link Similarity with Attributes In the following the idea of Link Communities is taken up, but in addition, the conditions leading to a tie between nodes, given by their features, are considered more explicitly. Let $a : N \mapsto [Q]^{|T|}$ be a mapping that assigns each node i a set of attributes (one of each type). The set of relevant attributes of an edge $e_{i,j}$, denoted as $r(e_{i,j}) = a(i) \cap a(j)$ are the attributes its two incident nodes have in common. For example, the relevant

attributes for the red edges in the small network in Fig. 1 are *city = Bristol* and *profession = Teacher* while other attributes such as age are not relevant. These relevant attributes are conceived to be the context in which an edge emerged.

In order to take into account the attributes of the endpoints of edges in addition to their structural embedding when calculating their similarity, Eq. 1 has to be modified to Eq. 2.

$$s_{LiC}(e_{i,j}, e_{i,k}) = \frac{\sum_{l \in n(j) \cap n(k)} |r(e_{i,j}) \cap r(e_{i,k}) \cap a(l)|}{|r(e_{i,j}) \cup r(e_{i,k})| * |n(j) \cup n(k)|} \quad (2)$$

$a(l)$ is the set of node attributes of a common neighbour l of nodes j and k . Apart from the context overlap (or common context $r(e_{i,j}) \cap r(e_{i,k})$) of two edges $e_{i,j}$ and $e_{i,k}$, the overlaps with the context of the edges incident to the non-shared nodes j and k are further explicitly considered. Thus, two links with a common endpoint have a high similarity if they can be explained by a common subset of their endpoints' attributes, and the two non-shared nodes have many common neighbours also sharing these attributes. The value is 1 if both edges share all their relevant attributes and the open ends have the same neighbourhood which also shares all these attributes.

Links in Context with Emphasis on Network Structure The variations for computing the similarity of links described before focuses very much on the attributes and less on the network structure neglecting phenomena such as triadic closure having no direct relationship with the node attributes. Thus, a compromise between the original Link Communities algorithm and the variation described above can be achieved if a new constant attribute is added to each node. It could, for example, be interpreted as “being part of the network”. The result is that the endpoints of each link have at least this pseudo attribute in common. Consequently, it cannot happen that the similarity between two links is 0 even if they are embedded in densely connected substructures of the network.

Deriving Link Clusters As in the original version of the Link Communities algorithm [1], in the next step the similarities of links are taken as input for single-linkage hierarchical clustering that partition the set of edges into link clusters. The resulting dendrogram allows for inferring the community structure on different levels of granularity, which maps the hierarchical and nested organisation of clusters.

Once a hierarchy of successively grown link clusters has been derived, a good criterion is needed to partition the links by cutting the dendrogram at a proper level. The original Link Communities method uses the partition density criterion (see [1]), which is based on the weighted link density of clusters.

A similar principle can be applied while additionally considering node attributes to calculate the attributed partition density *apd*. Here M is the total number of attributes that are relevant for the edges, i.e. attributes that both endpoints of an edge have in common. It can be seen as the weighted number of

edges if the edge weight is calculated as the number attributes shared by its incident nodes. The value m_c is the corresponding “weighted” number of edges that are associated to link cluster c . Since every two nodes can share T attributes, where T is the set of different attribute types, the right hand part of Eq. 3 is the fraction of shared and maximum possible shared attributes of nodes in cluster c . Note that $\frac{m_c}{M}$ sums up to one for all link clusters.

$$apd(C) = \frac{2}{M} \sum_{c \in C} m_c \frac{m_c}{|T||N|(|N| - 1)} \quad (3)$$

4 Evaluation

The methods described in Sect. 3, i.e. Links in Context and the original Link Communities approach (LC) are evaluated with respect to different aspects on different networks. The variation of Links in Context with an additional pesudo attribute to emphasise network structure (Sect. 3.2, §3) is named LiC2 while the standard version is referred to as LiC1. For these methods only non-trivial communities with more than 2 nodes are retained. Results are compared to the CESNA algorithm as one of the state of the art methods for community detection with node attributes [15]. As a typical representative of structural node partitioning, methods the Louvain modularity optimisation algorithm [2] for detecting non-overlapping communities was chosen.

4.1 General Observations

To first characterise the functionality and general properties of the compared approaches, the small example network given in Fig. 2 is used. By their nature, attribute-based clustering algorithms, such as k-means, detect the small group of red nodes as a cluster and the yellow nodes are merged into a single cluster (for $k = 2$) while modularity optimisation detects the two more or less separate parts of the graph as two clusters ignoring the color attribute of the nodes.

Surprisingly CESNA only detects the same clusters as Louvain, even if the number of clusters is bound to 3. This result is on the one hand reasonable since the yellow and the red nodes on the left side of the depiction nearly form a clique, and thus, can be considered as a densely connected group of nodes that have mixed attributes. On the other hand, the red group could also be considered as a nested substructure of a larger cluster that might also be of interest. Our Links in Context approach (LiC1) detects exactly 3 clusters, two yellow and the small red one if links without a context are completely ignored. More emphasis can be put on the network structure by introducing a new artificial attribute that is assigned to every node (LiC2). Using this modification Links in Context is able to identify 4 clusters. It detectes the two obviously structurally separated ones, as well as the yellow and red groups within the larger cluster on the left.

4.2 Characteristics of Identified Communities

To further study the properties of the proposed method, it is applied to networks assembled during a study of corporate law partnership that was carried out in the Northeastern US corporate law firm SG&R, 1988–1991 in New England by Emmanuel Lazega [11]. The nodes represent 71 attorneys (partners and associates) in the company. In a survey, the lawyers were asked to name their social connections within the company in three different dimensions. In particular, their coworker network, advice network, and friendship network. Apart from this, several attributes of the nodes are known (formal status, office in which they work, gender, law school attended, age, years in the company, and practice). Since Links in Context requires categorical attribute types, the numeric attribute types (age and years in company) were transformed to ranges of 5 years. Every edge has approximately 3 contextual attributes on average (attributes that are shared by both endpoints).

The community structures identified by LiC1 and CESNA for the friendship network are depicted in Fig. 3. Each cluster is represented as a node and directed links between a smaller and a larger cluster indicate that at least 75% of the smaller cluster is included in the larger one. The labels of the nodes are the descriptive attribute combinations of the communities that all of their nodes have in common (profiles).

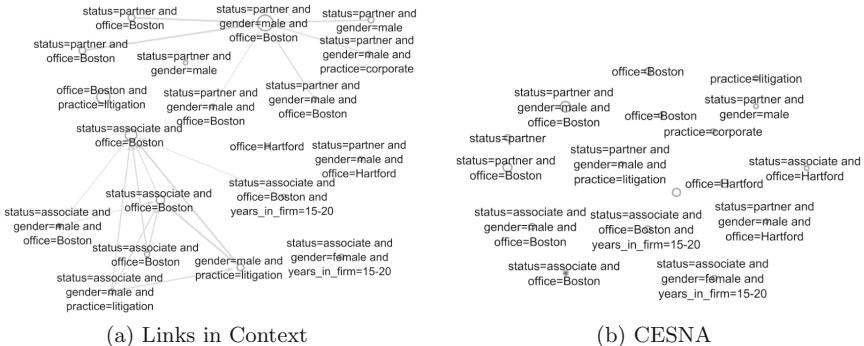


Fig. 3. Nested structure of the clusters in the lawyers friendship network derived by LiC and CESNA.

In both clusterings it can be seen that friendship clusters primarily emerge among lawyers who work in the same office (Boston or Hartford) and have the same status (partner or associate). Boston is by far the larger office and more clusters can be found within this office. Both algorithms also identify that the practice of lawyers (litigation or corporate) plays a role when it comes to the emergence of some of the social groups. Furthermore, there are several sub-communities that have similar characteristics with respect to the attributes they share. This indicates that homophily among people with similar profiles alone does not explain the emergence of the community structure. Otherwise, there

Table 1. Clustering for the Lawyer network for different algorithms. LiC1: Links in Context, LiC2: Links in Context with structure attribute, LC: Link Communities, CESNA, and Louvain

Network	Algorithm	Clusters	n-e ratio	Cluster size	Clusters/Node	att./clust.
Friend	LiC1	19	2.32	6.89	2.47	2.37
	LiC2	21	2.51	6.67	2.86	2.33
	LC	24	2.56	7.12	3.17	1.96
	CESNA	18	2.19	5.61	1.68	1.94
	Louvain	6	2.26	11.8	1	0.67
Co-Work	LiC1	10	2.28	9	1.61	2.2
	LiC2	20	1.85	6.9	2.23	1.76
	LC	23	1.94	6.57	3.02	1.78
	CESNA	17	1.89	5.47	1.52	1.82
	Louvain	4	2.99	17.8	1	0.75
Advice	LiC1	31	2.37	6.48	3.19	2.74
	LiC2	31	2.39	6.32	3.21	2.74
	LC	7	3.45	16.1	1.61	0.86
	CESNA	20	2.81	6.9	2.09	1.9
	Louvain	3	6.35	23.67	1	1

would be only one group per profile (see also discussion in Sect. 2). Links in Context differs from CESNA in that it identifies much larger overlaps between sub-communities. While CESNA mainly focuses on small groups, Links in Context is capable of identifying the partially hierarchical organisation of the social contexts, where smaller clusters can be almost fully included in larger clusters. Using the link-based approach, it becomes more salient that the office and the formal status are the more global social contexts in which smaller and more specific groups emerge. Furthermore, it is interesting to see that certain groups overlap less with other social contexts, for example, the group of female associates with a long history in the company, or the friendship circles in Hartford.

The quantitative characteristics of the community structure identified by different algorithms are summarised in Table 1. The number of clusters for the different variations of Links in Context were determined by cutting the link cluster dendrogram at the point with the maximal attributed partition density as defined in Eq. 3. For Link Communities the standard partition density was used. As expected, the pure structure-based methods Link Communities and Louvain typically yield denser clusters with a higher node-edge ratio (n-e ratio) on average. Louvain, thereby, detects less and larger clusters since it partitions the set of nodes into non-overlapping sets and does not account for smaller attribute-induced groups. Similar to Link Communities the number of clusters per node is high for Links in Context, which accounts very well for the fact that nodes form social ties with others in different contexts, and consequently, induce large overlaps between many different social groups.

The last column of Table 1 refers to the average number of descriptive attributes of node clusters. These are attributes that all the nodes of particular clusters have in common as an indicator how specifically a cluster can be interpreted. As expected, Louvain as an algorithm focussing only on network structure often yields node clusters which are densely connected but cannot be well described by a social context induced by node attributes. In this aspect, Links in Context yields the most specific clusters. However, the standard Link Communities methods as a pure structure based clustering algorithm also yields surprisingly good results in this regard.

As a general result, one can say that the most significant clusters can be obtained when node attributes are taken into account for the friendship network and the network based on advice relationships. A possible explanation is that these types of links are governed by the employees of the company themselves, while co-work and advice links are more externally influenced by the organisation. This goes along with the theory of focussed choice [7]. While people choose a company primarily for the job and not for their co-workers, people tend to choose their friends from their regular contacts within a focus group having certain personal characteristics.

4.3 Evaluation on Datasets with Ground Truth

In the following the Links in Context approach is applied to the publicly available Facebook social circles dataset¹. The data was collected for a study by McAuley and Leskovec [12] on predicting subsets of a user's friends corresponding to the user's social circles, for example, work colleagues, former schoolmates, etc. It consists of 10 ego networks of users who participated in the study. Each ego network is formed by the friends of ego and all the connections between them. The 10 ego users were asked to classify their Facebook connections into different social circles, thus, ground truth is available and the different networks are well suited to evaluate how well community detection algorithms perform on re-discovering these ground truth communities for each of the ego networks as done by Yang et al. [15]. In this work the ego networks and ground truth data were preprocessed. The ego nodes were ignored since they are connected to all other nodes in their network. Ground truth circles with less than two nodes were not considered. Furthermore, occasionally there are nodes without ground truth information, possibly because they could not be classified. These were also removed from the data. For space reasons only results for Links in Context with an additional structural attribute LiC2 which gives better results than LiC1 are discussed.

Since, both, the detected and actual node clusters overlap, and might differ in their number, the comparison between inferred and ground truth clusters cannot be done using standard comparison techniques such as adjusted Rand index. For this reason the same procedure as in [15] was applied for different

¹ <http://snap.stanford.edu/data/egonets-Facebook.html>.

quality measures ζ to compare an inferred clustering C' and a ground truth clustering C given in Eq. 4 below.

$$\hat{\zeta}(C, C') = \frac{1}{|C|} \sum_{C_i \in C} \max_{C'_j \in C'} \zeta(C_i, C'_j) + \frac{1}{|C'|} \sum_{C'_i \in C'} \max_{C_j \in C} \zeta(C'_i, C_j) \quad (4)$$

In this work the F1-score and the Jaccard overlap of clusters were used and the results averaged over the different ego networks are given in Table 2. It can be seen that Links in Context with emphasis on network structure (LiC2) performs well on the task of re-identifying the user labelled groups of nodes and almost similar to CESNA. The reason why LiC performs slightly worse than CESNA can be attributed to the different nature of the algorithms. LiC in most cases detects more clusters than in the ground truth dataset since several smaller clusters are included in larger ones, as explained in Sect. 4.2. The smaller clusters usually can be mapped to ground truth clusters very well, however, the higher total number of communities downweights the results. Surprisingly the difference between Links in Context and the original Link Communities algorithms is not very high in most of the dimensions. The reason is that the structural information is more important than the node attributes in a few of the circles in the Facebook dataset. The good performance achieved by only taking into account the network structure on the Facebook dataset is consistent with the results reported in previous work by Yang [15] and the original analysis of the dataset by McAuley and Leskovec [12] reporting structural belonging to a community as a strong explanatory factor for edges to emerge in addition to the attributes of the users.

Table 2. Average performance on re-identification of communities in the different Facebook networks

Algorithm	Avg. clusters	F1	Jaccard
LiC2	18.67	0.47	0.37
LC	19.22	0.45	0.35
CESNA	11.89	0.51	0.4

5 Conclusion

This paper introduced Links in Context as a method for identifying the nested community structure of node-attributed networks. The idea, thereby, is straight forward. Every edge is assumed to emerge in a certain social context constituted by the attributes its incident nodes have in common. By initially partitioning the links instead of nodes, every node can potentially be part of as many communities as its degree when the partition of links is turned into clusters of nodes leading to large overlaps between communities. This property makes the approach very different from other algorithms. The results on a network of lawyers

in New England (Sect. 4.2) showed that on a local level each node typically is part of several groups in possibly different contexts. The resulting nested and hierarchical community structure is better captured by the presented approach than by other techniques. However, when it comes to the identification of more or less separated groups of actors in node-attributed social networks (Sect. 4.3), CESNA, which finds small groups with less overlaps, performs better. Consequently, both types of algorithms complement each other focussing on different aspects of communities and targeting different applications.

References

1. Ahn, Y.Y., Bagrow, J.P., Lehmann, S.: Link communities reveal multiscale complexity in networks. *Nature* **466**(7307), 761–764 (2010)
2. Blondel, V.D., Guillaume, J.L., Lambiotte, R., Lefebvre, E.: Fast unfolding of communities in large networks. *J. Stat. Mech. Theory Exp.* **2008**(10), P10,008 (2008)
3. Bothorel, C., Cruz, J.D., Magnani, M., Micenková, B.: Clustering attributed graphs: models, measures and methods. *Netw. Sci.* **3**(3), 408–444 (2015)
4. Cruz Gomez, J.D., Bothorel, C., Poulet, F.: Semantic clustering of social networks using points of view. In: Proceedings of CORIA: Confrence en Recherche d'Information et Applications 2011 (2011)
5. Díaz Ferreyra, N.E., Hecking, T., Ulrich Hoppe, H., Heisel, M.: Access-control prediction in social network sites: examining the role of homophily. In: Proceedings of the 10th International Conference on Social Informatics, pp. 61–74. Springer International Publishing, Cham (2018)
6. Falih, I.: Attributed network clustering: application to recommender systems. Ph.D. thesis, University Sorbonne Paris Cité (2018)
7. Feld, S.L.: Social structural determinants of similarity among associates. *Am. Sociol. Rev.* 797–801 (1982)
8. Fortunato, S.: Community detection in graphs. *Phys. Rep.* **486**(3), 75–174 (2010)
9. Hric, D., Darst, R.K., Fortunato, S.: Community detection in networks: structural communities versus ground truth. *Phys. Rev. E* **90**(6), 062,805 (2014)
10. Lazarsfeld, P.F., Merton, R.K.: Friendship as a social process: a substantive and methodological analysis. *Free. Control. Mod. Soc.* **18**(1), 18–66 (1954)
11. Lazega, E.: The Collegial Phenomenon: The Social Mechanisms of Cooperation Among Peers in a Corporate Law Partnership. Oxford University Press (2001)
12. McAuley, J., Leskovec, J.: Learning to discover social circles in ego networks. In: Proceedings of the 25th International Conference on Neural Information Processing Systems, vol. 1, NIPS'12, pp. 539–547. Curran Associates Inc., USA (2012)
13. McPherson, M., Smith-Lovin, L., Cook, J.M.: Birds of a feather: homophily in social networks. *Ann. Rev. Sociol.* **27**(1), 415–444 (2001)
14. Misra, G., Such, J.M., Balogun, H.: Non-sharing communities? An empirical study of community detection for access control decisions. In: Proceedings of the 2016 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining, pp. 49–56 (2016)
15. Yang, J., McAuley, J., Leskovec, J.: Community detection in networks with node attributes. In: Proceedings of the 13th IEEE International Conference on Data Mining, pp. 1151–1156. IEEE (2013)



Overlapping Communities in Bipartite Graphs

Radek Marik¹✉ and Tomas Zikmund²

¹ Czech Technical University in Prague, Technicka 2, 166 27 Prague, Czech Republic
Radek.Marik@fel.cvut.cz

<https://comtel.fel.cvut.cz/en/users/marikr>

² Czech Technical University in Prague, Trojanova 13, 120 00 Prague, Czech Republic
zikmuto2@fjfi.cvut.cz

Abstract. Community detection in bipartite networks with overlapping communities carries difficult problems when it comes to complex network analysis. In this paper, we propose a new model for generating such graphs. We combine several approaches based on stochastic block models using edge probabilities following the Poisson distribution. The proposed model can be reduced into their original model versions, such as a model for a bipartite graph with non-overlapping communities only. We present results of the generator. Its performance is evaluated using several known community detection techniques. The evaluation criterion assesses both a community's identification and their overlaps.

Keywords: Stochastic block model · Bipartite graph
Overlapping communities · Graph generator

1 Introduction

In this article, we deal with a generative model that allows for the production of both a non-overlapping and overlapping community of objects, including unipartite and bipartite networks. The model can be used to generate synthetic data to verify properties of detection methods and compare them to each other. We believe that in the future the model will allow for the construction of a detection method that will focus on detecting overlapping communities in bipartite networks. As a motivational study case, we consider the bipartite networks of ancient Egyptian people and their titles. It is clear from the preserved Egyptian artifacts that the Old Kingdom's highly ranked officials were holders of a number of titles. Based on these titles, we can also reconstruct professional ties.

The rest of the article is organized as follows. In the next section, we describe the study case's data. The methods used to construct the proposed generating model are overviewed in Sect. 3. Section 4 introduces the proposed network-generating model covering non-overlapping or overlapping communities within a unipartite or bipartite network. We demonstrate the model's properties and results of selected community detection methods in Sect. 6.

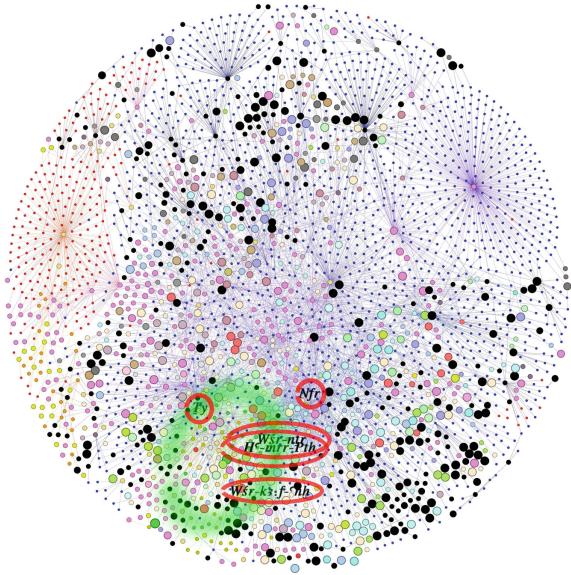


Fig. 1. The chart shows groups of people living during the 5th Dynasty. Small red, blue, green, yellow dots represent people (women, men, viziers, and royals, respectively); bigger circles of different colors are titles: yellow, members of the royal family, purple, priests, green, viziers, etc. The Fruchterman-Reingold method [7] implemented in Gephi [2] used for the layout of network nodes exhibits signs of clusters of officials with similar sets of titles. The green ellipse identifies a group of viziers and red ovals, highly ranked dignitaries.

2 Exemplar Study Case

The motivational dataset is a part of the Maat-base data, which was begun in 2006 by Egyptologist Veronika Dulikova and her colleagues [5]. Currently, 5000 highly ranked officials of the Old Kingdom of Egypt are recorded. Each record includes data on a person, such as his name, titles (testifying to employment and status), family relationships, identifications on the tomb, etc. Each official was the holder of up to dozens of titles, while the total number of titles is about 2000. Administrative titles are used to reconstruct the administrative structure and work activities of the individuals concerned. The titles are treated as binary attributes where 1 means the person holding the title and 0 otherwise.

If we select officers from the 5th Dynasty, then it is possible to construct the bipartite network shown in Fig. 1 from their binary vectors. This network contains two classes of nodes: officials and titles. These networks are bipartite (people and titles) with overlapping communities, with elements of a hierarchical arrangement that can change over time. Data exhibit a high level of noise and missing items to which many cluster or community detection methods are very sensitive. So far, there is no method known to address all these aspects in a consistent manner.

3 Related Methods

The aim of this article is to design a graph model that covers unipartite or bipartite graphs with non-overlapping and overlapping communities. Let $\mathcal{G}(V, E)$ be an undirected multigraph on n vertices, possibly including self-edges, and let \mathbf{A}_{ij} be an element of the adjacency matrix of the multigraph. \mathbf{A}_{ij} is equal to the number of edges between vertices i and j , the diagonal element \mathbf{A}_{ii} is equal to twice the number of self-edges from i to itself. The vertex degree d_i is defined as $d_i = \sum_{j \in V} \mathbf{A}_{ij}$. Then $\sum_{i \in V} d_i = \sum_{i,j \in V} \mathbf{A}_{ij} = 2m$, where $m = |E|$ is the number of edges.

3.1 Ordinary Stochastic Block Model

We have chosen a well-known and widely used ordinary stochastic block model as the initial model for the graph [9]. The ordinary block model considers a division of the vertices into k groups. Following [10] and [14] it is assumed the number of edges between each pair of vertices are independently distributed according to the Poisson distribution. Let ω_{rs} be a $k \times k$ matrix. Its element expresses the expected value of the adjacency matrix element \mathbf{A}_{ij} for vertices i and j lying in groups r and s , respectively. Thus, the expected number of edges (including self-edges) between any two vertices inside group r is $\frac{1}{2}\omega_{rr}$ prescribed by the diagonal elements. Furthermore, the graph is described by a mapping $g : V \rightarrow k$ assigning vertices into groups.

According to the ordinary stochastic block model (SBM) we express the probability $P(\mathcal{G}|\omega, g)$ of graph \mathcal{G} given the parameters and group assignments as [10]

$$P(\mathcal{G}|\omega, g) = \prod_{i < j} \frac{(\omega_{g_i g_j})^{\mathbf{A}_{ij}}}{\mathbf{A}_{ij}!} \exp(-\omega_{g_i g_j}) \cdot \prod_i \frac{(\frac{1}{2}\omega_{g_i g_i})^{\mathbf{A}_{ii}/2}}{(\mathbf{A}_{ii}/2)!} \exp(-\frac{1}{2}\omega_{g_i g_i}). \quad (1)$$

The maximum-likelihood estimated value $\hat{\omega}_{rs}$ for a given g can be found to be

$$\hat{\omega}_{rs} = \frac{m_{rs}}{n_r n_s} \leq 1, \quad m_{rs} = \sum_{i,j \in V} \mathbf{A}_{ij} \delta_{g_i, r} \delta_{g_j, s}, \quad (2)$$

where m_{rs} is the total number of edges between group r and s and n_r is the number of vertices assigned to group r , δ is the Kronecker function.

3.2 Degree Corrected Stochastic Block Model

The model (1) controls the probabilities of edges between groups, otherwise the edges are placed completely at random without taking the groups into account, in contrast to most real networks. In [10] the authors also introduced a stochastic block model that directly incorporates arbitrary heterogeneous degree distributions. This idea is built around the null model used in an objective function, a so-called modularity, which is used in community detection methods. A new

vector θ of the size n is added to the parameters of the multigraph \mathcal{G} . The θ_i values fulfill the normalization constraint

$$\sum \theta_i \delta(r, g_i) = 1, \quad \forall r \in \{1, \dots, k\}. \quad (3)$$

It means θ_i can be treated as the probability that an edge connected to the community to which i belongs is incident with i itself. According to the degree-corrected stochastic block model (DSBM) a graph \mathcal{G} has a probability [10]

$$P(\mathcal{G}|\theta, \omega, g) = \prod_{i < j} \frac{(\theta_i \theta_j \omega_{g_i g_j})^{A_{ij}}}{A_{ij}!} \exp(-\theta_i \theta_j \omega_{g_i g_j}) \cdot \prod_i \frac{(\frac{1}{2} \theta_i^2 \omega_{g_i g_i})^{A_{ii}/2}}{(A_{ii}/2)!} \exp(-\frac{1}{2} \theta_i^2 \omega_{g_i g_i}). \quad (4)$$

The maximum-likelihood values of the parameters θ_i and ω_{rs} for a given g are determined as

$$\hat{\theta}_i = \frac{d_i}{d(g_i)}, \quad \hat{\omega}_{rs} = m_{rs}, \quad \text{where} \quad d(r) = \sum_{i \in V} d_i \delta_{g_i, r}. \quad (5)$$

Let $\theta_i = \frac{1}{\sum_j \delta_{g_i, g_j}} = \frac{1}{n_{g_i}}$. Then

$$\theta_i \theta_j \hat{\omega}_{g_i g_j} = \frac{m_{g_i g_j}}{n_{g_i} n_{g_j}}. \quad (6)$$

Thus, if θ_i is distributed uniformly then the DSBM degrades into an ordinary SBM.

3.3 Bipartite Stochastic Block Model

A bipartite network is based on two types of vertices, a and b , and only vertices of different types can be connected. Vertices of a bipartite graph are described using two disjunctive sets V and W . We add this constraint to the graph's description so that a mapping t determines the type of its vertices

$$t : V \cup W \rightarrow \{a, b\} : t_i \mapsto \begin{cases} a & \text{for } i \in V, \\ b & \text{for } i \in W. \end{cases} \quad (7)$$

Let vertex i be of type t_i and belong to group g_i . Let T_r be the type of group r , i.e. $T : k \rightarrow \{a, b\}$. Vertex types and group types must match, therefore $t_i = T_{g_i}$. Then we can add the constraint for edges: $t_i = t_j \implies p(i, j) = 0$.

The probability the bipartite SBM (biSBM) generates a particular graph \mathcal{G} is [11]:

$$P(\mathcal{G}|\omega, g, t) = \prod_{i < j, t_i \neq t_j} \frac{(\omega_{g_i g_j})^{A_{ij}}}{A_{ij}!} \exp(-\omega_{g_i g_j}) \quad (8)$$

with a restriction on ω

$$\omega_{rs} = 0 \quad \text{when} \quad T_r = T_s. \quad (9)$$

Similarly, the biSBM can be extended to the bipartite degree corrected SBM (biDSBM) that generates a particular graph \mathcal{G} with the probability [11]:

$$P(\mathcal{G}|\theta, \omega, g, t) = \prod_{i < j, t_i \neq t_j} \frac{(\theta_i \theta_j \omega_{g_i g_j})^{A_{ij}}}{A_{ij}!} \exp(-\theta_i \theta_j \omega_{g_i g_j}) \quad (10)$$

with a restriction (9) on ω .

3.4 Stochastic Block Model and Overlapping Communities

In [1] the authors introduced a class of graph models that allow the description of so-called *overlapping communities*. It means the vertices are not only partitioned into several communities, but they can also simultaneously have memberships of different strengths in several communities. If two vertices share some communities then there exists an edge between them with a high probability [13]. There might be an edge between vertices with no common communities, but with a very low probability ϵ . This idea is captured in Fig. 2. The model uses a $n \times k$ matrix \mathbf{F} where k is the number of communities and n is the number of multigraph vertices. The element \mathbf{F}_{ir} represents the propensity of vertex i to community r . The expected number of edges generated by all communities between two vertices i and j is $\sum_z \mathbf{F}_{iz} \mathbf{F}_{jz}$ (or $\frac{1}{2} \sum_z \mathbf{F}_{iz} \mathbf{F}_{iz}$ for self-edges). The probability of generating a multigraph \mathcal{G} with adjacency matrix elements A_{ij} with edges independently distributed according to the Poisson distribution is [1] (shortly overSBM)

$$P(\mathcal{G}|_F) = \prod_{i < j} \frac{(\sum_z \mathbf{F}_{iz} \mathbf{F}_{jz})^{A_{ij}}}{A_{ij}!} \exp(-\sum_z \mathbf{F}_{iz} \mathbf{F}_{jz}) \cdot \prod_i \frac{(\frac{1}{2} \sum_z \mathbf{F}_{iz} \mathbf{F}_{iz})^{A_{ii}/2}}{(A_{ii}/2)!} \exp(-\frac{1}{2} \sum_z \mathbf{F}_{iz} \mathbf{F}_{iz}) \quad (11)$$

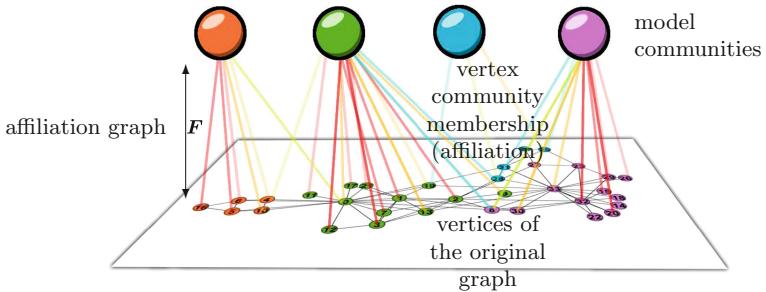


Fig. 2. A model with overlapping communities. In fact, there are two graphs. The model can be understood as a bipartite graph with weighted edges (affiliations). Model vertices are located at the top, while the vertices of the original graph are at the bottom. The model treats the edges of the original graph as projections.

4 Stochastic Block Model for Overlapping Communities in a Bipartite Graph

In this section, we propose a new SBM model that is capable of covering overlapping communities in a bipartite graph. In fact, the model is a suitable combination of the particular methods introduced in the previous section. Of course, the model can also describe both non-overlapping and overlapping communities in unipartite graphs.

Recall that the models SBM, DSBM, biSBM, biDSBM, overSBM operate with the following matrices of the expected number of edges between vertices i and j

$$\text{SBM } \Lambda_{i,j} = \omega_{g_i g_j}, \quad (12)$$

$$\text{DSBM } \Lambda_{i,j} = \theta_i \omega_{g_i g_j} \theta_j, \quad (13)$$

$$\text{biSBM } \Lambda_{i,j} = \theta_i \omega_{g_i g_j} \theta_j \text{ with the restriction } \omega_{rs} = 0 \text{ if } T_r = T_s. \quad (14)$$

$$\text{overSBM } \Lambda_{i,j} = (\mathbf{F}^T \mathbf{F})_{ij}. \quad (15)$$

The proposed model of a generated graph is described using two matrices. The first $k \times n$ matrix \mathbf{G} assigns community memberships to the vertices. The number of communities is k , the number of vertices is n . The membership is expressed as a non-zero matrix element. A vertex can be a member of more than one community, if so, then the given matrix \mathbf{G} column contains additional non-zero elements. The second matrix ω records community behavior: how the communities are linked to each other. The matrix ω is symmetric. The matrix of the expected number of edges between vertices i and j is proposed in the following way (biOverDSBM):

$$\text{biOverDSBM } \Lambda_{i,j} = (\mathbf{G}^T \omega \mathbf{G})_{ij}. \quad (16)$$

In the case of a bipartite graph, its vertices must belong to only one of the two parts and no edge can exist between any two vertices of the same part. Therefore, communities can also be divided into two parts. No vertex can have any membership in the community of the other part, and so the probability of an edge between any two communities of the same part is zero. Then matrices \mathbf{G} and ω can be rearranged by permutations of rows and columns into a block shape

$$\omega = \begin{pmatrix} \mathbf{0} & \mathbf{C} \\ \mathbf{C}^T & \mathbf{0} \end{pmatrix}, \quad \mathbf{G} = \begin{pmatrix} \mathbf{A} & \mathbf{0} \\ \mathbf{0} & \mathbf{B} \end{pmatrix}, \quad (17)$$

where matrices \mathbf{A} and \mathbf{B} contain affiliations of vertices of the first and second type, while matrix \mathbf{C} describes relationships between the communities. Therefore, the expected number of edges (i, j) can be expressed as the matrix product $\mathbf{A}^T \mathbf{C} \mathbf{B}$:

$$\mathbf{G}^T \omega \mathbf{G} = \begin{pmatrix} \mathbf{A}^T & \mathbf{0} \\ \mathbf{0} & \mathbf{B}^T \end{pmatrix} \begin{pmatrix} \mathbf{0} & \mathbf{C} \\ \mathbf{C}^T & \mathbf{0} \end{pmatrix} \begin{pmatrix} \mathbf{A} & \mathbf{0} \\ \mathbf{0} & \mathbf{B} \end{pmatrix} = \begin{pmatrix} \mathbf{0} & \mathbf{A}^T \mathbf{C} \mathbf{B} \\ \mathbf{B}^T \mathbf{C}^T \mathbf{A} & \mathbf{0} \end{pmatrix}. \quad (18)$$

In fact, the entire model is a combination of two graphs. The first one has ω as its adjacency matrix and it links communities. The second graph is bipartite with weighted edges described by incident matrix \mathbf{G} and it links the vertices of the multigraph to their communities. This idea is depicted in Fig. 3. It is obvious that the proposed structure $\mathbf{G}^T \omega \mathbf{G}$ enables other community forms such as mixed communities [4, 12] that are not discussed in this paper.

5 Model Reduction Cases

The proposed model can be classified as a generating stochastic block model. If matrix \mathbf{G} is defined as $G_{ri} = \delta(r, g_i)$, then the proposed model is reduced into the *standard SBM* with matrix ω with elements ω_{rs} describing the probability of edges between vertices of communities r and s , see Sect. 3.1.

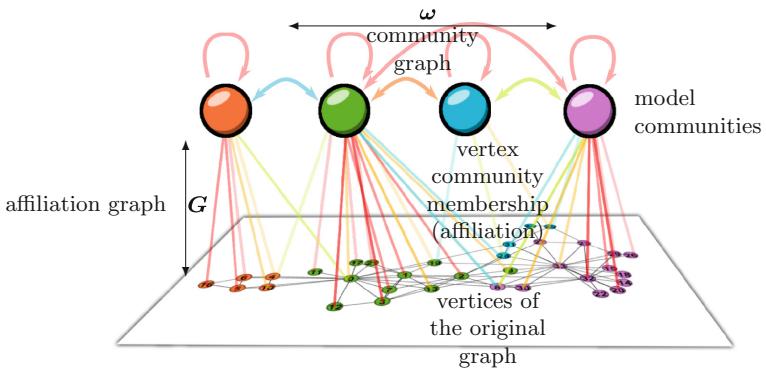


Fig. 3. The chart illustrates that the resulting graph is a consequence of the model represented as two graphs. The community graph with adjacency matrix ω is shown in the upper zone. The affiliation graph with incident matrix \mathbf{G} is depicted in the middle zone. The original graph is located at the bottom. In the case of models of overlapping communities of a unipartite graph presented in [1, 15] the community graph contains only loops as $\omega = \mathbf{I}$.

If matrix \mathbf{G} is defined as $G_{ri} = \theta_i \delta(r, g_i)$, then the proposed model is reduced into the *degree-corrected SBM* with matrix ω with elements ω_{rs} describing the probability of edges between vertices of communities r and s , and with vector θ satisfying the normalization constraint (3) that prescribes a vertex strength transformed into a vertex degree, see Sect. 3.2.

If matrix \mathbf{G} is defined as $\mathbf{G} = \mathbf{F}^T$ and $\omega = \mathbf{I}$, then the proposed model is reduced into the *overlapping SBM* as the matrix of expected numbers of edges can be reduced in the following manner, compare with Sect. 3.4, $\mathbf{G}^T \omega \mathbf{G} = \mathbf{F} \mathbf{I} \mathbf{F}^T = \mathbf{F} \mathbf{F}^T$. The equality $\omega = \mathbf{I}$ accurately captures the classical notion of communities as groups of vertices with dense linking between them compared to the rest of the graph.

6 Experiments

Our generator implementation is rather simple. Based on model parameters, the parameters are converted into the matrix Λ with an expected number of edges. Then the generator adds an edge (i, j) between a pair of vertices i and j if a generated random number is below the probability $1 - \exp(-\Lambda_{ij})$ computed according to the Poisson distribution with the expected number of edges Λ_{ij} [8].

We performed two types of experiments. First, we tested if a model prescription generates networks with selected properties, e.g. input parameters aiming a unipartite network with overlapping communities converted into the matrix Λ would lead to a resulting network by checking the node connectivity. Secondly, we selected several community detection methods and compared the resulting communities with those prescribed. In other words, the known community methods designed for a given kind of networks are expected to confirm that the generated network has the prescribed form of communities. We tested the following methods:

- detection of non-overlapping communities in unipartite graphs using the *Louvain algorithm* [3] and the *overlapping SBM* [1] with the vertices partition based on the strongest membership,
- detection of overlapping communities in unipartite graphs using the *overlapping SBM* [1] and the *BigClam* method [15].
- detection of non-overlapping communities in bipartite graphs using *bipartite SBM* [11].

6.1 Evaluation Function

Furthermore, we need an evaluation function that will determine how the prescribed and detected structures are different because different numbers of communities can be detected and the communities can be permuted. We use the following simple approach. At first, each detected community is assigned to a prescribed community based on conformity according to the Jaccard index [6]¹ $J(A, B) = \frac{|A \cap B|}{|A \cup B|}$, where A, B are countable sets and $A \cup B \neq \emptyset$. The detected communities are merged according to their highest value on the Jaccard index with the prescribed community and called *aggregated communities*. The Jaccard index reflects a relative overlapping of two sets. Then the number of aggregated communities is equal to the number of prescribed communities. Each pair of an aggregated community and its prescribed community is compared using the Rand index $R(A, B) = \frac{|A \cap B|}{|X|}$, where $X \neq \emptyset$ is a countable set and $A, B \subset X$ its subsets. The Rand index can be understood as a measure of an absolute overlap. Thus, a square matrix is obtained and its elements are ratios of the shared vertices to the number of vertices on the graph. The diagonal elements reflect the size of the community and the off-diagonal elements measure

¹ We are aware of the trivial case in which each vertex is assigned to its own community and is assessed as the best detection.

the overlap sizes for both aggregated and generated communities, R^A and R^G respectively $\mathbf{R}_{rs}^A = R(M_r^A, M_s^A)$, $\mathbf{R}_{rs}^G = R(M_r^G, M_s^G)$, where M_r^A and M_r^G are subsets of membership vertices of aggregated and generated communities for $r, s \in \{1, 2, \dots, k\}$ and k is the number of generated communities.

The resulting evaluation criterion is given by a matrix distance using the Frobenius norm

$$\text{EVAL} = \|\mathbf{R}^A - \mathbf{R}^G\|_F. \quad (19)$$

6.2 Generated Examples

We manually selected 40 different combinations of graph parameters covering unipartite and bipartite graphs with non-overlapping and overlapping communities. 100 graphs with 100 vertices were generated for each prescription. An example of one such generated bipartite graph with 3 and 2 overlapped communities for each node type (50 + 50 vertices) is depicted in Fig. 4a, b, c, and d with the results of the selected detection methods. The matrices ω_1 and G_1 in (17) are prescribed by

$$\mathbf{C}_1 = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \end{pmatrix}, \quad \mathbf{A}_1 = \begin{pmatrix} \boldsymbol{\alpha}_{11} & \boldsymbol{\alpha}_{12} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{\alpha}_{22} & \boldsymbol{\alpha}_{23} \end{pmatrix}, \quad \mathbf{B}_1 = \begin{pmatrix} \boldsymbol{\beta}_{11} & \boldsymbol{\beta}_{12} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{\beta}_{22} & \boldsymbol{\beta}_{23} & \boldsymbol{\beta}_{24} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \boldsymbol{\beta}_{34} & \boldsymbol{\beta}_{35} \end{pmatrix}, \quad (20)$$

where submatrices $\boldsymbol{\alpha}_{12}, \boldsymbol{\alpha}_{22}$ and $\boldsymbol{\beta}_{12}, \boldsymbol{\beta}_{22}, \boldsymbol{\beta}_{24}, \boldsymbol{\beta}_{34}$ represent the overlaps containing 10 and 5 + 5 nodes respectively, the elements \mathbf{A}_{ij} and \mathbf{B}_{ij} of one-row

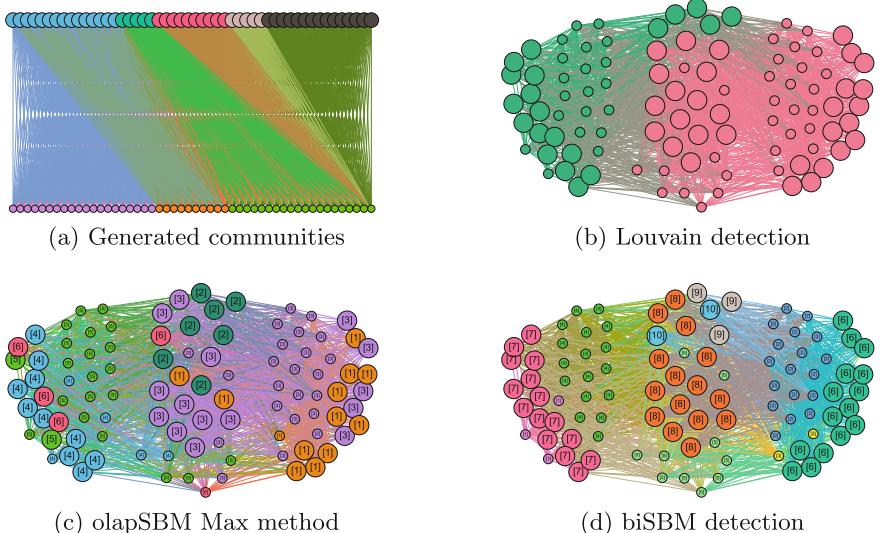


Fig. 4. A bipartite graph with overlapping communities is prescribed (a), generated by the proposed model, two node types which differ in size. Communities detected by the three methods are shown in (b, c, d). There are 5 (3 + 2) colored communities. Their overlaps are also colored differently.

matrices α_{kl}, β_{kl} follow the power law distribution of the node degrees, and $\sum_i A_{ij} = \mathbb{E} d_i^A, \sum_i B_{ij} = \mathbb{E} d_i^B$ (\mathbb{E} denotes the expected value).

Next, the communities and their potential overlaps were detected using the selected methods. For example, the overall assessment table in Fig. 5a contains evaluation scores for the case of unipartite graphs defined by prescriptions consisting of 4 cases of linked communities without overlaps and 3 cases of unlinked communities with overlaps of 6, 10, 20 vertices. The first 4 cases are prescribed by the matrix ω_2 with the mixing parameter $\mu \in \{0, 0.01, 0.1, 0.2\}$ and by the matrix G_2 (the rows γ_{kk} follow the power law distribution). The last 3 cases are prescribed by the matrices $\omega_3 = I$ and G_3 (the one-row submatrices γ_{12}, γ_{22} represent the overlaps, the one-row matrix rows γ_{ij} follow the power law distribution):

$$\omega_2 = \begin{pmatrix} 1 - \mu & \mu & 0 \\ \mu & 1 - 2\mu & \mu \\ 0 & \mu & 1 - \mu \end{pmatrix}, G_2 = \begin{pmatrix} \gamma_{11} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \gamma_{22} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \gamma_{33} \end{pmatrix}, G_3 = \begin{pmatrix} \gamma_{11} & \gamma_{12} & \mathbf{0} \\ \mathbf{0} & \gamma_{22} & \gamma_{23} \end{pmatrix}. \quad (21)$$

It is obvious in Fig. 5a that the community detection methods detect the communities as expected in the networks of the form for which they were designed. The related EVAL values are very low and highlighted with a blue background. Thus, we can state the proposed model and the evaluation function work correctly because the relevant methods are assessed successfully. Furthermore, the Louvain method returns the errors proportional to the size of overlaps (the cases 5–7).

The overall assessment table in Fig. 5b contains evaluation scores for the case of bipartite graphs with 3 + 2 overlapped communities defined by prescriptions

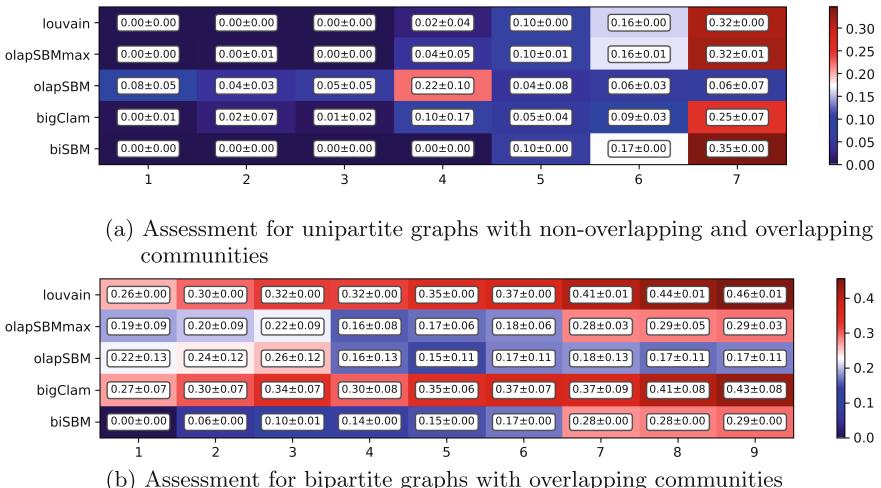


Fig. 5. The assessment table with evaluation scores EVAL for a set of unipartite graphs with 2 overlapping communities is in (a). The assessment table with evaluation scores EVAL for a set of bipartite graphs with 3 + 2 overlapping communities is in (b).

consisting of 9 cases with overlaps of 0, 10, 20 vertices in the first node type and with 0, 10, 16 vertices in the second type prescribed by the matrices C_1, A_1, B_1 . None of the detection methods used was designed for this case of networks, so there is always a range of configurations for which a given method returns a non-conforming result. The given prescription is signaled by higher EVAL values and a red background.

It is quite interesting that the biSBM performs fairly well even for overlapping communities in both unipartite and bipartite graphs. Generally, the methods return weaker results if the communities' overlap is bigger.

7 Conclusion

In this work, we proposed a new variant of the stochastic block model that is capable of generating both unipartite and bipartite multigraphs with both non-overlapping and overlapping communities. The model extends the previous standard SBM, degree corrected SBM, bipartite SBM, and SBM modifications for overlapping communities in a uniform way. We presented the results of graph generations covering significant combinations. Thus, we could assess how the stochastic block model detection methods performed and highlight their strengths and weaknesses. We are currently investigating an appropriate method design for the proposed model detection. We believe that the proposed probability based model will robustly treat any noisy and missing items.

Acknowledgement. Sponsored by the project for GAČR, No. 16-072105: Complex network methods applied to ancient Egyptian data in the Old Kingdom (2700–2180 BC).

References

1. Ball, B., Karrer, B., Newman, M.E.J.: An efficient and principled method for detecting communities in networks. CoRR **abs/1104.3590** (2011). <http://arxiv.org/abs/1104.3590>
2. Bastian, M., Heymann, S., Jacomy, M.: Gephi: an open source software for exploring and manipulating networks (2009). <http://www.aaai.org/ocs/index.php/ICWSM/09/paper/view/154>
3. Blondel, V.D., Guillaume, J.L., Lambiotte, R., Lefebvre, E.: Fast unfolding of communities in large networks. J. Stat. Mech. Theory Exp. **2008**(10), P10,008 (2008)
4. Du, N., Wang, B., Wu, B., Wang, Y.: Overlapping community detection in bipartite networks. In: 2008 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology, vol. 1, pp. 176–179 (2008). <https://doi.org/10.1109/WIAT.2008.98>
5. Dulíková, V.: The reign of king Nyuserre and its impact on the development of the Egyptian state. A multiplier effect period during the Old Kingdom. Ph.D. thesis, Charles University in Prague, Faculty of Arts, Czech Institute of Egyptology (2016)
6. Fortunato, S.: Community detection in graphs. Phys. Rep. **486**(3–5), 75–174 (2010). <https://doi.org/10.1016/j.physrep.2009.11.002>

7. Fruchterman, T.M.J., Reingold, E.M.: Graph drawing by force-directed placement. *Softw. Pract. Exp.* **21**(11), 1129–1164 (1991). <https://doi.org/10.1002/spe.4380211102>
8. Heeger, D.: Poisson model of spike generation (2000). <http://www.cns.nyu.edu/~david/handouts/poisson.pdf>
9. Holland, P.W., Laskey, K.B., Leinhardt, S.: Stochastic blockmodels: first steps. *Soc. Netw.* **5**(2), 109–137 (1983). [https://doi.org/10.1016/0378-8733\(83\)90021-7](https://doi.org/10.1016/0378-8733(83)90021-7)
10. Karrer, B., Newman, M.E.J.: Stochastic blockmodels and community structure in networks. *Phys. Rev. E* **83**, 016,107 (2011). <https://doi.org/10.1103/PhysRevE.83.016107>
11. Larremore, D.B., Clauset, A., Jacobs, A.Z.: Efficiently inferring community structure in bipartite networks. *CoRR* **abs/1403.2933** (2014). <http://arxiv.org/abs/1403.2933>
12. Lehmann, S., Schwartz, M., Hansen, L.K.: Biclique communities. *Phys. Rev. E* **78**, 016,108 (2008). <https://doi.org/10.1103/PhysRevE.78.016108>. <https://link.aps.org/doi/10.1103/PhysRevE.78.016108>
13. Newman, M.E.J., Girvan, M.: Finding and evaluating community structure in networks. *Phys. Rev. E* **69**(2), 026,113+ (2003). <https://doi.org/10.1103/physreve.69.026113>
14. Yan, X., et al.: Model selection for degree-corrected block models. *J. Stat. Mech. Theory Exp.* **2014**(5), P05,007 (2014)
15. Yang, J., Leskovec, J.: Overlapping community detection at scale: a nonnegative matrix factorization approach. In: Proceedings of the Sixth ACM International Conference on Web Search and Data Mining, pp. 587–596. ACM (2013)



Communities as Well Separated Subgraphs with Cohesive Cores: Identification of Core-Periphery Structures in Link Communities

Frank Havemann¹(✉), Jochen Gläser², and Michael Heinz¹

¹ Institut für Bibliotheks- und Informationswissenschaft, Humboldt-Universität zu Berlin, Germany

{frank.havemann,michael.heinz}@ibi.hu-berlin.de

² Center for Technology and Society, TU Berlin, Berlin, Germany

Abstract. Communities in networks are commonly considered as highly cohesive subgraphs which are well separated from the rest of the network. However, cohesion and separation often cannot be maximized at the same time, which is why a compromise is sought by some methods. When a compromise is not suitable for the problem to be solved it might be advantageous to separate the two criteria. In this paper, we explore such an approach by defining communities as well separated subgraphs which can have one or more cohesive cores surrounded by peripheries. We apply this idea to link communities and present an algorithm for constructing core-periphery structures in link communities and first test results.

Keywords: Networks · Communities · Link clustering
Core and periphery

1 Introduction

Communities in networks are commonly considered as subgraphs with dense internal but sparse external connections (cf., e.g., [3, 9], and [8]). In other words, a community should be a highly cohesive subgraph which is well separated from the rest of the network. Maximum cohesion of nodes is reached in fully connected subgraphs (cliques), maximum separation for subgraphs without external connections (components). In many practical cases the two essential features of communities cannot be maximised both at the same time, which is why a compromise is sought by some methods for the construction of communities (for reviews of community finding, see [8, 26], and [2]). When a compromise is not appropriate for the problem to be solved, it might be advantageous to separate the two criteria. In this paper, we explore such an approach by defining communities as well separated subgraphs which can have one ore more cohesive cores surrounded by less cohesive peripheries. Due to the size bias of cohesion measures

we prefer separation as the defining feature. Methods for finding core-periphery structures were reviewed by Csermely et al. [6].

We apply this idea to link communities in networks. Link clustering was introduced by Evans and Lambiotte [7] and by Ahn et al. [1]. The aim of our paper is to operationalise the argument for separating the evaluation of cohesion and separation for link communities, and to propose an algorithm that identifies core-periphery structures in well separated link communities. In Sect. 2 we discuss the conceptual problems of simultaneously maximising cohesion and separation. In Sect. 3 a method for determining core-periphery structures of link communities is derived. In Sect. 4 it is applied to results of a local link clustering exercise [10] and to the karate-club network [30].

2 Cohesion and Separation

If communities in networks are considered as highly cohesive and well separated subgraphs, all cliques without external links are ideal but trivial and very rare communities of nodes. In nearly all cases we have to content ourselves with imperfect communities. In this section, we discuss three problems of community construction, namely (A) the existence of real-world problems for which the maximisation of cohesion is likely to create artifacts, (B) the necessity to compromise between maximising cohesion and separation for all other real-world problems, and (C) the size bias of most measures of cohesion.¹

(A) Maximisation of cohesion can produce artifacts: Some real-world problems are represented by communities that contain the boundaries of other communities. When communities form a hierarchy, larger communities contain smaller ones. In this case, the smallest communities of a hierarchy can be very cohesive but supercommunities can only be cohesive if their subcommunities are not very well separated [20, 21]. The same problem occurs when communities overlap pervasively (i.e. not only in boundary nodes). Here, too, boundaries of one community run through another, thereby lowering the cohesion and violating the demand that communities should be hard to split [11, 13, 28]. Applying a cohesion-maximising algorithm to these problems might lead to important communities being excluded from consideration, or to artefactual communities being included.

(B) Finding an appropriate compromise between cohesion and separation may be difficult: The best way to compromise between cohesion and separation may be difficult to determine. If whole networks are partitioned into disjoint communities, this compromise is often built into the algorithm and cannot be altered (e.g. in the case of modularity-maximising algorithms [16]). If an algorithm evaluates single communities, cohesion and separation are unlikely to

¹ Conductance and other often used evaluation functions which rather measure separation than cohesion (cf. discussion in Sect. 2) also favour communities of certain sizes but their size dependence is not as strict as for cohesion measures. Therefore, it does not prevent algorithms based on them to find communities of all sizes [13]. We thank a reviewer for pointing us to this issue.

be maximal for the same subgraph, which raises the question as to how such a compromise should look like. Pizzuti [18] introduced a bi-objective optimisation which allows to choose a compromise between the two features. She used a genetic algorithm to maximise internal and to minimise external connectivity of a partition's communities. Kannan et al. [11] proposed to solve this problem by setting a minimum level for one feature and maximising the other under this minimum condition. However, these strategies are unlikely to work for all real-world problems.

(C) Most cohesion measures have a size bias: One of the major – and so far underappreciated – problems of community construction is the size bias of most cohesion measures. In general, *global* cohesion of a set can be measured by the ratio of the number of directly connected element pairs to the theoretical maximum of this number.² In imperfect communities, links may be so unevenly distributed that the communities contain well separated and cohesive subgraphs. This is why some authors demand that in addition to being highly cohesive, a community should be “hard to split” and measure cohesion by internal conductance, i.e. the minimal conductance of all possible splits [11, 13, 28]. However, the calculation of internal conductance depends on the identification of the best split, which creates significant problems for community construction. Furthermore, this demand cannot be upheld for the problems described under (A) above. For some practical problems it is sufficient that members of communities have a high *local* cohesion; cf., e.g., [27]. Local cohesion of nodes can be measured by their degree or the local clustering coefficient. Most cohesion measures that are solely based on network topology are scaling with size in sparse networks. Link density tends to be smaller for larger subgraphs (cf. [23], p. 50).³ When average internal degree is used to evaluate the cohesion of a community the opposite size bias is observed. For scale-free networks the average clustering coefficient decreases with size [20].⁴

A further option to measure cohesion seems to be to relate the sum of internal degrees $k_{\text{in}}(C)$ of nodes in C to the sum of their total degrees $k(C) = k_{\text{in}}(C) + k_{\text{out}}(C)$. This ratio equals the probability that a random walker found

² For the cohesion of nodes in monopartite topological graphs this ratio equals *link density* which is maximal for cliques. For link communities, we derived an analogue to link density named *connectedness density* of links (see App., p. 227). It is maximal for star subgraphs.

³ Similar to the link density of nodes, connectedness density of links scales with size: larger link communities tend to have lower values.

⁴ Our discussion here is limited to topological networks. When a suitable measure of node distance can be defined, we transcend network topology and global cohesion can be defined as some aggregate of distances between a subgraph's nodes. A suitable measure would be a distance which is not maximal for unlinked nodes. If all unlinked nodes have the same distance the ends of a long chain would have the same distance as two nodes in the chain with a third node between them. Distance should also not strongly depend on the position of individual links which is the case for length of the shortest path and its derivatives.

in node community C does stay within C in the next step and is therefore called *persistence probability* [17, 22]:

$$P_{\text{pers}}(C) = \frac{\sum_{i \in C} k_i^{\text{in}}(C)}{\sum_{i \in C} k_i(C)} = \frac{k_{\text{in}}(C)}{k(C)}. \quad (1)$$

$P_{\text{pers}}(C)$ appears to measure cohesion of nodes but is insensitive to the distribution of connections. Two subgraphs with the same number of external and of internal links have the same persistence probability but can have a rather different cohesion of nodes measured by their link density or their internal conductance. The random walker needs many internal and a few external links to walk within C for a while but the internal structure is not relevant. For example, C can also be a chain of nodes or even be disconnected. Thus, persistence probability $P_{\text{pers}}(C)$ measures separation rather than cohesion as defined here. With increasing external degree $k_{\text{out}}(C)$ persistence probability decreases. This can be made explicit when we rewrite it: $P_{\text{pers}}(C) = 1 - k_{\text{out}}(C)/k(C)$, where $k_{\text{out}}(C)/k(C)$ equals the probability of a random walker found in C to leave the community in the next step, also called *escape probability* and denoted here by $P_{\text{esc}}(C)$ [8], cf. App., p. 228.

Piccardi [17] pointed out that $P_{\text{pers}}(C)$ is related to the definition of *communities in the weak sense* given by Radicchi et al. [19] with the criterion $k_{\text{in}}(C) > k_{\text{out}}(C)$. If $P_{\text{pers}}(C) > 1/2$ then this criterion is fulfilled. The strong definition of communities demands that each node has to have more internal than external links. If $k_{\text{out}}(C)$ is small, both definitions tolerate communities which can be split. Consider, for example, a subgraph comprising two 4-cliques with one external link per clique and one link between both cliques. Both cliques are also communities in the strong and weak sense. Escape probability $P_{\text{esc}}(C) = k_{\text{out}}(C)/k(C)$ is used in cut-based measures of separation as *conductance* and *normalised cut*. In Appendix, p. 228, we discuss these measures and add *normalised node-cut*, a measure of separation for link communities we proposed recently [10].

In summary, maximising cohesion of communities in topological graphs is difficult when (A) maximising cohesion, (B) compromising between cohesion and separation, or (C) a size bias of cohesion measures may lead to the disregard of subgraphs that are important to solving specific real-world problems. In these cases, separation and cohesion can be measured for different objects. This can be achieved if we introduce the notion of cohesive community cores. Like whole networks, communities can have a core-periphery structure [12, 29]. A cohesive core can be linked to many peripheral nodes, which means that it is not well separated. Separation can be improved by including the core's periphery into the community which simultaneously diminishes its internal cohesion. In order to realise separate measurements, we propose to consider communities in networks as well separated connected subgraphs and to reserve the feature of high cohesion for community cores. In other words, we propose to change the common notion of communities in networks when (A), (B) or (C) apply but to retain separation and cohesion as (now separated) aims of optimisation. In the language of Kannan

et al. [11] we maximise the subgraph's separation while the minimum condition for its cohesion is its connectedness. We then look for cohesive cores of communities. We propose to define a cohesive core and its periphery not in absolute terms but as a sequence of nested subgraphs with decreasing cohesion. Such a continuous approach was already introduced by Borgatti and Everett [5]. For communities with a power-law distribution of degrees (called *hyperbolic* communities) core-periphery structures have also been defined in a continuous manner [15].⁵

3 Core-Periphery Structures in Link Sets

There are several methods for finding cohesive cores of graphs or node communities [5, 31].⁶ We construct core-periphery structures in link communities as nested subgraphs with decreasing connectedness density (see App., p. 227). We start from subgraphs with local maxima of local density of links which are sufficiently distant from other subgraphs with local maxima analogously to methods for node-community finding proposed by Liu et al. [14] and by Wang et al. [25]. The simplest way to translate the local node density used by these authors into the world of link clustering is to define local density of links as the number of neighbouring links attached to a node. Thus, local density of links equals local node density. Large stars as link sets with maximal connectedness density have then also a high local density of links. Therefore, we start from the largest star of a link community L for constructing its core-periphery structures.

Our aim is to determine core-periphery structures (named *towns*, for short) in a given connected subgraph induced by link set L . We define a town as an ordered cluster of stars where two stars are never indirectly connected via smaller stars only. Two stars are directly connected if they share a link or one of their outer nodes. A star is connected to a town if it shares a link or a minimum number of outer nodes with the set of town stars of equal or larger size; otherwise it becomes the centre of an independent town. The minimum number of outer nodes is determined by a resolution parameter q with $0 \leq q < 1$ which is used as a minimum threshold of relative overlap for a star to be attached to a town. If $q = 0$ one common node of star and town is enough to unite both link sets. If $q = 1/2$ more than half of the star's outer nodes have to be already inside the town. If there are two or more towns a star is connected to it is split and its parts are attached to these towns.

The algorithm for finding cores and peripheries in link communities (CPLC algorithm) can be described as follows (cf. Algorithm 1). All star subgraphs of the community are ranked with regard to their size. To construct a town T it is initialised by the largest star. The next star on the rank list is attached to town T with node set N_T if the number of the star's outer nodes shared

⁵ We are grateful to the reviewer who alerted us to this work.

⁶ The simple k -core algorithm does not decide to which core a peripheral node belongs if there are multiple cores [12]. This unwanted feature remains after the translation of k -cores for link communities. We therefore abandoned this algorithm.

Algorithm 1 Pseudocode of CPLC (cores and peripheries of link communities)

```

all stars of a (sub)graph are centre candidates;
the largest star is a centre (select one randomly from two or more largest stars);
remove it from set of centre candidates;
initialise its town with the centre itself;
while there are centre candidates do
    if the largest centre candidate  $j$  shares at least one link or more than  $qk_j^{\text{in}}$  of its
    outer nodes with any town then
        if there is only one such town then
            centre candidate  $j$  is united with the town;
        else
            the links of  $j$  to a town are united with the town;
            the remaining links of  $j$  are united with all towns with overlap;
        end if
    else
        candidate  $j$  becomes a new centre;
        initialise its town with the centre itself;
    end if
    remove stars which have all links within towns from set of centre candidates;
    skip the next centre candidate(s) which share all links to towns with towns;
end while

```

with the town fulfils the minimum condition given by resolution parameter q : $|\text{adj}(j) \cap N_T| > qk_j^{\text{in}}$, where $\text{adj}(j)$ denotes the adjacent nodes of the star's central node j . A direct link between two star centres is also a sufficient condition for being included in the town. If a star could be united with two or more existing towns then we add to each town its links with this town. Its remaining links are united with all towns involved. Then we delete all (mostly small) stars from the list of candidates which now have no links outside any town. We skip candidate stars that share all links to towns with these towns. We found this feature useful in our experiments with link communities in a nearly bipartite network where different kinds of nodes can be centres of candidate stars. Skipping these stars does also work in the unipartite karate-club network.

The number of towns obtained depends on resolution q . Instead of arbitrarily setting parameter q we explore its whole range by starting with minimal resolution $q = 0$ and increasing it stepwise to obtain more towns. At $q = 0$, the subgraph has the minimal number of towns. While $q < 1$, we then recursively increase it to a value at which it is possible to obtain at least one more town. Therefore, the next value of q is set to the value of the smallest relative node overlap of any star and the town with which it was united. This guarantees that in the next run of CPLC this star and all stars with the same relative overlap to any of the towns are not united with these towns.

To select resolution levels at which relatively well separated towns are obtained we calculated normalised node-cut $\Psi(L)$ of link set L for each town (cf. Eq. 5 in App.; here E is the link community analysed). Because towns are

not optimised with regard to separation we also calculated function Ψ for L subtracted by all links in overlaps between towns. For each town we choose the better (lower) of both values and evaluated the resolution level with the worst (largest) Ψ of any town. If there are two or more levels with same worst Ψ and same number of towns, we selected the lowest level with minimum link overlap between towns.

4 Experiments

The karate club analysed by Zachary [30] has only one town for lowest resolution level $q = 0$ with node 34 as the centre. We obtain two towns for resolution $q \geq 1/4$ with centres 1 and 34. For $q \geq 4/9$ their link overlap reduces to three links: (3, 9), (3, 14), and (9, 31). Besides the nodes of these three links, the two towns overlap in nodes 20 and 32. Thus, the two towns are compatible with the final splitting of the karate club due to conflicts between the two leaders 1 and 34. For the town with centre node 1, $\Psi = 0.1770$ and $\Psi = 0.1723$ for the other town. A better value $\Psi = 0.1638$ (in fact the best we have found) can be obtained for a disjoint link splitting of the karate-club network where the towns' link overlap is split up (s. Fig. 1). With a statistical approach to link clustering using generative network models Ball et al. [4] obtained exactly the same two link communities. Further comparisons have to be published elsewhere due to lack of space.

Searching for link communities with locally minimal Ψ -values in a nearly bipartite network of 14,770 papers published 2010 in astronomy and astrophysics journals (including also geophysical papers) and their cited sources we found 127 overlapping link communities which cover the network and form a poly-hierarchy of topics [10]. We clustered citation links rather than papers because papers are often dealing with more than one topic and citation links are often monothematic. Figure 2 shows the small community h80 which is an example of a well separated subgraph ($\Psi = 0.0458$) that is not very cohesive (it can be split into two subcommunities). Here we have two towns already at zero

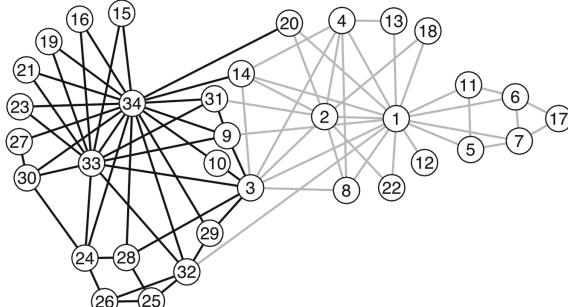


Fig. 1. Network of 34 karate fighters organised in a club and observed by Zachary (1977) in their connections outside the club. The split into black and grey links has minimal Ψ -value (cf. text).

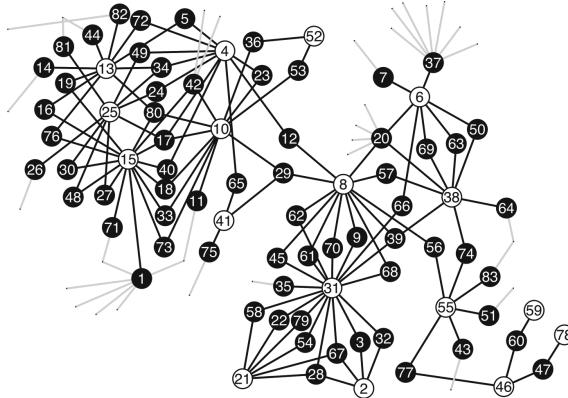


Fig. 2. Community of citation links between 17 papers (white nodes) and 66 cited sources (black nodes). Grey links are not community members but neighbours. A minimal Ψ -value within the subgraph is obtained if the two black nodes 12 and 29 in the centre of the subgraph are cut through (cf. text).

resolution. Increasing it to $q = 1/10$ causes their overlap to decrease from 16 to 2 links ((29, 41) and (41, 75)) and Ψ of towns within the subgraph reaches a minimal value of 0.0187 if the two overlapping links are deleted from the town on the right-hand side. For $q \geq 1/3$ we find solutions with four and more towns but relatively bad separation (worst $\Psi \geq 0.1313$). The centres of the two towns are the large stars with central nodes 15 and 31. The two towns correspond to two communities we had found with our search algorithm [10]. They represent different topics: The ten papers in the town on the right-hand side of the subgraph deal with lightnings and similar electromagnetic phenomena in the atmosphere (h95), the seven papers on the left-hand side mainly deal with effects of seismic activities measured in the ionosphere (h98). Communities h95 and h98 correspond to subtopics of h80. This illustrates the hierarchy problem discussed in Sect. 2 in point (A).

The number of stars and the number of towns both increase with the number of links m . We therefore expect run-time of CPLC to scale with m^2 which is confirmed by experiments with the 127 communities mentioned above and the whole network with $m = 536,020$. Due to space limitation we cannot present further statistics of results.

5 Summary and Discussion

If a real-world problem is likely to be represented by a hierarchy of communities or overlapping communities or by communities of varying and unknown sizes, or if it is difficult to determine the best compromise between cohesion and separation, it seems advantageous to separate the maximisation of cohesion and separation. In this paper, we propose a strategy that starts from communities as well separated subgraphs and identifies cohesive cores of such subgraphs.

We applied this strategy to the analysis of link communities but formally the algorithm proposed here operates on any connected graph or subgraph. To determine core-periphery structures as nested subgraphs with decreasing cohesion in a link community we start from local maxima of local link density, i.e., from the largest stars. The examples presented here demonstrate that the algorithm we have tested is able to separate core-periphery structures (*towns*) if there are two or more such structures in a (sub)graph. Our next task is to evaluate each town with regard to their distinctness. If all stars have nearly the same size it would be difficult to speak of centre and periphery. A further task is to assess the correspondence of core-periphery structures with the real-world problem of research topic detection. Towns of communities are expected to correspond to sub-topics of the larger topic represented by the community. How differ topics of core and peripheral papers of a topical community of citation links?

Beside high cohesion, another often assumed feature of cores is their network centrality [6, p. 94], e.g., indicated by low average distance to peripheral nodes. We are also interested in non-central cores of link communities.

Towns of the whole graph can be used as seeds for local link clustering. Towns of communities found can recursively serve as seeds for finding smaller communities.

Appendix

Cohesion of Link Sets

Let $k_i^{\text{in}}(L)$ be the number of links in set L attached to node i , also called its internal degree. The number of links in L a link (i, j) is connected to equals $k_i^{\text{in}}(L) + k_j^{\text{in}}(L) - 2$. For the total number of (ordered) pairs of directly connected links in L we find

$$N(L) = \sum_{(i,j) \in L} (k_i^{\text{in}}(L) + k_j^{\text{in}}(L) - 2) = \sum_{i=1}^n k_i^{\text{in}}(L) \cdot (k_i^{\text{in}}(L) - 1). \quad (2)$$

In the sum each node i occurs $k_i^{\text{in}}(L)$ times with connections from one link to $k_i^{\text{in}}(L) - 1$ others. $N(L)$ is an absolute measure for cohesion of link sets. It is not maximal if the $|L|$ links form a clique of nodes. Indeed, for a clique of four nodes connected by six links we have $N(L) = 24$. If the six links form a star we obtain a higher value $N(L) = 30$. In the star graph all links are directly connected but in cliques of at least four nodes not. This corresponds to the fact that the line graph of a star is a clique. If L has the form of a star and c denotes its central node then $k_c^{\text{in}}(L) = |L|$. For all other nodes we have $k_i^{\text{in}}(L) = 1$, i.e., $N(L) = |L|(|L| - 1)$. Link sets are maximally connected if all their links are directly connected by a node, i.e., stars are maximally connected link sets.

We can define a relative measure of cohesion of link sets by relating absolute node connectedness of links $N(L)$ to its maximum reached by stars. That means, as *connectedness density* D of a link set L we can define

$$D(L) = \frac{\sum_i k_i^{\text{in}}(L)(k_i^{\text{in}}(L) - 1)}{|L|(|L| - 1)}. \quad (3)$$

This measure is useful for both one-mode and also for two-mode networks (where there are no cliques). In both types of networks stars are the most cohesive link sets. Analogously to this measure, link similarity as defined by Ahn et al. [1] is not maximal for all link pairs in a clique of n nodes but in a star of $n(n - 1)/2$ links.

The Random Walker and Separation of Communities

Supposing that a random walker is on any node in set C (in an unweighted and undirected network), his probability to be on node i equals $k_i / \sum_{i \in C} k_i$. He leaves C in the next step with probability $k_i^{\text{out}}(C) / k_i$. Then his probability to leave C from node i is the product of both probabilities $k_i^{\text{out}}(C) / \sum_{i \in C} k_i$ and the probability to leave C from any node (escape probability) is $P_{\text{esc}}(C) = \sum_{i \in C} k_i^{\text{out}}(C) / \sum_{i \in C} k_i$.

Escape probability $P_{\text{esc}}(C) = k_{\text{out}}(C) / k(C)$ equals *conductance* of C for $k(C) < m$ with m the number of all links [8]. For $k(C) > m$ cut $k_{\text{out}}(C)$ is normalised by $k(V - C)$ (with V the set of all vertices) because subgraphs larger than half the whole graph tend to have smaller cuts $k_{\text{out}}(C) = k_{\text{out}}(V - C)$. A smoother normalisation which takes this tendency into account is achieved in *normalised cut* defined by Shi and Malik [24] as

$$\Phi(C) = \frac{k_{\text{out}}(C)}{k(C)} + \frac{k_{\text{out}}(C)}{k(V - C)}. \quad (4)$$

In the case of link communities we have to cut not links but nodes to separate a link set L from the rest of the graph. *Normalised node-cut* $\Psi(L)$ is a measure of separation of link communities derived from normalised cut $\Phi(C)$ by us [10]. It is given by

$$\Psi(L) = \frac{\sigma(L)}{k_{\text{in}}(L)} + \frac{\sigma(L)}{k_{\text{in}}(E - L)}, \quad \text{with } \sigma(L) = \sum_{i=1}^n \frac{k_i^{\text{in}}(L)k_i^{\text{out}}(L)}{k_i}, \quad (5)$$

where i runs through all n nodes but $k_i^{\text{in}}(L) = 0$ for all nodes which are not attached to a link in L . Set E includes all m edges and n is the number of all nodes. Note, that $\sigma(L) = \sigma(E - L)$ and $k_{\text{in}}(E - L) = 2m - k_{\text{in}}(L)$. Evans and Lambiotte [7] introduced a random walker who jumps from a link to one of its nodes with probability 1/2 and then chooses one of the links attached to this node. The ratio $\sigma(L) / k_{\text{in}}(L)$ equals the escape probability of such a *link-node-link random walker*: The probability of a link-node-link random walker to start at any link in set L and to arrive on node i is $p_i(L) = k_i^{\text{in}}(L) / \sum_{i=1}^n k_i^{\text{in}}(L)$. That means, his probability to leave L from i is $p_i(L)k_i^{\text{out}}(L) / k_i$ and the escape probability is $P_{\text{esc}}(L) = \sigma(L) / k_{\text{in}}(L)$, where $k_{\text{in}}(L) = \sum_{i=1}^n k_i^{\text{in}}(L)$ and $\sigma(L) = \sum_{i=1}^n k_i^{\text{in}}(L)k_i^{\text{out}}(L) / k_i$ (cf. Eq. 5). This is a new derivation of function $\sigma(L)$ used by us for defining normalised node-cut $\Psi(L)$ which now appears as

$\Psi(L) = P_{\text{esc}}(L) + P_{\text{esc}}(E - L)$ with $\max(\Psi(L)) = 2$ and not 1, as stated by us. Both probabilities reach a maximum of 1 for a ring graph where each second link is in L .

References

1. Ahn, Y.Y., Bagrow, J.P., Lehmann, S.: Link communities reveal multi-scale complexity in networks. *Nature* **466**, 761–764 (2010)
2. Amelio, A., Pizzuti, C.: Overlapping community discovery methods: a survey. In: *Social Networks: Analysis and Case Studies*, p. 105 (2014)
3. Bagrow, J.P., Bollt, E.M.: Local method for detecting communities. *Phys. Rev. E* **72**(4), 046,108 (2005). <https://doi.org/10.1103/PhysRevE.72.046108>
4. Ball, B., Karrer, B., Newman, M.E.J.: Efficient and principled method for detecting communities in networks. *Phys. Rev. E* **84**(3), 036,103 (2011). <https://doi.org/10.1103/PhysRevE.84.036103>
5. Borgatti, S.P., Everett, M.G.: Models of core/periphery structures. *Soc. Netw.* **21**(4), 375–395 (2000)
6. Csermely, P., London, A., Wu, L.Y., Uzzi, B.: Structure and dynamics of core/periphery networks. *J. Complex Netw.* **1**(2), 93–123 (2013). <https://doi.org/10.1093/comnet/cnt016>
7. Evans, T.S., Lambiotte, R.: Line graphs, link partitions, and overlapping communities. *Phys. Rev. E* **80**(1), 16,105 (2009)
8. Fortunato, S.: Community detection in graphs. *Phys. Rep.* **486**, 75–174 (2010)
9. Girvan, M., Newman, M.E.J.: Community structure in social and biological networks. *PNAS* **99**, 7821–7826 (2002)
10. Havemann, F., Gläser, J., Heinz, M.: Memetic search for overlapping topics based on a local evaluation of link communities. *Scientometrics* 1–30 (2017). <https://doi.org/10.1007/s11192-017-2302-5>
11. Kannan, R., Vempala, S., Vetta, A.: On clusterings: good, bad and spectral. *J. ACM* **51**(3), 497–515 (2004). <https://doi.org/10.1145/990308.990313>
12. Kojaku, S., Masuda, N.: Finding multiple core-periphery pairs in networks. *Phys. Rev. E* **96**(5), 052,313 (2017)
13. Leskovec, J., Lang, K.J., Mahoney, M.: Empirical comparison of algorithms for network community detection. In: *Proceedings of the 19th International Conference on World Wide Web, WWW '10*, pp. 631–640, New York (2010). <https://doi.org/10.1145/1772690.1772755>
14. Liu, D., Su, Y., Li, X., Niu, Z.: A novel community detection method based on cluster density peaks. In: *Natural Language Processing and Chinese Computing. Lecture Notes in Computer Science*, pp. 515–525. Springer (2017). https://doi.org/10.1007/978-3-319-73618-1_43
15. Metzler, S., Günemann, S., Miettinen, P.: Hyperbolae are no hyperbole: modelling communities that are not cliques. In: *Data Mining (ICDM), 2016 IEEE 16th International Conference on Data Mining*, pp. 330–339. IEEE (2016)
16. Newman, M.E.J., Girvan, M.: Finding and evaluating community structure in networks. *Phys. Rev. E* **69**, 026,113 (2004)
17. Piccardi, C.: Finding and testing network communities by lumped Markov chains. *PloS one* **6**(11), e27,028 (2011)
18. Pizzuti, C.: A multi-objective genetic algorithm for community detection in networks. In: *21st IEEE International Conference on Tools with Artificial Intelligence*, pp. 379–386. IEEE (2009)

19. Radicchi, F., Castellano, C., Cecconi, F., Loreto, V., Parisi, D.: Defining and identifying communities in networks. *PNAS* **101**, 2658–2663 (2004)
20. Ravasz, E., Barabási, A.L.: Hierarchical organization in complex networks. *Phys. Rev. E* **67**(2), 026,112 (2003). <https://doi.org/10.1103/PhysRevE.67.026112>
21. Rezvani, M., Wang, Q., Liang, W.: Fast Algorithm for Detecting Cohesive Hierarchies of Communities in Large Networks, pp. 486–494. ACM (2018). <https://doi.org/10.1145/3159652.3159704>. <http://dl.acm.org/citation.cfm?doid=3159652.3159704>
22. Rossa, F.D., Dercole, F., Piccardi, C.: Profiling core-periphery network structure by random walkers. *Sci. Rep.* **3**, 1467 (2013). <https://doi.org/10.1038/srep01467>
23. Schaeffer, S.E.: Graph clustering. *Comput. Sci. Rev.* **1**(1), 27–64 (2007). <https://doi.org/10.1016/j.cosrev.2007.05.001>
24. Shi, J., Malik, J.: Normalized cuts and image segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.* **22**(8), 888–905 (2000). <https://doi.org/10.1109/34.868688>
25. Wang, X., Liu, G., Li, J., Nees, J.P.: Locating structural centers: a density-based clustering method for community detection. *PLOS ONE* **12**(1), e0169,355 (2017). <https://doi.org/10.1371/journal.pone.0169355>
26. Xie, J., Kelley, S., Szymanski, B.K.: Overlapping community detection in networks: the state-of-the-art and comparative study. *ACM Comput. Surv.* **45**(4), 43:1–43, 35 (2013). <https://doi.org/10.1145/2501654.2501657>
27. Xu, X., Yuruk, N., Feng, Z., Schweiger, T.A.J.: SCAN: a structural clustering algorithm for networks. In: Proceedings of the 13th ACM SIGKDD, KDD '07, pp. 824–833. ACM, New York, NY, USA (2007). <https://doi.org/10.1145/1281192.1281280>
28. Yang, J., Leskovec, J.: Defining and evaluating network communities based on ground-truth. *Knowl. Inf. Syst.* **42**(1), 181–213 (2013). <https://doi.org/10.1007/s10115-013-0693-z>
29. Yang, J., Leskovec, J.: Overlapping communities explain core-periphery organization of networks. *Proc. IEEE* **102**(12), 1892–1902 (2014)
30. Zachary, W.: An information flow model for conflict and fission in small groups. *J. Anthropol. Res.* **33**(4), 452–473 (1977)
31. Zhang, X., Martin, T., Newman, M.E.J.: Identification of core-periphery structure in networks. *Phys. Rev. E* **91**(3), 032,803 (2015). <https://doi.org/10.1103/PhysRevE.91.032803>. <https://link.aps.org/doi/10.1103/PhysRevE.91.032803>



Ensemble Clustering for Graphs

Valérie Poulin and François Théberge^(✉)

Tutte Institute for Mathematics and Computing, Ottawa, Canada
vpoulin@gmail.com, theberge@ieee.org

Abstract. We propose a new ensemble clustering algorithm for graphs (ECG) which is based on the Louvain algorithm and the concept of consensus clustering. We validate our approach by replicating a recently published study comparing graph clustering algorithms over artificial networks, showing that ECG outperforms the leading algorithms from that study. We also illustrate how the ensemble obtained with ECG can be used to quantify the presence of community structure in the graph.

Keywords: Graph · Clustering · Consensus

1 Introduction

Many data-sets are relational in nature, describing interactions between entities, such as friendship networks, communications or geographical co-locations. Most networks that arise in nature exhibit complex structure [11, 18] with subsets of vertices densely interconnected relative to the rest of the network, which we call communities or clusters. Binary relational data-sets are typically represented as graphs $G = (V, E)$, where vertices $v \in V$ represent the entities, and edges $e \in E$ represent the relations between pairs of entities. For analyzing and exploring complex relational data-sets, graph clustering is commonly used.

Graph clustering aims at finding a partition of the vertices $V = C_1 \cup \dots \cup C_l$ into good clusters. This is an ill-posed problem [8], as there is no universal definition of good clusters, leading to a wide variety of graph clustering algorithms [3, 5, 11, 19, 23, 26, 27, 29], with different objective functions.

Over the last decade, several studies were conducted to compare graph clustering algorithms [8, 13, 20, 21, 36]. In a recent study [36], several state-of-the art algorithms implemented in the `igraph` [6] package were compared over a wide range of artificial networks generated via the LFR benchmark [15].

Those studies indicate that the multilevel Louvain (ML) algorithm [3] offers one of the best tradeoff between the quality of the clusters it produces and its speed. There are however some issues with this algorithm: it is unstable, i.e., results from successive runs on the same data can vary a lot, and it suffers from the well-known resolution limit problem [7], i.e., the ML algorithm often yields a coarsening of the true structural clusters. Methods such as re-weighting edges with respect to presence in short cycles [2] do help to improve the stability, but are computationally expensive.

Ensemble clustering, often referred to as consensus clustering, offers a wide selection of algorithms [16, 33] for multivariate data. The general idea behind consensus clustering is to combine several partitions over the same data-set into a final partition. This strategy has been motivated by the success of ensemble methods for supervised learning [4, 10] which, despite their simplicity, are often among the best classification methods. Consensus clustering consists of two steps: (i) the *generation* step, where several partitions of the data is obtained, and (ii) the *integration* step, where the final partition is computed using a consensus function [33]. The consensus function is typically based on co-occurrences of objects in clusters [9, 14, 30] (co-association consensus) or on finding a median partition [16] which maximizes the similarity with all partitions obtained in the generation step.

In this paper, we propose ECG (Ensemble Clustering for Graphs), a graph clustering method based on the concept of co-association consensus clustering. It is similar to other consensus clustering algorithms, in particular [14], but differs in two major points: (1) the choice of an algorithm that alleviates the resolution limit issue for the generation step, and (2) the restriction to endpoints of edges for co-occurrences of vertex pairs, which keeps low computational complexity. We show that this approach identifies very high quality clusters by replicating the study in [36] and comparing ECG against the best performing algorithms. We also show that ECG is stable, and that it reduces significantly the resolution limit problem, yielding a number of clusters very close to the ground truth partition size. Finally, it provides information about the strength of the associations between entities which can be used to determine the presence or absence of communities in the network.

The rest of the paper is organized as follows. We describe ECG algorithm in Sect. 2. Empirical results following the protocol used in [36] are presented in Sect. 3, followed by a discussion of the resolution limit, stability, and parameter selection in Sect. 4. We conclude the paper in Sect. 5.

2 Algorithm

Let $G = (V, E)$ be a graph where $V = \{1, 2, \dots, n\}$ is the set of vertices, and $E \subseteq \{(u, v) \mid u, v \in V, u < v\}$ is the set of edges. We consider undirected graphs. Edges can have weights $w(e) > 0$ for each $e \in E$. For un-weighted graphs, we let $w(e) = 1 \forall e \in E$. The 2-core of a graph G is its maximal subgraph whose vertices have degree at least 2. Let $P_i = \{C_i^1, \dots, C_i^{l_i}\}$ be a partition of V of size l_i . We refer to the C_i^j as *clusters* of vertices. We use $\mathbf{1}_{C_i^j}(v)$ to denote the indicator function for $v \in C_i^j$.

2.1 Review of Multilevel Louvain

The ML algorithm produces a hierarchy of partitions where the level- i partition is a coarsening of the level- $(i-1)$ partition in the hierarchy. The level-0 partition is the partition consisting of all singletons on V . To obtain each subsequent level, the following two phases are repeated:

1. Perform a random permutation of the vertices. For each vertex, evaluate the change in modularity obtained by removing this vertex from its current community, and placing it in the community of its neighbor which yields the highest increase in modularity. Perform community change according to the largest gain in modularity for this vertex (if any) otherwise leave the vertex in its current community. This is repeated until no gain of modularity is achieved.
2. Build a weighted quotient graph using this partition, i.e., each community becomes a vertex.

Those two phases are repeated until no further significant improvement in modularity can be achieved. This algorithm has several advantages: it is very efficient, it returns a hierarchy of partitions, and the number of clusters need not be specified beforehand. However, randomization of the vertex ordering tends to yield very different solutions albeit with similar modularity, so the algorithm is unstable. This statement is true for the final level partition of ML, but it applies even more to the lower levels of the hierarchy. Obtaining several randomized level-1 partitions on a given graph yields many weakly correlated (fine) partitions of the graph. This is what we use in the generation process of ECG.

2.2 ECG Algorithm

The ECG algorithm is a consensus clustering algorithm for graphs. As previously mentioned, its generation step consists of independently obtaining k randomized level-1 ML partitions: $\mathcal{P} = \{P_1, \dots, P_k\}$. Its integration step is performed by running ML on a weighted version of the initial graph $G = (V, E)$. The weights are obtained through co-association, i.e., the weight of an edge $e = (u, v) \in E$ is defined as

$$W_{\mathcal{P}}(u, v) = \begin{cases} w_* + (1 - w_*) \cdot \left(\frac{\sum_{i=1}^k v_{P_i}(u, v)}{k} \right), & (u, v) \text{ in the 2-core of } G \\ w_*, & \text{otherwise} \end{cases} \quad (1)$$

where $0 < w_* < 1$ is some minimum weight and $v_{P_i}(u, v) = \sum_{j=1}^{l_i} \mathbf{1}_{C_i^j}(u) \cdot \mathbf{1}_{C_i^j}(v)$ indicates if the vertices u and v co-occur in the same cluster of P_i or not. Note that all $W_{\mathcal{P}}(e) \in [w_*, 1]$ for $e \in E$ and the minimum weight w_* is assigned to edges whose endpoints are in different clusters for all partitions in \mathcal{P} , or to edges outside the 2-core. We discuss this choice in Sect. 4. This process can be seen as an efficient alternative to re-weighting via the enumeration of short cycles such as triangles, since edges which belong to several short cycles will see their vertices often put in same level-1 ML clusters. It has been shown that using *weak* clustering algorithms in the generation step can produce high quality results [32]. Running a single level of the ML algorithm provides weak learners that do not suffer from the modularity's resolution limit issue as they are early stops of the optimization process. When running the ECG algorithm, the size k of the ensemble and the minimum edge weight w_* are the only parameters that need to be supplied.

3 Comparison Study Revisited

In this section, we re-visit a recently published study of graph clustering algorithms [36], comparing the best performing algorithms from that study with ECG algorithm. In that study, eight different state-of-the-art graph clustering algorithms are evaluated, all of which are included in the *igraph* package. The algorithms are compared on graphs generated with the LFR benchmarks for undirected and unweighted graphs and with non-overlapping communities. The key parameter when generating an LFR graph G is the *mixing parameter* μ , which sets the expected proportion of edges in G for which the two endpoints are in different communities. The parameter μ can be interpreted as the expected proportion of noisy edges in a pure community graph (graph with disconnected communities).

3.1 Comparison Measures

An LFR graph G is generated along with its ground-truth partition P_G . Hence, we can define the accuracy of a partition P on G via a similarity measure $sim(P, P_G)$. Most measures used as graph partition similarities are in fact set partition similarity scores either based on pair counting [1, 12] or on Shannon's information [17, 34, 35]. It is highly recommended to use their adjusted versions [12, 34]. In our study, we report the non-adjusted normalized mutual information (NMI) measure as used in [36], but also the adjusted rand index (ARI) recommended when the number of data points is large relative to the number of parts in the partition [28]. As demonstrated in a recent paper [25], it is also very important to include a *graph-aware* measure when comparing graph partitions. Those measures are shown to be complementary to the set partition measures and superiority in both types of measures is shown to be necessary to assess the superiority of an algorithm over another. As a consequence, we will also report the adjusted graph-aware rand index (AGRI) in our experiments.

3.2 Main Results

Comparisons over several graphs are presented in [36], with graph sizes (number of vertices) ranging from 233 to 31,948. For most experiments, the graph sizes are in the set $n \in \{233, 482, 1000, 3583, 8916, 22186\}$, and the mixing parameter is in the set $\mu \in \{0.03, 0.06, \dots, 0.72, 0.75\}$. For each combination of parameters (n, μ) , we generate 100 LFR graphs with the parameters listed in Table 1 of [36]. We use the *igraph* (version 0.7.1) implementation of the graph clustering algorithms, running in Python version 3.6. For ECG, we use our own implementation [31]. For the similarity measures (NMI, ARI), we use the implementation in *scikit-learn* [22] (version 0.19.0) while we implement the AGRI measure ourselves as described here [24]. For NMI, several normalizations exist [35]; we use the *mean of entropy* as in [36]. Other normalization yield very similar plots and equivalent conclusions.

We are interested in algorithms that can handle larger graphs, not only graphs with a few hundred edges. From [36] and other studies [13, 21], three algorithms stand out as good choices for small as well as large graphs: ML, the Walktrap [23] (WT) algorithm, and the InfoMap (IM) algorithm [29]. We compare ECG to these algorithms.

In Fig. 1, we compare the accuracy of each algorithm over three of the graph sizes n , and over the range of mixing parameter μ . The conclusions are the same for the other choices of n . From these plots, we see that WT and IM do well for small and mid-range values of μ , and ML is generally a better choice, in particular with noisier graphs. The ECG algorithm outperforms all other algorithms, and the difference is more pronounced for larger graphs.

For each algorithm, we compute the ratio of the number of clusters it found \hat{C} , divided by the true number of communities C from the LFR generator. We plot the results in Fig. 2. This is computed for the same set of values n and μ as in Fig. 1. We also compare the empirical mixing parameter $\hat{\mu}$ computed from the clustering obtained with each algorithm. From those plots, we see that ML has a tendency to yield a smaller number of clusters, which would indicate a coarsening of the true communities. WT, on the other hand, tends to return a larger number of clusters, likely indicating that true communities are broken up to produce a refinement of the true partition. IM also yields a larger number of clusters for the larger graphs. The ECG algorithm returns a number of clusters that is remarkably similar to the true value for a wide range of the mixing parameter μ .

Looking at the empirical mixing parameter $\hat{\mu}$, we see that IM is close to the true parameter μ only for less noisy graphs while ML is generally quite good, with ECG producing slightly better results, in particular for larger graphs.

4 Discussion

In the previous section, we saw the clear advantage of the ECG algorithm over a wide range of tests in terms of accuracy, number of communities found, and empirical mixing parameter. In this section, we clarify some reasons for this success by verifying that known issues of ML are indeed solved with ECG. Moreover, we analyze the parameter selection robustness and, finally, we make use of the information contained in ECG weights as a way of determining if networks do contain strong community structure or not.

4.1 Resolution Limit and Stability Issue

Common issues with graph clustering algorithms are that (i) clusters can be broken up, leading to a refinement of the true partition, or (ii) clusters can be amalgamated, leading to a coarsening of the true partition. It was shown [25] that graph-agnostic measures such as the ARI gives high scores for refinements of the true partition, while graph-aware measures such as the AGRI gives high scores for a coarsening of the true partition. ML algorithm tends to produce a

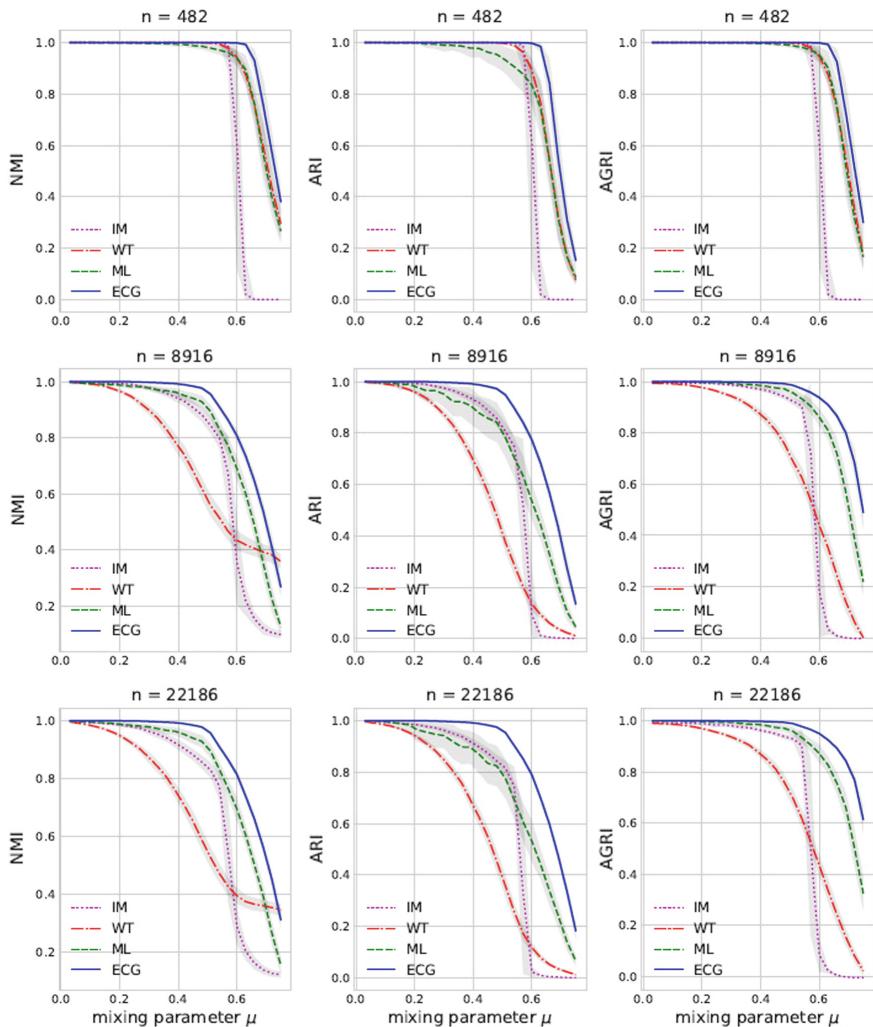


Fig. 1. Accuracy results with respect to the ground truth communities for the 4 graph clustering algorithms considered for the LFR graphs where $n \in \{482, 8916, 22186\}$. Comparison is done via the NMI, ARI and AGRI similarity measures. Each curve shows the mean over 100 different LFR graphs for each value of μ , the shaded area is the standard deviation. In all cases, we see that ECG outperforms all other algorithms.

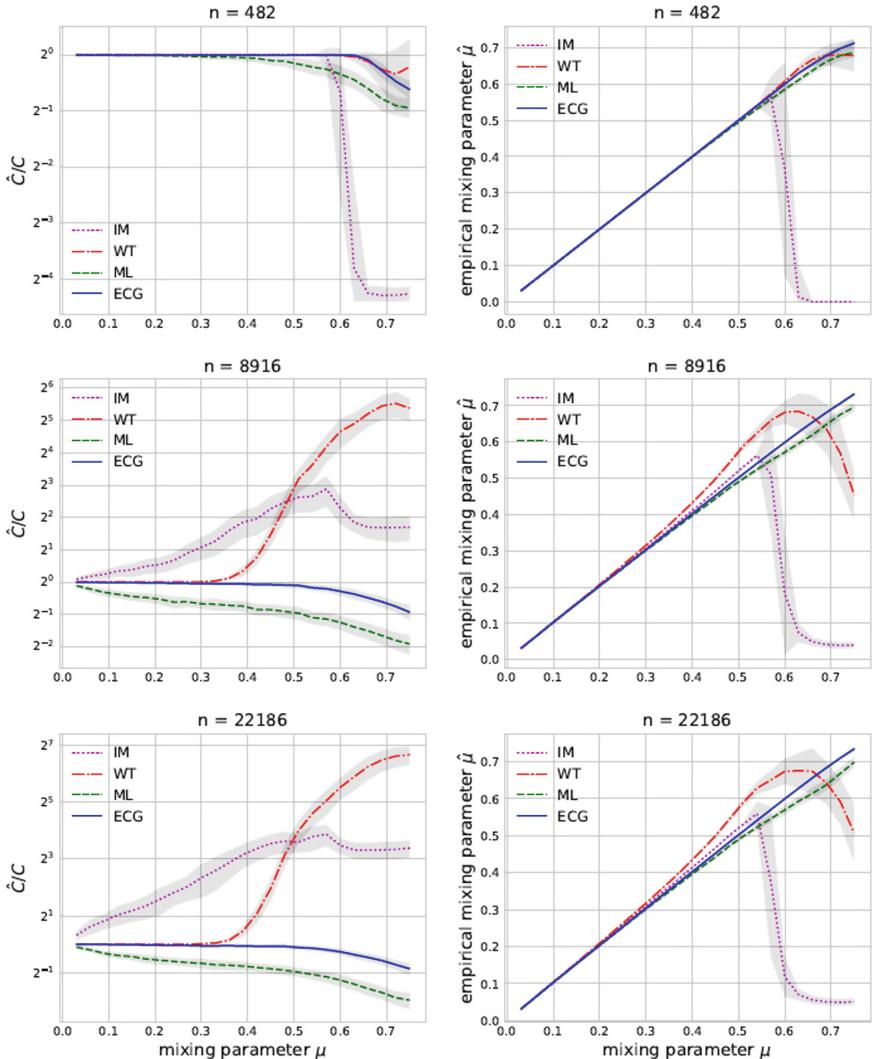


Fig. 2. Comparing the 4 graph clustering algorithms on the LFR graphs with graph sizes $n \in \{482, 8916, 22186\}$. In the left column, we look at the ratio of the number of computed clusters vs true communities, \hat{C}/C for each value of μ . We see that ECG remains very close to the desired value $\hat{C}/C = 1$. In the right column, we compute the empirical mixing parameter $\hat{\mu}$ for each value of μ . We see that the ECG algorithm stays very close to the curve $\hat{\mu} = \mu$. Each curve shows the mean over 100 different LFR graphs for each value of μ , the shaded area is the standard deviation.

coarsening of the true partition, due in part to the well-known resolution limit of the modularity function [7]. On the other hand, if we stop ML algorithm at the first level, which we denote as L1, a refinement of the true partition into several smaller clusters is obtained. Looking at the example in Fig. 3, we see that if we only use ARI, we conclude that L1 is superior to ML, while if we only use AGRI, we conclude the opposite. One of the main advantage of ECG is that the size of the partition it returns is much closer to the true size, thus avoiding breaking up or merging clusters. As a consequence, ECG outperforms ML and L1 with respect to both measures.

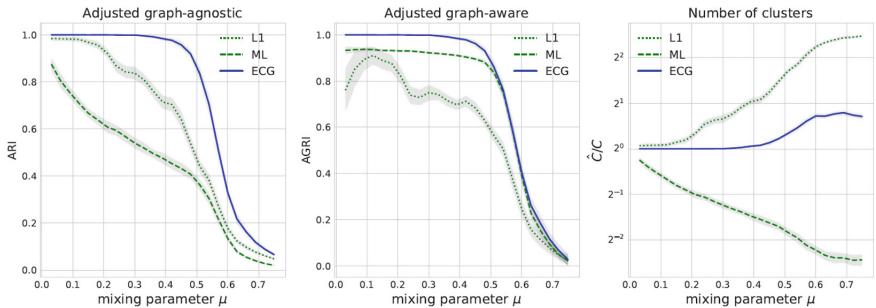


Fig. 3. For each μ , we generated 50 LFR graphs of size 1000 with vertices of degree 8 and community size in the range [10,15]. With respect to the two measures ARI and AGRI, ECG is more accurate than both ML and L1, while conclusions between ML and L1 is different depending on the measure. Looking at the third plot, we see that ML tends to return a smaller number of clusters than the true number, while L1 returns a larger number. On the other hand, ECG is quite close to the true number.

The resolution limit issue is well illustrated by the infamous ring of cliques example, where the n vertices in G are organized in l cliques of size m , wired together as a ring. For some choices of l and m , grouping pairs of adjacent cliques yields a higher modularity value than the natural choice of each clique forming its own cluster [7]. The latter yields higher modularity if and only if $m(m-1) > l-2$. Hence, ring of cliques can easily be designed so that modularity-based algorithms group pairs of cliques. If we consider the weighted ring of cliques with weights of α inside the cliques and β on ring edges, then the condition becomes $m(m-1) > (\beta/\alpha)(l-2)$. If we assume $\beta = 1/n$, then we have that the condition is satisfied for all l, m values as long as $\alpha > 1/(m-1)$. With the choice of L1 as weak learner, the ECG weights defined in (1) will be very close to the minimal value w_* for ring edges, for any values l, m , while it would not be the case with ML (see Fig. 4). Therefore, setting $w_* = 1/n$ guarantees that ECG alleviates the resolution issue on the ring of cliques example.

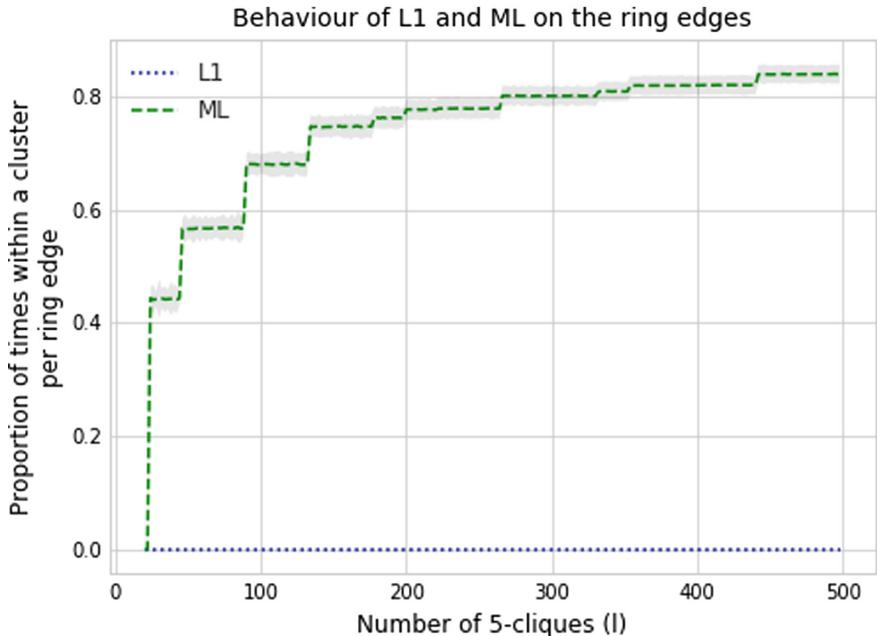


Fig. 4. We consider ring of cliques graphs with cliques of size 5. We vary the number of cliques from 20 to 500. With L1 as our weak learner, vertices on ring edges are never in the same cluster, and are thus assigned some small weight w_* in (1). If we use ML as our weak learner, those same vertices are often put in the same cluster.

Another advantage of ECG is its stability. This is illustrated in the first plot of Fig. 5, where we use the LFR graphs with $n = 482$ vertices from the previous section. For each graph, we ran both ECG and ML twice, and we compute the ARI score between the two partitions we obtained. We plot the mean ARI for the different values of the mixing parameter. The stability of ML starts to drop around $\mu = 0.3$, while ECG remains very stable until about $\mu = 0.6$.

4.2 Parameter Selection

There are two parameters in the ECG algorithm. However, the results are not too sensitive to their choice, and the default values are usually suitable. In the second plot of Fig. 5, we illustrate the impact of the ensemble size k over the LFR graphs of size $n = 482$, with $\mu = 0.63$. We see that as long as the ensemble is not too small (roughly $k \geq 8$), the results are comparable. Larger ensemble size does not hurt in terms of accuracy, but could have an impact on the running time. In our experiments, we found the default value $k = 16$ to be suitable. In the third plot of Fig. 5, we look at the impact of the minimum weight parameter w_* . Recall that a smaller value for w_* gives more importance to the ensemble. From that plot, we see that a small value for w_* is preferable, as we already

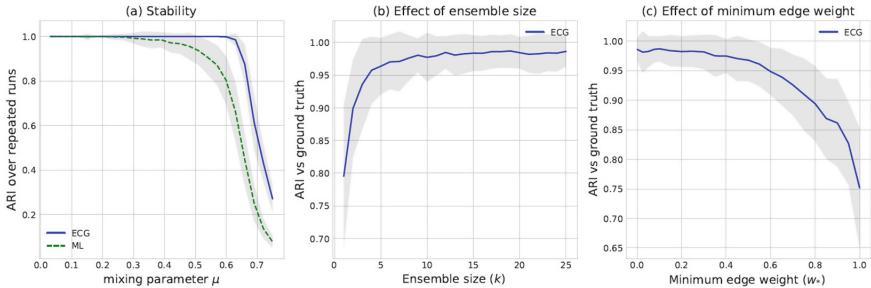


Fig. 5. In (a), we compare the stability of the ECG and ML algorithms by computing the ARI measure on two runs over the same graph. We consider LFR graphs of size $n = 482$ from the previous section. We see that ECG remains stable over a wider range of values for μ . In (b) and (c), we look at the effect of the two parameters in ECG: the ensemble size k and the minimum weight w_* . We consider the 100 LFR graphs of size $n = 482$ and $\mu = 0.63$. We see that the default parameters ($k = 16$, $w_* = 0.05$) are suitable, and that the algorithm is robust over the choice of parameters. The standard deviations are indicated by the shaded areas.

illustrated with the ring of cliques example. In our experiments, we found the default value $w_* = 0.05$ to be suitable.

4.3 Detecting Community Presence

With ECG, new edge weights are derived, which are used for the final ML run. Edges outside the 2-core are assigned the minimal weight w_* . The rationale for this exception is that we aim at finding communities supported by shared relations and in that sense, trees (free cycle graphs) have no such communities. The non-2-core subgraph is in fact a forest of trees each having a root in the 2-core. Without the above exception, these edges would get high weights due to the fact that there is very little diversity in their predicted communities.

The weights generated by ECG can also be used to quantify the strength of clusters in the graph. As an example, in Fig. 6, we plot the distribution of ECG weights for edges *within* clusters and edges *between* clusters. We consider both the ground-truth clusters, and the clusters found by ECG. This can be used as a diagnostic tool. For example, for the graphs with $\mu \leq .5$, the weights are very polarized (1 for edges within clusters, 0 for the other edges), which indicates a strong community structure. On the other hand, for large values of μ , the weight distribution are not as well separated, indicating a weak community structure.

5 Conclusion

We proposed ECG, a new graph clustering algorithm based on the fast multilevel Louvain algorithm, and on the concept of consensus clustering. We ran numerous experiments on artificial graphs, following a recently published study. All results

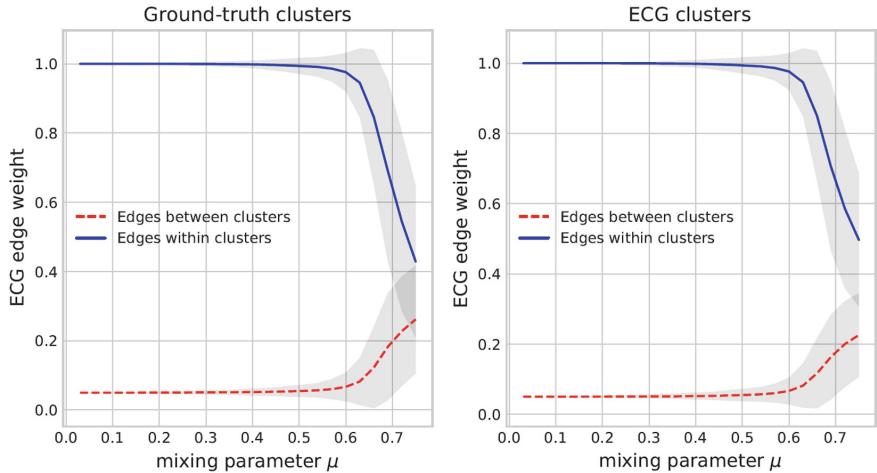


Fig. 6. We plot the distribution of edge weights respectively for edges within or between clusters. We consider LFR graphs of size $n = 482$ from the previous section. On the left plot, the ground-truth LFR clusters are used while on the right plot, we use the clusters returned by ECG.

obtained show an impressive accuracy and stability for ECG when compared to state-of-the-art algorithms. Moreover, ECG does not suffer from the resolution limit issue as much as other algorithms. It also provides weights which can be used to quantify the strength of the communities. Code is openly available [31].

References

1. Albatineh, A.N., Niewiadomska-Bugaj, M., Mihalko, D.: On similarity indices and correction for chance agreement. *J. Classif.* **23**(2), 301–313 (2006)
2. Berry, J., Hendrickson, B., LaViolette, R.A., Phillips, C.A.: Tolerating the community detection resolution limit with edge weighting. *Phys. Rev. E* **83** (2009)
3. Blondel, V.D., Guillaume, J.L., Lambiotte, R., Lefebvre, E.: Fast unfolding of communities in large networks. *J. Stat. Mech.* **08**(P10008) (2008)
4. Breiman, L.: Random forests. *J. Mach. Learn.* **45**(1), 5–32 (2001)
5. Clauset, A., Newman, M.E., Moore, C.: Finding community structure in very large networks. *Phys. Rev. E* **70**(6), 066,111 (2004)
6. Csardi, G., Nepusz, T.: The igraph software package for complex network research. *Int. J. Complex Syst.* (2006). <http://igraph.org>
7. Fortunato, S., Barthélémy, M.: Resolution limit in community detection. *Proc. Natl. Acad. Sci.* **104**(1), 36–41 (2007)
8. Fortunato, S., Hric, D.: Community detection in networks: a user guide. *Phys. Rep.* **659**, 1–44 (2016)
9. Fred, A.L., Jain, A.K.: Combining multiple clustering using evidence accumulation. *IEEE Trans. Pattern Anal. Mach. Intell.* **27**(6), 835–850 (2005)
10. Freund, Y., Schapire, R.E.: A decision-theoretic generalization of on-line learning and an application to boosting. *J. Comput. Sys. Sci.* **55**(119) (1997). <https://doi.org/10.1006/jcss.1997.1504>

11. Girvan, M., Newman, M.E.: Community structure in social and biological networks. *Proc. Natl. Acad. Sci.* **99**(12), 7821–7826 (2002)
12. Hubert, L., Arabie, P.: Comparing partitions. *J. Classif.* 193–218 (1985)
13. Lancichinetti, A., Fortunato, S.: Community detection algorithms: a comparative analysis. *Phys. Rev. E* **80**, 056,117 (2009)
14. Lancichinetti, A., Fortunato, S.: Consensus clustering in complex networks. *Nat. Sci. Rep.* **2**, 336 (2012)
15. Lancichinetti, A., Fortunato, S., Radicchi, F.: Benchmark graphs for testing community detection algorithms. *Phys. Rev. E* **78**(046110) (2008)
16. Li, T., Ogihara, M., Ma, S.: On combining multiple clusterings: an overview and a new perspective. *Appl. Intell.* **33**(2), 207–219 (2010). <https://doi.org/10.1007/s10489-009-0160-4>
17. Meilă, M.: Comparing clusterings—an information based distance. *J. Multiv. Anal.* **98**(5) (2007)
18. Newman, M.E.: The structure and function of complex networks. *SIAM Rev.* **45**, 167–256 (2003)
19. Newman, M.E.: Finding community structure in networks using the eigenvectors of matrices. *Phys. Rev. E* **74**(3), 036,104 (2006)
20. Orman, G.K., Labatut, V.: A comparison of community detection algorithms on artificial networks. In: International Conference on Discovery Science, pp. 242–256. Springer (2009)
21. Orman, G.K., Labatut, V., Cherifi, H.: Comparative evaluation of community detection algorithms: a topological approach. *J. Stat. Mech.* (2012). <https://doi.org/10.1088/1742-5468/2012/08/P08001>
22. Pedregosa, F., et al.: Scikit-learn: machine learning in python. *J. Mach. Learn. Res.* **12**, 2825–2830 (2011)
23. Pons, P., Latapy, M.: Computing communities in large networks using random walks. *Comput. Inf. Sci. ISCS* 284–293 (2005)
24. Poulin, V., Théberge, F.: Adjusted graph-aware rand index for comparing graph partitions (2018). <https://codeocean.com/2018/07/01/adjusted-graph-aware-rand-index-for-comparing-graph-partitions/code>
25. Poulin, V., Théberge, F.: Comparing graph clusterings: set partition measures vs. graph-aware measures (2018). [arXiv:1806.11494](https://arxiv.org/abs/1806.11494)
26. Raghavan, U.N., Albert, R., Kumara, S.: Near linear time algorithm to detect community structures in large-scale networks. *Phys. Rev. E* **76**(3), 036,106 (2007)
27. Reichardt, J., Bornholdt, S.: Statistical mechanics of community detection. *Phys. Rev. E* **74**(1), 016,110 (2006)
28. Romano, S., Vinh, N.X., Bailey, J., Verspoor, K.: Adjusting for chance clustering comparison measures. *J. Mach. Learn. Res.* **17**(1), 4635–4666 (2016)
29. Rosvall, M., Bergstrom, C.T.: Maps of random walks on complex networks reveal community structure. *PNAS* **105**(4), 1118–1123 (2007)
30. Strehl, A., Ghosh, J.: Cluster ensembles: a knowledge reuse framework for combining multiple partitions. *J. Mach. Learn. Res.* 583–617 (2002)
31. Théberge, F., Poulin, V.: Ensemble clustering for graphs (2018). <https://codeocean.com/2018/09/11/ensemble-clustering-for-graphs/code>
32. Topchy, A., Jain, A.K., Punch, W.: Clustering ensembles: models of consensus and weak partitions. *IEEE Trans. Pattern Anal. Mach. Intell.* **27**(12), 1866–1881 (2005)
33. Vega-Pons, S., Ruiz-Shulcloper, J.: A survey of clustering ensemble algorithms. *Int. J. Pattern Recognit. Artif. Intell.* **25**(3), 337–372 (2011)

34. Vinh, N.X., Epps, J., Bailey, J.: Information theoretic measures for clusterings comparison: is a correction for chance necessary? In: Proceedings of the 26th International Conference on Machine Learning (2009)
35. Vinh, N.X., Epps, J., Bailey, J.: Information theoretic measures for clusterings comparison: variants, properties, normalization and correction for chance. *J. Mach. Learn. Res.* **11**, 2837–2854 (2010)
36. Yang, Z., Algesheimer, R., Tessone, C.J.: A comparative analysis of community detection algorithms on artificial networks. *Nat. Sci. Rep.* **6**, 30,750 (2016)



A Community-Aware Approach for Identifying Node Anomalies in Complex Networks

Thomas J. Helling¹, Johannes C. Scholtes², and Frank W. Takes^{1,3(✉)}

¹ Department of Computer Science (LIACS), Leiden University, Leiden,
The Netherlands

t.j.helling@umail.leidenuniv.nl, takes@uva.nl

² Department of Data Science and Knowledge Engineering, Maastricht University,
Maastricht, The Netherlands

j.scholtes@maastrichtuniversity.nl

³ CORPNET, University of Amsterdam, Amsterdam, The Netherlands

Abstract. The overwhelming amount of network data that is nowadays available, leads to an increased demand for techniques that automatically identify anomalous nodes. Examples are network intruders in physical networks or spammers spreading unwanted advertisements in online social networks. Existing methods typically identify network anomalies from a local perspective, only considering metrics related to a node and connections in its direct neighborhood. However, such methods often miss anomalies as they overlook crucial distortions of the network structure that are only visible at the macro level. To solve these problems, in this paper, the CADA algorithm is proposed, which identifies irregular nodes from a global perspective. It does so by measuring the extent to which a node connects to many different communities while not obviously belonging to one community itself. Results on synthetic and real-world data show that the incorporation of the community aspect is of critical importance, as our algorithm significantly outperforms previously suggested techniques. In addition, it scales well to larger networks of hundreds of thousands of nodes and millions of links. Moreover, the proposed method is parameter-free, enabling the hassle-free identification of anomalies in a wide variety of application domains.

Keywords: Anomaly detection in networks · Node anomalies
LFR benchmark · Community detection

1 Introduction

An important problem in a large number of complex systems is identifying phenomena that diverge from what is considered to be normal. Doing this automatically based on data, is often referred to as anomaly detection [6]. Due to the rapidly growing amount of data that is nowadays available, methods to automatically identify data points that do not conform to expected behavior have seen

increased interest in a multitude of applications, such as fraud detection [25] and intrusion detection [7].

In network data, anomaly detection deals with identifying nodes that show anomalous behavior. Most network-based anomaly detection techniques do not label a node as anomalous, but rather assign an anomaly score to each node as an indicator of its deviance. Since the boundary of what should be considered anomalous can often not be determined in a straightforward manner, a domain expert is often involved in the final steps of the process of anomaly detection [13]. As a result, the focus of automated methods, such as the ones discussed in this paper, is to make a good initial selection of which nodes are likely anomalous.

Real-world networks obey many non-random statistical properties, such as heterogeneity of the degree distribution and the natural existence of communities; groups of nodes that are closely connected to each other [3]. This makes determining which nodes of the network are anomalies a far from trivial task. Although different types of network anomalies exist, we focus on the node anomaly. One example of the node anomaly is the node ego-network (i.e., a node and the connections between these nodes) with a star or near-star structure [1, 14].

Nodes with star structure are nodes that connect to many nodes that are not connected to each other. Such nodes are often referred to as nodes that are, roughly speaking, “not aware of the global structure” as they connect to seemingly random nodes in the network rather than closeby neighbors; they are community-agnostic. Examples of such anomalies are spammers in e-mail networks or advertisement calls in communication networks [14]. The majority of proposed anomaly detection methods approach the problem from a local perspective, not in the last place to keep methods scalable, as global methods typically take more computation time. For example, in [1] the OddBall algorithm was introduced, which proposes to fit power laws on a variety of statistics of the ego-network and then labels nodes deviating from these distributions as anomalous. However, as discussed above, connections made by anomalous nodes are typically not local, leaving a number of anomalous nodes undetected.

To solve this problem, we propose Community-Aware Detection of Anomalies (CADA), a global, community-aware and parameter-free algorithm to detect node anomalies in large real-world networks. We demonstrate with both synthetic and large-scale real-world complex network data that it effectively and efficiently identifies anomalies, outperforming previously proposed approaches.

The remainder of this paper is structured as follows. In Sect. 2 we provide necessary definitions and a formal problem statement. Next, in Sect. 3 we discuss previous work related to network-based anomaly detection. In Sect. 4 the proposed approach is outlined. Then in Sect. 5 the synthetic and real-world data sets are described. In Sect. 6 a set of experiments is performed to evaluate the approach. Finally, Sect. 7 concludes the paper.

2 Preliminaries and Problem Statement

This section provides various definitions of concepts and techniques, leading to the problem statement addressed in this paper.

2.1 Network Terminology

A network $G = (V, E)$ consists of a set of nodes V (also called vertices or objects) and a set of edges E (also called links or connections). Throughout this paper, the direction and weight of a link between two nodes are not taken into account, although the proposed approach can easily be extended to both directed and weighted networks. The total number of nodes $|V|$ and edges $|E|$ are referred to as n and m , respectively. The degree of a node is the number of edges connected to a node, and k is the average degree over all nodes in the network. The ego-network of a node is the subgraph consisting of the direct neighborhood of that node, including the node itself, and all edges between those nodes.

Over the years, a lot of properties of real-world networks have been discovered, e.g., a power law degree distribution, the small-world phenomenon [22] and a relatively high clustering coefficient [23]. Furthermore, it was found that in real-world systems groups of nodes tend to cluster together, forming so-called communities. The process of automatically identifying a division of a network into communities is called community detection [8]. For further details on this and other network science terminology, see [3].

2.2 Problem Statement

In network anomaly detection, the goal is to identify anomalous behavior in the network. We focus on the *node anomaly*, defined as a node that based on its connections seems unaware of the global network structure, recognized, e.g., by a star-like ego-network. Although relatively easily defined, the automated detection of these anomalies is far from trivial.

The aim of this paper is to devise an algorithm to compute a score $f(v)$ for each node $v \in V$ that indicates to what extent node v is a node anomaly. Based on a ranking induced by these scores, we can identify the most anomalous nodes. In Sect. 4 we will discuss approaches to tackle this problem.

3 Related Work

This section discusses related work on the topic of anomaly detection in networks. Typically, three types of anomalies in static networks can be distinguished: node anomalies, edge anomalies, and sub-graph anomalies [2, 4]. A variety of methods have been developed, ranging from approaches that incorporate structural network features to identify parts of the network that deviate from common distributions [1, 11], to proximity-based methods that measure the relevance of nodes by randomly walking through weighted and directed networks [10, 15, 17]. Other methods attempt to identify bridge nodes in networks by clustering nodes together by either using Personalized PageRank [20], or by capturing the underlying correlations of the network by decomposing the adjacency matrix into a low-rank approximation of which the residual matrix that can be used to indicate parts of the networks that do not correspond to the expected patterns [21].

Henceforth we focus on node anomalies, nodes that are not aware of the global network structure such as intruders, spammers, or telecommunication advertisement bureaus [1, 14]. The OddBall algorithm was one of the first to detect the node anomaly by (1) extracting the number of nodes and edges from the ego-network of each node, (2) identifying a pattern of normal behaviour by defining so-called power laws on these features, and (3) identifying points that deviate from these power laws. It was found that ego-networks with an equal number of nodes and edges is a network with a star-structure, were found to deviate from the derived power law [1]. Others attempted to improve the identification of nodes with star or clique structure by introducing alternate local features of the ego-network [9, 14], incorporating features from the extended neighborhoods into the model [9, 12], or by identifying so-called hubs, nodes that do not share a certain fraction of common neighbors with any of their adjacent nodes [24]. It should be noted that the node anomaly is considerably different from merely selecting high betweenness centrality nodes in the network, because this centrality measure does not differentiate between edges in or between different parts of the network, while node anomalies typically do [12].

In this work, we aim to move away from previously approached locally operating algorithms, and take the global structure of the network into account.

4 Approach

First two approaches that already focus on identifying the node anomaly are discussed in Sect. 4.1. Then, the proposed community-aware approach CADA is introduced in Sect. 4.2. Recall, that each method assigns an anomaly score to all nodes in the network, where a higher score indicates a higher divergence from expected behavior.

4.1 Existing Approaches

OddBall [1] extracts the number of nodes N_i and edges E_i of the ego-network of each node $i \in V$, to exhibit normal patterns in the data set by fitting the Egonet Density Power Law in the form of $E_i \propto N_i^\alpha$. Ego-networks that deviate the most from the fitted power law are considered to be an anomaly. The OddBall anomaly score $ob(i)$ of a node i is computed as follows:

$$ob(i) = \frac{Cx_i^\theta}{y_i} \log(y_i - Cx_i^\theta + 1)$$

Here, Cx_i^θ is the expected number of edges with x_i number of nodes, while y_i is the true number of edges. Computing this metric can be done in $O(n \cdot k^2)$ time, checking for each of the n nodes the k^2 neighbor pairs in its ego network, where k is the node degree. Although this method has the advantage of taking the natural existence of power laws in real-world network data into account, one obvious shortcoming of this method is that solely the direct neighborhood of a node is taken into consideration.

Embed [12] is a network embedding approach that preserves the local linkage structure of nodes by assigning each node i to a r -dimensional vector X_i , where X_i^r describes the relationship between node i and region r under the minimization of the objective function O , which is defined as follows.

$$O = \sum_{(i,j) \in E} \|X_i - X_j\|^2 + \alpha \sum_{(i,j) \notin E} (1 - \|X_i - X_j\|)^2$$

Here, α is a balancing factor between existent edges and non-existent edges. To minimize the computational cost of the algorithm, the authors propose to (1) approximately represent O by removing α and equal the number of existent and non-existent edges to balance the optimization, (2) to initialize the vectors with equal embedding values if nodes belong to the same partition according to network partitioning method METIS, and (3) reduce the number of dimensions in vectors of the embedding to $k + \beta$, where k is the average degree and β a toleration factor for the number of regions node anomalies connect to. According to the authors, $\beta = k/4$ is sufficient. After successful minimization of the network embedding, we represent the correlation of node i with r regions as follows:

$$NB(i) = y_i^1, \dots, y_i^r = \sum_{(i,j) \in E} (1 - \|X_i - X_j\|) \cdot X_j$$

The Embed anomaly score $em(i)$ of a node i can then be computed as follows.

$$em(i) = \sum_{j=1}^r \frac{y_i^j}{y_i^*}$$

Here, $y_i^* = \max(y_i^1, \dots, y_i^r)$. Embed runs in $O(t \cdot m \cdot (k + \beta))$, where t is the iteration threshold of gradient descent, again with m the number of edges and k the average degree. A t of 50 is sufficient. However, a drawback of the Embed algorithm is that the results are dependent on the number of dimensions r , that are chosen, and as such the approach is not parameter-free.

4.2 Proposed Approach: CADA

The methods, discussed in the previous subsection, mainly focus on identifying the node anomalies from a local perspective. Here, we use community detection to also take the global perspective into account. The proposed **Community-Aware Detection of Anomalies** algorithm consists of two steps.

First, CADA assigns each node to a particular community using an out-of-the-box community detection method [8]. In this paper, we employ two well-known community detection algorithms that both scale linearly in the number of edges and as such run in $O(m)$: the Louvain algorithm and the Infomap approach. Louvain [5] is a greedy optimization method that optimizes the so-called modularity function Q , a measure for the quality of a division of a network into communities. It is based on the relation between the number of edges that connect nodes in the same community and the expected value for the same size of

the network if the same number of edges are randomly distributed. Infomap [18, 19] is an information-theoretic approach that utilizes random walks to compress the information that is needed to minimally describe how information randomly flows through the network. We refer to CADA_L or cd_L when Louvain is used as community detection method, and to CADA_I or cd_I when Infomap is used as community detection method. Both assign each node to a community, and can handle undirected and directed networks (Louvain would ignore link direction), and can incorporate weights.

The second step of CADA is to assign an anomaly score to each node, based on the communities each node connects to. The anomaly score describes to what extent the neighbors of a node belong to a diverse number of communities, while the node itself does not strongly belong to one of them. Thus, for each node i , we create a vector g_i , where g_i^c represents the number of neighboring nodes that belong to community c . g_i^* represents the maximum number of neighboring nodes that belong to the same community. We can then compute an anomaly score for each node as follows:

$$cd(i) = \sum_{j=1}^c \frac{g_i^j}{g_i^*}$$

5 Data

A challenge in anomaly detection in networks is that there do not exist many publicly available labeled data sets with ground truth anomalies. Therefore, we illustrate the results of our anomaly detection methods on both real-world network data sets (see Sect. 5.1) and synthetic network data sets (as discussed in Sect. 5.2). Finally, the anomaly types are discussed in Sect. 5.3.

5.1 Real-World Network Data Sets

Three different real-world data sets (see Table 1) are chosen to qualitatively assess the performance of the considered anomaly detection algorithms. *Douban* (<http://socialcomputing.asu.edu/datasets/Douban>) is a Chinese recommendation website where a link exists between two users if they had an explicit friendship connection. *Amazon* (<http://snap.stanford.edu/data/amazon0601.html>) is a co-purchasing network where a link exists between two articles if these products are frequently purchased together on Amazon. *DBLP* (<http://projects.csail.mit.edu>).

Table 1. Properties of the real-world network data sets.

Data set	Description	Number of nodes	Number of edges
<i>Douban</i>	Social network	154,907	327,162
<i>Amazon</i>	Co-purchase network	403,394	2,443,408
<i>DBLP</i>	Co-authorship network	1,412,414	5,947,085

mit.edu/dnd/DBLP) is a collaboration network based on co-authorship between mostly computer scientists, extracted from the popular DBLP listing website.

5.2 Synthetic Network Data Sets

Synthetic networks are generated using the Lancichinetti-Fortunato-Radicchi (LFR) benchmark, which allows one to generate networks of different sizes [16]. The LFR benchmark is chosen because it creates networks adhering to real-world network properties, such as degree heterogeneity and community size heterogeneity. They are generated as follows:

1. A network of n nodes is generated where each node has a degree based on power law exponent γ_1 with k_{\min} and k_{\max} so that the average degree is k .
2. Community sizes are generated from the power law minus exponent γ_2 with community sizes s_{\min} and s_{\max} so that $s_{\min} > k_{\min}$ and $s_{\max} > k_{\max}$, and the sum of community sizes is n . If these are not chosen, the community sizes will be chosen close to the degree extremes.
3. Each node is assigned to a community, as long as the community does not exceed the set community size. If the number of edges within the community exceeds the community size, the node becomes homeless. Homeless nodes are randomly assigned to a community. If the community size is exceeded then a randomly selected node of that community becomes homeless. The process terminates if all communities are completed.
4. The algorithm rewrites the edges so that each node has a fraction of approximately μ internal neighbors and $1 - \mu$ external neighbors.

We generate networks with a size ranging from 1,000 to 500,000 nodes, and mixing parameters between 0.1 and 0.6. To provide a fair performance comparison in the experiments, the parameters are chosen as in [12], as shown in Table 2.

5.3 Anomaly Types

Following the approach suggested in [12] we employ two generative processes to insert anomalies in the synthetic networks, that reflect on the node anomaly defined in Sect. 2.2.

Table 2. Parameter settings for generating LFR benchmark networks.

Parameter	Description	Setting
n	The number of nodes	From 10^3 to $5 \cdot 10^5$
k	Average degree	$2 \cdot n^{1.15}/n$
k_{\max}	Maximum degree	$n^{1/(t_1-1)}$
γ_1	Minus exponent for degree distribution	3
γ_2	Minus exponent for community size distribution	2
μ_t	Mixing parameter for topology	From 0.1 to 0.6

Random anomaly is inspired by the fact that infiltrating nodes are not aware of the global network structure and therefore connect to random nodes in the network. The anomalies are injected by adding $n/100$ nodes that connect to x random existing nodes, where x is between k and k_{max} . For each inserted node, the value of x is set by drawing a value from the the same power law degree distribution as that of the synthetic network.

Replaced anomaly first generates $n + a$ nodes with the LFR benchmark. The goal is to replace a nodes to obtain $n + n/100$ nodes. We randomly select x existing nodes in the network that have a degree lower than $2 \cdot k$. An anomaly is injected by rewiring all edges from the x nodes to the new anomaly. The x nodes are then removed from the network. x ranges from 2–21, with an increment of 1, until $n + n/100$ nodes are obtained.

6 Experiments

In this section, we describe how and what experiments were executed to measure the performance of CADA.

6.1 Experimental Setup

For the experiments, the parameters are set as follows. For Embed we set the number of dimensions to \sqrt{n} , to let the networks scale properly with the network size. Note that we flagged nodes anomalous if the anomaly score of the nodes exceeded a certain threshold Θ . To fairly evaluate and compare the algorithms, we obtained the k nodes that exceeded threshold Θ for CADA_I. Then, we extracted the top- k most anomalous nodes according to the other methods to fairly compare the most anomalous nodes. For the synthetic data sets, Θ was set to 4 to maximize performance with CADA_I. For the real-world data sets, Θ was set on 5, 5, and 9 for *Amazon*, *DBLP*, and *Douban*, respectively. Θ was chosen so that we at least covered $n/100$ nodes in the network. All reported are averages obtained from executing the experiments 10 times.

The experiments were run on a 2.8 GHz Intel Core i7 CPU with 16 GB RAM, using a combination of Python and C++ algorithms. Python package *NetworkX* was used for network operations. CADA was implemented in Python and is available at <https://github.com/thomashelling/cada>.

6.2 Evaluation Metrics

As quantitative evaluation of the algorithms on synthetic network data sets, we use the F_1 -score, which is based on recall and precision. Recall is the fraction of true positives (discovered anomalies) divided by the total number of ground-truth anomalies, while precision is equal to the true positives divided by the number of nodes that are flagged anomalous. The F_1 -score is then:

$$F_1 = 2 \cdot \frac{precision \times recall}{precision + recall}$$

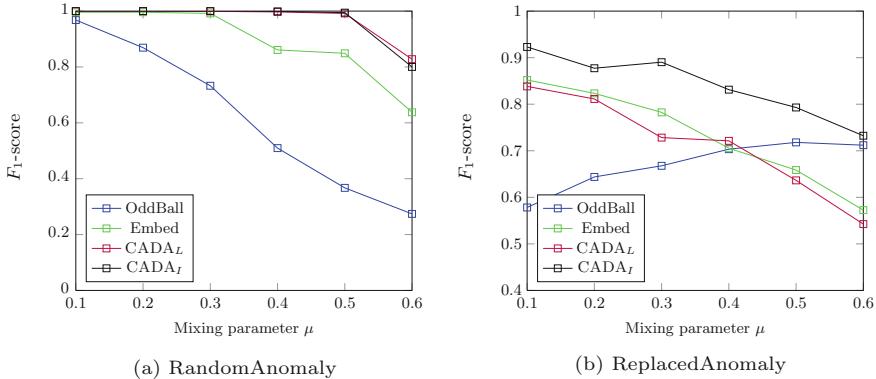


Fig. 1. F_1 -score for different mixing parameter values for networks with 100,000 nodes.

To qualitatively compare between the different anomaly detection algorithms, we propose to measure the similarity in the 1% most anomalous nodes for each anomaly detection algorithm on the three real-world data sets. Note that this could be slightly more than 1% because nodes could have the same anomaly score, in which case we included all nodes with the same score as the node at the exact cutoff. For two sets of discovered node anomaly sets A_1 and A_2 , we use the Jaccard similarity, defined as follows:

$$J(A_1, A_2) = \frac{|A_1 \cap A_2|}{|A_1 \cup A_2|}$$

6.3 Results on Synthetic Data

Figure 1 shows the F_1 -score of each of the methods for different values of the mixing parameter. For both anomaly types, the result is significantly higher in case of our CADA method. Embed outperforms OddBall, while OddBall is closing in on CADA and Embed once the community structure diminishes for ReplacedAnomaly. In general, results when using Infomap were higher than for Louvain, which is likely due to the resolution limit, which starts to play a larger role for higher values of the mixing parameter.

Figure 2 shows the F_1 -score for a fixed value of 0.4 for the mixing parameter, but for varying sizes of the network. As the figures show, for RandomAnomaly, even for smaller size networks, CADA performs best, with Embed closing in, but only for networks larger than 200,000 nodes. Although the difference is modest, for ReplacedAnomaly, CADA with Infomap consistently performs best.

6.4 Results on Real-World Data

Although on real-world data we cannot report F_1 -scores, we can look at the agreement between the different methods, of which results in the form of Jaccard similarity are reported in Table 3.

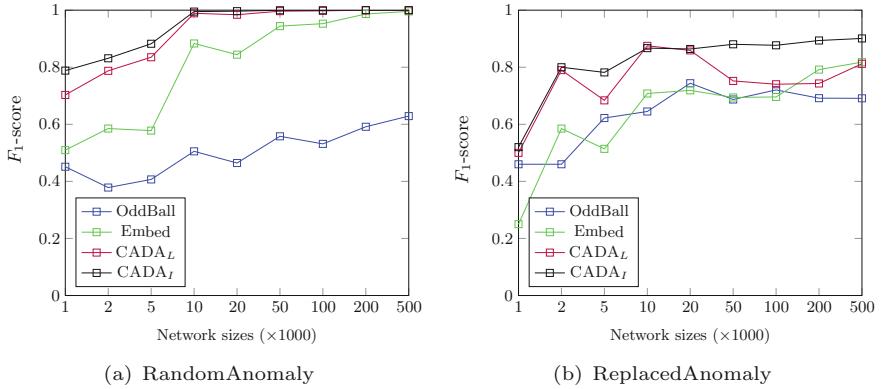


Fig. 2. F_1 -score for different network sizes with fixed mixing parameter 0.4.

Zooming in on the *DBLP* dataset and the outliers found, two noteworthy findings were obtained. Two anomalies were solely identified by CADA with Infomap, but ignored by OddBall and Embed. First, we found authors that have published with many different authors, such as prof. dr. H. Vincent Poor, who was president of the IEEE Information Theory Society, and at the time of the data set has published with over 400 authors from all over the world. Second, many discovered anomalies were actually authors with the same name, such as 63 distinct authors named ‘Wei Liu’ that collaboratively published with over 1332 different authors. Other such authors were ‘Jing Li’, ‘Yan Zhang’, and ‘Yu Zhang’. This illustrates that apart from outliers such as authors with extremely large number of publications, also errors in the underlying data can efficiently be identified using anomaly detection techniques.

6.5 Discussion

The experiments demonstrate that CADA is suitable for the purpose of identifying anomalous nodes, and appears to do so effectively on both synthetic and real-world data. Evaluation on synthetic networks with varying mixing parameters shows that the method performs consistently better compared to other methods to identify node anomalies. It reveals that methods that solely observe

Table 3. Jaccard similarity of the most anomalous nodes on *DBLP*, *Amazon*, and *Douban* (left to right). OddBall (ob), Embed (em), CADA Louvain (cd_L), and CADA Infomap (cd_I).

	em	cd_L	cd_I		em	cd_L	cd_I		em	cd_L	cd_I		
ob	0.02	0.02	0.02		ob	0.11	0.11	0.17		ob	0.00	0.00	0.00
em	-	0.22	0.24		em	-	0.17	0.20		em	-	0.37	0.43
cd_L	-	-	0.24		cd_L	-	-	0.21		cd_L	-	-	0.34

the anomalies from a local perspective may possibly oversee the global community structure and therefore miss anomalous nodes. Moreover, the community detection methods chosen scale linearly with the number of edges in the network, and therefore provide an efficient method to identify node anomalies [8]. One limitation of CADA, is that it is dependent on the community detection performance of the network, and unfortunately there is no universal method to detect communities most accurately in each network. This is illustrated in Fig. 1(a) and (b), where $CADA_I$ consistently outperforms $CADA_L$. Moreover, Table 3 shows that the overlap between anomalous nodes for $CADA_I$ and $CADA_L$ in real-world networks still varies a lot, demonstrating that CADA relies on the performance of community detection methods to accurately flag nodes as most anomalous. However, Embed outperforms OddBall, but is dependent on the number of dimensions to accurately detect anomalies. The community detection methods can optionally be parameter free, while performance could be enhanced by uncovering smaller or bigger communities, for example using the resolution parameter in case of the Louvain algorithm.

7 Conclusions and Future Work

Previously proposed techniques approached the network anomaly detection problem from a local perspective. In this paper we proposed the CADA algorithm, which identifies anomalous nodes based on whether they connect to large number of communities, while not belonging to one distinct community themselves. An advantage of this community-aware approach is that it scales linearly with the number of edges, improving upon previous techniques. Furthermore, it is parameter free and highly effective. Experiments showed that our proposed community-aware methodology can spot anomalies in both synthetic and real-world data sets that were not discovered by local methods. Furthermore, on all synthetic benchmark datasets, CADA outperformed previous approaches.

In future work we want to investigate which community detection methods are most robust for the purpose of node anomaly detection, and whether a hybrid method combining both global and local features, may yield more accurate and relevant anomalies.

References

1. Akoglu, L., Mcglohon, M., Faloutsos, C.: Anomaly detection in large graphs. In: CMU-CS-09-173 Technical Report (2009)
2. Akoglu, L., Tong, H., Koutra, D.: Graph based anomaly detection and description: a survey. Data Min. Knowl. Discov. **29**(3), 626–688 (2015)
3. Barabási, A.L.: Network Science. Cambridge University Press (2016)
4. Bindu, P.V., Thilagam, P.S.: Mining social networks for anomalies: methods and challenges. J. Netw. Comput. Appl. **68**, 213–229 (2016)
5. Blondel, V.D., Guillaume, J.L., Lambiotte, R., Lefebvre, E.: Fast unfolding of communities in large networks. J. Stat. Mech. Theory Exp. **10008**(10), 6 (2008)

6. Chandola, V., Banerjee, A., Kumar, V.: Anomaly detection: a survey. *ACM Comput. Surv.* **41**(September), 1–58 (2009)
7. Denning, D.E.: An intrusion-detection model. In: *Proceedings of IEEE Symposium on Security and Privacy*, pp. 118–131 (2012)
8. Fortunato, S.: Community detection in graphs (2010)
9. Hassanzadeh, R., Nayak, R., Stebila, D.: Analyzing the effectiveness of graph metrics for anomaly detection in online social networks. In: *Lecture Notes in Computer Science*, vol. 7651, pp. 624–630 (2012)
10. Haveliwala, T.H.: Topic-sensitive PageRank. In: *Proceedings of 11th International Conference on World Wide Web*, pp. 517–526 (2002)
11. Henderson, K., et al.: It's who you know: graph mining using recursive structural features. In: *Proceedings of 17th ACM International Conference on Knowledge Discovery and Data Mining*, p. 663 (2011)
12. Hu, R., Aggarwal, C.C., Ma, S., Huai, J.: An embedding approach to anomaly detection. In: *Proceedings of 32nd IEEE International Conference on Data Engineering*, pp. 385–396 (2016)
13. Janssens, J.: Outlier Selection and One-Class Classification. Maastricht University (2013)
14. Kaur, R., Singh, S.: A comparative analysis of structural graph metrics to identify anomalies in online social networks. *Comput. Electr. Eng.* **57**, 294–310 (2017)
15. Krishnan, V., Raj, R.: Web spam detection with anti-trust rank. *AIRWeb* **6**, 37–40 (2006)
16. Lancichinetti, A., Fortunato, S., Radicchi, F.: Benchmark graphs for testing community detection algorithms. *Phys. Rev. E* **78**(4) (2008)
17. Page, L., Brin, S., Motwani, R., Winograd, T.: The PageRank Citation Ranking: Bringing Order to the Web. *World Wide Web Internet Web Inf. Syst.* **54**(1999–66), 1–17 (1998)
18. Rosvall, M., Bergstrom, C.: Maps of random walks on complex networks reveal community structure. *Proc. Natl. Acad. Sci.* **105**(4), 1118–1123 (2008)
19. Rosvall, M., Bergstrom, C.T.: Multilevel compression of random walks on networks reveals hierarchical organization in large integrated systems. *PLoS ONE* **6**(4) (2011)
20. Sun, J., Qu, H., Chakr, D., Faloutsos, C.: Neighborhood formation and anomaly detection in bipartite graphs. In: *Proceedings of SIAM Conference on Data Mining*, pp. 1–8 (2008)
21. Tong, H., Lin, C.: Non-negative residual matrix factorization with application to graph anomaly detection. In: *Proceedings of SIAM Conference on Data Mining*, pp. 143–153 (2011)
22. Travers, J., Milgram, S.: An experimental study of the small world problem. *Sociometry* **32**(4), 425 (1969)
23. Watts, D.J., Strogatz, S.H.: Collective dynamics of ‘small-world’ networks. *Nature* **393**(6684), 440–442 (1998)
24. Xu, X., Yuruk, N., Feng, Z., Schweiger, T.A.J.: SCAN: a structural clustering algorithm for networks. In: *Proceedings of 13th ACM International Conference on Knowledge Discovery and Data Mining*, pp. 824–833 (2007)
25. Kou, Y., Lu, C.-T., Sirwongwattana, S., Huang, Y.-P.: Survey of fraud detection techniques. In: *Proceedings of IEEE International Conference on Networking, Sensing and Control*, 2004, pp. 749–754 (2004)



Is Community Detection Fully Unsupervised? The Case of Weighted Graphs

Victor Connes¹, Nicolas Dugué¹⁽⁾, and Adrien Guille²

¹ Le Mans Université, LIUM, EA 4023, Laboratoire d’Informatique de l’Université du Mans, Le Mans, France

{victor.connes, nicolas.dugue}@univ-lemans.fr

² University of Lyon, ERIC, 5, avenue Pierre Mendès France, 69676 Bron Cedex, France

Abstract. In the field of NLP, word embeddings have recently attracted a lot of attention. A textual corpus is represented as a sparse words co-occurrences matrix. Then, the matrix can be factorized, for example using SVD, which allows to obtain a shorter matrix with dense and continuous vectors. To help SVD, PMI measure is applied on the initial co-occurrence matrix, assigning a relevant weight to the co-occurrences by normalizing them using both the considered words frequencies. In this paper, we follow this idea to study if weighted networks can benefit from pre-processing that can help community detection. We first design a benchmark using LFR networks. Then, we consider PMI and another NLP inspired measure as a preprocessing of the links weights, and show that PMI worsens the results while the other one improves them. By distinguishing links inside communities and links between communities into two classes, we show that this is due to the weights distributions of these links. Links between communities are in average bigger, leading to bigger values of PMI. From this analysis, we design another set of experiments that show that it is possible to classify efficiently links into these two classes, using a small set of features. Finally, we introduce the Supervised Label Propagation (SLP) algorithm that takes into account the classification results during the propagation. This algorithm clearly improves the results, leading us to a major questioning: is community detection on weighted networks a fully unsupervised task? We conclude with our thoughts on this topic.

1 Introduction

Recently, a lot of progress has been done in the field of Natural Language Processing. This progress is mostly due to the recent approaches based on distributional semantics that allow to learn vectors representing words meaning efficiently [11]. Among these approaches, one of them is particularly interesting in our context. This approach discussed by Levy [9] captures semantic information by using the term-term co-occurrence matrix extracted from a corpora of interest.

Starting from this matrix, PPMI vectors are built. These vectors encode the co-occurrences normalized by the terms frequencies. Finally, from these big sparse vectors, short dense vectors are extracted using Singular value decomposition. Skipping the step where PPMI vectors are built, and extracting short dense vectors from the basic term-term co-occurrence matrix leads to poorer results. We can thus state that pre-processing the raw data with PPMI makes the learning process performing better.

In this paper, we aim to explore if the same analysis can be transposed to complex networks, in particular to the problem of community detection in weighted graphs. Sarkar and Dong [15] have shown the pertinence of using SVD in the context of community detection. Since that, one can postulate than pre-processing of a term-term co-occurrence matrix which improves SVD results could also improve community detection results. Namely, we wonder whether revising edge weights may improve the results of community detection, as applying PPMI improve SVD results. Indeed, this problem is quite similar with the one of NLP presented: term-term co-occurrences matrices can be represented as weighted graphs. As a starting point for the study, we therefore consider applying measures such as the PPMI to normalize weights and observe the results. In our experiments, we consider the LFR benchmark [8] to design artificial but realistic graphs. As far as we know, there are no publicly available weighted graphs with groundtruth communities, which motivates our choice of using LFR graphs. To our surprise, while PPMI worsens the results, another normalization inspired from Pennington et al. [13] improves them. Some experiments show that links that connect nodes of different communities are in average bigger, leading to bigger values of PMI. These links are thus considered more important by the community detection algorithms leading to poorer results. Nodes connected by these links seem to be also of higher degrees. This means that links connecting different communities have specific features. We show that it is possible to design quite efficient classifiers using these features and a few more based on the nodes local connectivity. Results of these classifiers are finally used to design a new community detection algorithm based on label propagation: Supervised Label Propagation (SLP). Experiments using SLP show how it outperforms classic algorithms such as Label propagation [14] or Louvain [2].

The paper is organized as follows. In Sect. 2, we introduce the community detection problem and the LFR benchmarks. Sect. 3 is dedicated to the NLP based normalization of weights and their impact on community detection results. In Sect. 4, we exploit the results from Sect. 3 to train efficient classifiers using straightforward features. Section 5 describes the SLP algorithm and how it performs on LFR. Finally, we conclude and discuss the results of this work, raising the following question: Is community detection fully unsupervised or may it benefit from supervision?

2 Community Detection

Let $G = (V, E, W)$ a weighted graph, with V the set of vertices, E the set of edges, and W the set of weights assigned to the edges. We note $N(u) = \{v \in V : uv \in E\}$ the *neighborhood* of node u , i.e. the set of nodes connected to u in G . The degree of a node u is noted $d(u) = |N(u)|$. The weighted degree of the node, also called *strength* s , is calculated as the sum of the weighted edges between u and its neighbors: $s(u) = \sum_{v \in N(u)} w_{uv}$ with w_{uv} the weights assigned to the uv edges.

Weighted graphs modeling real systems were extensively studied by the complex networks community. Barthélémy et al. [1] showed the relevance of considering weights on edges, weights bringing more information. Newman [12] brought the concept of community structure from unweighted to weighted graphs. In particular, Newman shows that *modularity* defined for unweighted graphs can be easily transposed to weighted graphs. Modularity (Definition 2) is a measure of the quality of a network partitioning, considering the difference between the edges that connect nodes belonging to the same communities and the expected value of the same quantity if edges are assigned at random.

Definition 1 (Modularity for weighted graphs).

$$\frac{1}{2m} \cdot \sum_{u,v} \left[w_{uv} - \frac{s(u) \cdot s(v)}{2m} \right] \cdot \delta(c_u, c_v) \quad (1)$$

with $m = \frac{1}{2} \cdot \sum_{u,v} w_{uv}$ and $\delta(c_u, c_v)$ is 1 if u and v belongs to the same community, i.e. $c_u = c_v$.

The modeling of such kind of graphs is a major issue to the community. Lancichinetti and Fortunato introduced the LFR benchmarks [8] that allows to build realistic artificial weighted graphs and to vary some important parameters related to the graphs topology. It is thus possible to vary the network size ($|V| = n$), the degree and strength distribution, the communities sizes range, the maximal degree of the nodes, and also the clarity of the communities, namely the mixing parameter μ . In the rest of this paper, we use a set of graphs built with the LFR¹. We use a set of parameters similar to the one used in [8]. The number of nodes is set to 5000, we vary the average degree that ranges in [15, 20, 25], the mixing parameter that ranges in [0.2, 0.3, 0.4] and the maximum degree that ranges in [100, 300, 500]. Other parameters are fixed to their usual values. We do not aim here to be exhaustive, but to show the relevance of our experiments in a general framework. For each set of parameters, we generate 50 graphs.

In the rest of this paper, we use Blondel et al. [2] algorithm, also called Louvain to compute the communities. We also use Label propagation of Raghavan et al. [14]. To evaluate the partitions returned by these algorithms, we compare them to the groundtruths provided by LFR. To that aim, we use the Adjusted Rand Index (ARI) [7] and the Normalized Mutual Information (NMI) [16]:

¹ Experiments and graphs can be found at <https://github.com/nicolasdugue/WeightedCommunityDetection/>

Definition 2 (Normalized Mutual Information).

$$NMI(\mathcal{U}, \mathcal{V}) = \frac{I(\mathcal{U}, \mathcal{V})}{\sqrt{H(\mathcal{U}) \cdot H(\mathcal{V})}} \quad (2)$$

with I the mutual information, and H the conditional entropy function.

In the following section, we demonstrate that on these graphs, PMI-inspired measures worsen the results of community detection, and that the Glove-inspired one improves them.

3 Normalizing Weights

The Pointwise Mutual Information (PMI) is defined as follows:

Definition 3 (PMI and PPMI).

$$PMI(x, y) = \log\left(\frac{p(x, y)}{p(x)p(y)}\right) \quad (3)$$

$$PPMI(x, y) = \max(PMI(x, y), 0) \quad (4)$$

In the NLP case, x and y are terms, and the PMI evaluates whether x and y occur together more than expected regarding their corpus frequency. In our case, PMI is used to evaluate whether two vertices are more or less connected than expected regarding their degree. Positive Pointwise Mutual Information (PPMI) is defined as the minimum between PMI and 0. The function used in Glove [13] (Definition 4) allows to avoid over-weighting of frequent co-occurrences;

Table 1. Community detection results on the whole set of LFR graphs. PLP stands for Label Propagation, and “/” means that no weight pre-processing was applied.

Rank		Normalization	Algorithm	Average		Std	
ARI	NMI			ARI	NMI	ARI	NMI
1	1	Glove $x_{max} = 50 \alpha = 0.2$	PLP	0.974	0.979	0.048	0.040
2	2	Glove $x_{max} = 30 \alpha = 0.3$	PLP	0.967	0.973	0.058	0.046
3	5	/	PLP	0.913	0.935	0.103	0.076
4	3	Glove $x_{max} = 50 \alpha = 0.2$	Louvain	0.877	0.947	0.112	0.046
5	4	Glove $x_{max} = 30 \alpha = 0.3$	Louvain	0.863	0.940	0.118	0.048
6	6	PPMI	PLP	0.819	0.906	0.254	0.137
7	7	PPMI	Louvain	0.796	0.891	0.149	0.109
8	8	/	Louvain	0.778	0.889	0.147	0.079
9	9	PMI	PLP	0.736	0.843	0.276	0.158
10	10	PMI	Louvain	0.633	0.798	0.207	0.163

Definition 4 (Glove function).

$$f(x) = \begin{cases} (x/x_{max})^\alpha & \text{if } x < x_{max} \\ 1 & \text{otherwise.} \end{cases} \quad (5)$$

with x_{max} and α parameters. In Pennington et al., x_{max} is 100 and $\alpha = 3/4$.

In our experiments, we apply PMI, PPMI and the so-called Glove function to the weights of the LFR graphs generated as described Sect. 2. We then apply standard Louvain algorithm and label propagation to detect communities [2]. As one can see Table 1, PMI and PPMI worsens the results. NMI and ARI are lower when these functions are applied. However, Glove normalization allows both Louvain and Label Propagation algorithm to perform better. It seems that penalizing the largest co-occurrences helps the algorithm. About the Glove normalization, we have observed that these results remain valid for numerous variations of the parameters. Nevertheless, we do not aim to be exhaustive and thus present a few examples of the tested variations.

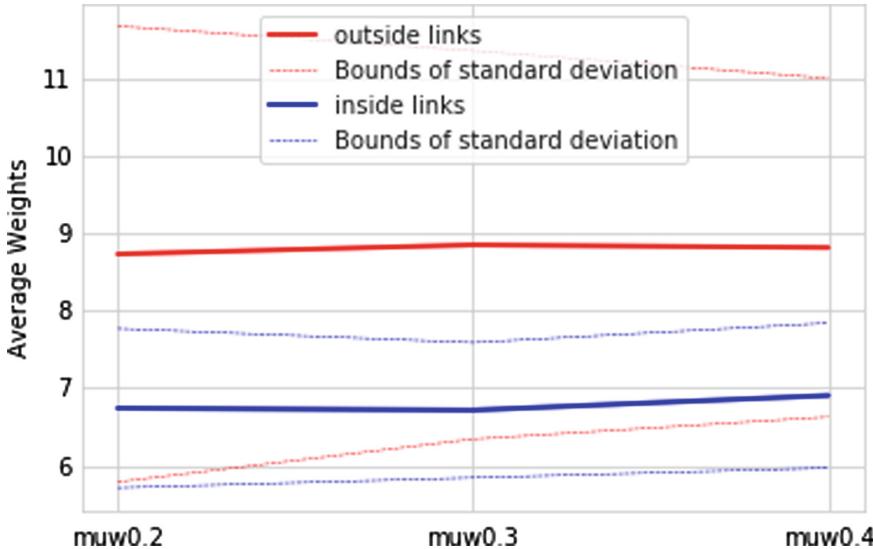


Fig. 1. Average weights of links according to their class (outside or inside) and to the mixing parameter μ .

Let call the edges that connect nodes belonging to different communities (resp. to the same community) **outside** (resp. **inside**) links. It is interesting to notice on Fig. 1 that in average, outside links weights are higher than the inside links weights, which may explain the results previously detailed. Indeed, Glove normalization reduces the weights of the outside links, which leads to better results.

This analysis leads us to train classifiers to distinguish outside links from the inside ones. The performance of these classifiers and the basic features we use to train them are discussed in the next section.

4 Classifying Links: Inside or Outside Communities?

In this section, we train and evaluate classifiers on the task of *outside* links detection. The idea is that detecting the links that connect distinct communities in a supervised way may then be relevant to help the unsupervised algorithms to proceed. Our classifiers are based on XGBoost [4], a gradient tree boosting system that performs well. It is indeed used in most of the data science challenges such as the ones provided by Kaggle. Other classic classifiers such as logistic regression or SVM were also tried. However, their results are poorer than the ones obtained with XGBoost in all cases. Because our goal is not to be exhaustive, but to raise questions about the relevance of supervision in this context, we only show the results obtained with XGBoost. Furthermore, for the same reason, we choose a small set of straightforward features to describe links, as one can see Fig. 2. These features are inspired from the results of the previous section. They are based on local information and easy to compute.

- Minimum strength of nodes at the ends of the edge ;
- Maximum strength of nodes at the ends of the edge ;
- Minimum clustering coefficient of nodes at the ends of the edge ;
- Maximum clustering coefficient of nodes at the ends of the edge ;
- Edge weight.

Fig. 2. List of features used by XGBoost to separate *outside* links from the others.

Classifiers performance are evaluated with classic measures from the machine learning field, such as precision (Eq. 6), recall (Eq. 7) and F-measure (f1 on the Figures) which is the harmonic mean of the two previous measures:

$$Precision = \frac{TP}{TP + FP} \quad (6)$$

$$Recall = \frac{TP}{TP + FN} \quad (7)$$

with TP that stands for true positive, FP for false positives and FN for false negatives.

We split our set of graphs generated with the LFR into two sets: the train set and the test set. The train set is made of 10 graph for each combination of values of mu , k and mk . There are 9 combinations and thus 90 graphs. The test set is also constituted of 90 graphs and they it is built with the same method.

In our experiments, we train one model per graph of the train set. Figure 3 (resp Fig. 4) shows the average results when classifier is trained on graphs generated with $\mu = 0.2$, and tested on graphs generated with $\mu = 0.2$ (resp. $\mu = 0.4$).

It shows a F-measure around 0.7 in both of these cases that are typical of the other results. It is much higher than a random classifier and thus demonstrates the relevance of our supervised approach. Finally, we observe that the feature contributions to the model is balanced, showing the relevance of this basic set of features.

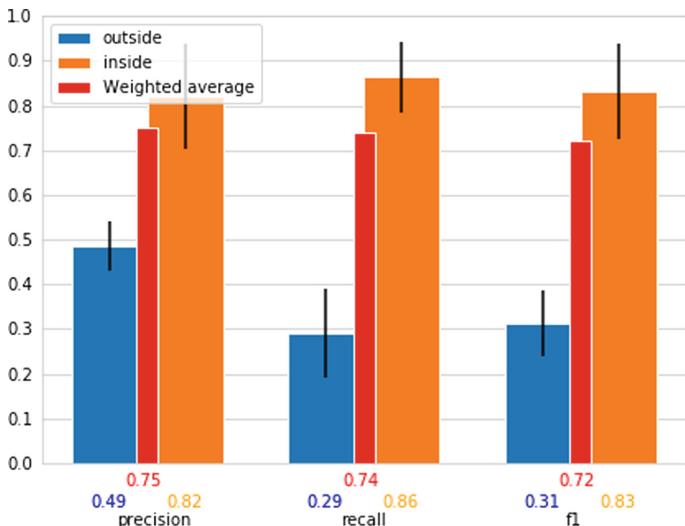


Fig. 3. Average classification results on the set of graphs with $\mu = 0.2$ with a model trained on others graphs with the same mu.

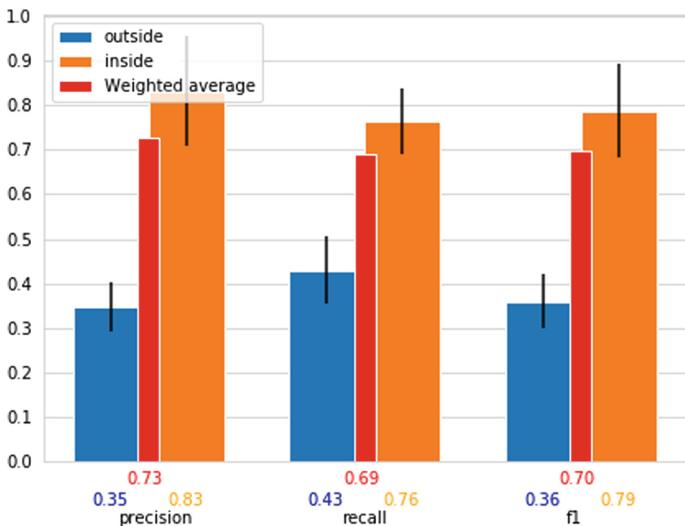


Fig. 4. Average classification results on the set of graphs of $\mu = 0.4$ with a model trained on graphs of $\mu = 0.2$.

5 Supervised Label Propagation: SLP

To further test the effect of supervision on the task of community detection, we introduce Supervised Label Propagation (SLP). The algorithm is mostly based on Label propagation as described by Raghavan et al. [14].

Supervised community detection has already been studied by Van Laarhoven and Marchiori [17] for the case of unweighted networks, with an algorithm mostly based on the classifier predictions. More recently, Bruma and Li [3] studied the case of unweighted networks with an approach based on the spectral properties of the networks. As far as we know, our approach is the first to study the case of weighted networks. Furthermore, our work introduces a novel semi-supervised algorithm combining classifier predictions with new label propagation heuristics. As we shall see, results show the interest of this approach.

However, SLP uses a classifier as an input and a confidence score as a parameter. The classifier and this parameter are used to split the links in three sets: *outside* links, *inside* links, and links the algorithm cannot predict the class with enough confidence. Inside links are then used to initialize the algorithm. In the original algorithm, each node is assigned to its own community. In our version, nodes that are connected by links considered as *inside* links by the classifier are assigned to the same community. Afterwards, during the label propagation phase, labels are not propagated through links that were classified as *outside* links, as one can see Algorithm 1.

Algorithm 1 Supervised Label Propagation

```

Require:  $G$  a graph,  $C$  a classifier,  $conf$  a confidence score
 $outside, inside, others \leftarrow C.predict(G.nodes, conf)$ 
 $communityCounter \leftarrow 0$ 
for  $n$  in  $G.nodes()$  do
    if  $n.c$  is empty then
         $n.c \leftarrow communityCounter$ 
        for  $v \in inside[n]$  do
             $v.c \leftarrow communityCounter$ 
        end for
         $communityCounter \leftarrow communityCounter + 1$ 
    end if
end for
while  $convergence$  do
    for  $n$  in  $G.nodes()$  do
         $labels \leftarrow n.neighbors() - outside[n]$ 
         $label \leftarrow labels.mostcommon()$ 
         $n.c \leftarrow label$ 
    end for
end while

```

To test the performances of SLP, we learn one model per graph in our training set. Then, each of these models is tested on the graphs of our test set. Results are

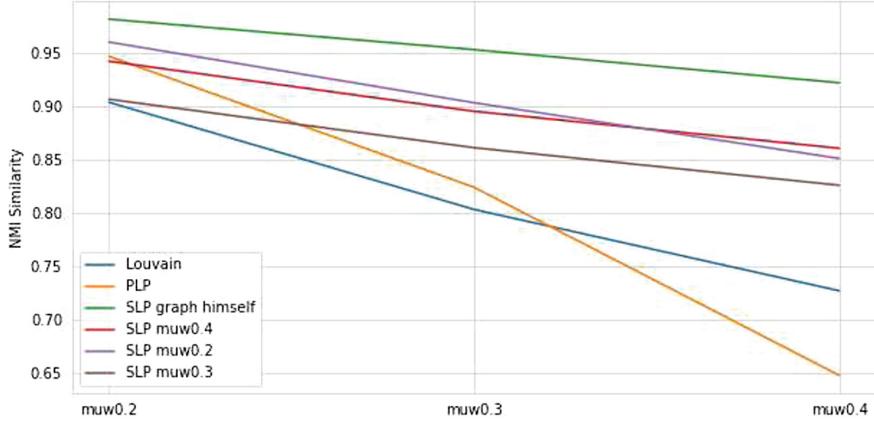


Fig. 5. Average NMI comparison of SLP considering training and test on distinct μ , Propagation Label (PLP) and Louvain.

finally averaged and presented in Fig. 5. As one can see, information extracted using the classifiers help SLP to converge towards a better solution than classic Label Propagation algorithm most of the time. It is especially the case when dealing with higher μ . Similar results are obtained with ARI similarity.

6 Conclusion and Future Work

In this paper, we investigated how supervision can help community detection. To that aim, we built a benchmark of experiments based on the LFR. On these graphs, we first demonstrated that pre-processing the graphs weights strongly influence community detection. In particular, experiments show that PMI and PPMI worsens the results whereas Glove-like normalization improves them. As shown by further experiments, this is probably due to the higher weights of the links that connect nodes of different communities (*outside* links). This analysis caused us to consider training classifiers to distinguish *outside* from *inside* links, and to evaluate their performances. We showed that with a few features describing local information and easy to compute, a trained classifier can predict quite efficiently links classes. The set of features used is basic, it would be interesting to consider global features, such as edge centralities [5] to enhance the performance of the classifiers in future work. The classifier itself is not enough, but it is useful to guide the community detection. The algorithm we introduce, SLP, demonstrates indeed that information provided by the classifier improves the community detection results. It would be relevant to compare in further work the performance of SLP with similar approaches such as those of Van Laarhoven and Marchiori [17], and Bruma and Li [3]. Other semi-supervised approaches using prior information during the learning phase could also be considered [18].

In our case, the nice results obtained on artificial graphs raise the following question: can these results be generalized to real networks? The perspectives of

this work are mostly based on this question. First, the experiments show that supervised models are more efficient if they are used to predict links classes on a graph which is similar to the one they were trained on. In particular, μ is very important, but difficult to estimate with no groundtruth. However, recent work show that in the case of unweighted networks, LFR graphs can be used to train models that can then be used efficiently on real graphs [10]. Second, LFR benchmark may not reflect perfectly the topology of graphs that model real-life systems. Are the weights of links connecting distinct communities higher in average in real graphs? It would enhance the robustness of the bridges between communities. Furthermore, it would be consistent with the work of Dugué et al. [6] that shows that nodes with huge degree tend to be highly connector nodes.

References

1. Barthélémy, M., Barrat, A., Pastor-Satorras, R., Vespignani, A.: Characterization and modeling of weighted networks. *Phys. A Stat. Mech. Appl.* **346**(1), 34–43 (2005)
2. Blondel, V.D., Guillaume, J.L., Lambiotte, R., Lefebvre, E.: Fast unfolding of communities in large networks. *J. Stat. Mech. Theory Exp.* **2008**(10), P10,008 (2008)
3. Bruna, J., Li, X.: Community detection with graph neural networks. [arXiv:1705.08415](https://arxiv.org/abs/1705.08415) (2017)
4. Chen, T., Guestrin, C.: Xgboost: A scalable tree boosting system. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 785–794. ACM (2016)
5. De Meo, P., Ferrara, E., Fiumara, G., Ricciardello, A.: A novel measure of edge centrality in social networks. *Knowl.-Based Syst.* **30**, 136–150 (2012)
6. Dugué, N., Labatut, V., Perez, A.: A community role approach to assess social capitalists visibility in the twitter network. *Soc. Netw. Anal. Min.* **5**(1), 26 (2015)
7. Hubert, L., Arabie, P.: Comparing partitions. *J. Classification* **2**(1), 193–218 (1985)
8. Lancichinetti, A., Fortunato, S.: Benchmarks for testing community detection algorithms on directed and weighted graphs with overlapping communities. *Phys. Rev. E* **80**(1), 016,118 (2009)
9. Levy, O., Goldberg, Y., Dagan, I.: Improving distributional similarity with lessons learned from word embeddings. *Trans. Assoc. Computat. Linguistics* **3**, 211–225 (2015)
10. Lu, X., Kuzmin, K., Chen, M., Szymanski, B.K.: Adaptive modularity maximization via edge weighting scheme. *Informat. Sci.* **424**, 55–68 (2018)
11. Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. *Advanc. Neural Informat. Process. Syst.* 3111–3119 (2013)
12. Newman, M.E.: Analysis of weighted networks. *Phys. Rev. E* **70**(5), 056,131 (2004)
13. Pennington, J., Socher, R., Manning, C.: Glove: Global vectors for word representation. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pp. 1532–1543 (2014)
14. Raghavan, U.N., Albert, R., Kumara, S.: Near linear time algorithm to detect community structures in large-scale networks. *Phys. Rev. E* **76**(3), 036,106 (2007)

15. Sarkar, S., Dong, A.: Community detection in graphs using singular value decomposition. *Phys. Rev. E* **83**, 046,114 (2011)
16. Strehl, A., Ghosh, J.: Cluster ensembles—a knowledge reuse framework for combining multiple partitions. *J. Mach. Learn. Res.* **3**(Dec), 583–617 (2002)
17. Van Laarhoven, T., Marchiori, E.: Network community detection with edge classifiers trained on LFR graphs. In: ESANN (2013)
18. Wang, J., Leng, M.: A new active learning semi-supervised community detection algorithm in complex networks. In: Proceedings of Recent Developments in Mechatronics and Intelligent Robotics (2019)



Is it Correct to Project and Detect? Assessing Performance of Community Detection on Unipartite Projections of Bipartite Networks

Tristan J. B. Cann^(✉), Iain S. Weaver, and Hywel T. P. Williams

College of Engineering, Mathematics and Physical Sciences, Harrison Building,
Streatham Campus, University of Exeter, North Park Road, EX4 4QF Exeter, UK

tc471@exeter.ac.uk

Abstract. Many real-world systems can be represented by bipartite networks that link two classes of node. However, methods for analysing bipartite networks are not as well-developed as those for unipartite networks. In particular, community detection in bipartite networks is often approached by projecting the network onto a unipartite network incorporating just one of the bipartite node classes. Here we apply a simple model to generate bipartite networks with prescribed community structure and then test the performance of community detection using four different unipartite projection schemes. Several important performance issues emerge from this treatment, particularly when the original bipartite networks have a long-tailed degree distribution. We find that a “hyperbolic” projection scheme mitigates performance issues to some extent, but conclude that care must be taken when interpreting community detection algorithm performance on projected networks. These findings have implications for any scenario where a unipartite projection is analysed in place of a bipartite system, including common applications to online social networks, social media and scientific collaboration networks.

1 Introduction

Bipartite networks are a useful representation of many real-world systems, but methods for their analysis lag behind those for unipartite networks. In bipartite networks the nodes can be divided into two disjoint sets called modes, with the additional restriction that edges exist only between nodes in different modes. Community detection in particular is difficult for large bipartite networks due to computational issues, leading many studies to approach the task by applying unipartite community detection methods to a unipartite projection of the system. Unipartite projection infers a new network linking nodes from one mode of the bipartite network, with edges created when two nodes share a neighbour in the bipartite network. The intention with this approach is that community detection on the unipartite network will capture community structure in the

bipartite network. In this study, we assess the efficacy of this approach, applying four different unipartite projection schemes to bipartite networks generated with known community structure and then seek to recover that structure from the unipartite projections.

For small bipartite networks, various community detection algorithms have been shown to be effective. Barber [1] introduced a form of modularity metric tailored to bipartite networks, adapting the null model to respect the criterion that edges must only exist between nodes in different modes. Several other algorithms, including for weighted bipartite modularity maximisation, are reported in [2]. However, optimisation of bipartite modularity (e.g. through implementations such as the Modular package [12] or those reported by [2]) poses high computational demands and may not scale to large networks from (e.g.) online social media. Furthermore, community detection methods tailored for bipartite networks are less well-known and less widely implemented in common network analysis packages, motivating many researchers to seek to apply unipartite methods to bipartite networks via unipartite projection.

Unipartite projection is applied widely in the study of social media user activity. Del Vicario et al. [6] study the interaction of Facebook users with pages around the 2016 EU Referendum in the UK. They make use of the unipartite projection onto page nodes to identify communities of pages with which particular user-groups interact. Schmidt et al. [16] use a similar methodology to understand groups of Facebook users who frequently liked or commented on the same content. Twitter posts have also been studied using community detection on a unipartite projection. Williams et al. [19] examine the news-sharing habits of Twitter users by projecting onto the article network. Community detection on the article network revealed clusters of news domains that were frequently shared by the same people.

Each of the previous examples make explicit reference to the methodology of unipartite projection, but other studies do so implicitly in their network construction. Starbird [17] studies the alternative news domains shared by Twitter users around mass shooting events. This implicit projection from a two- to one-mode network is still subject to the biases inherent in the projection scheme.

Despite frequent application of projection and community detection in empirical work, there has been limited theoretical study of the relationship between bipartite community structure and the community structure recovered from unipartite projection. Everett and Borgatti [7] show that despite the loss of information inherent in unipartite projection, meaningful statistics can be recovered by considering both projections together. Melamed [13] uses this dual-projection approach to refine the community structure of the unipartite projections to find communities in the bipartite network. Although there is increased accuracy when considering both projections together, many applications are only interested in a single mode, and hence a single unipartite projection. Newman [15] recognised the disproportional influence of high-degree nodes in unipartite projections, and introduced a method of weighting edges to mitigate this impact. Guimerà et al. [8] introduce a model for bipartite networks with a known community structure, parameterised by a “team preference” p determining the likelihood of an edge

existing within a community. They also extend modularity with a new measure for bipartite networks, and test this against different projection weightings, finding that bipartite modularity and weighted projection modularity performed similarly well. Bongiorno et al. [4] use the concept of statistically validated networks to detect stable cores in the bipartite communities, which can then be used to inform community detection. Li and You [11] explore how different types of unipartite projection affect various network metrics. They found that some metrics such as clustering coefficient were affected, but others such as degree correlation were not.

Increasing study of social systems (including e.g. social media and online social networks, as well as scientific citation and collaboration networks) implies that many applications of the unipartite projection approach to community detection occur on bipartite networks with some form of long-tailed degree distribution. In this case, the majority of vertices have few connections, but some may have many thousands of incident edges. Projection from such high-degree nodes in the bipartite network creates dense cliques in the unipartite network. Careful consideration needs to be given to how the properties of the bipartite degree distribution influence the structure of the projected unipartite network and the performance of community detection methods.

In this study, we explore the performance of community detection in bipartite networks using the unipartite projection approach, in particular for the case where the bipartite network has a long-tailed degree distribution. We first apply generative models to create ensembles of bipartite networks with different degree distributions and known community structure. Then we take unipartite projections using four different weighting schemes before testing the ability of community detection algorithms to recover this known structure. Section 2 outlines the network model, the weighting schemes used and the metrics used to measure community detection performance. Section 3 details the findings of our experiments and Sect. 4 discusses their consequences.

2 Methods

This section outlines the generative bipartite network model, the weighting schemes used in the unipartite projections, and the process of testing the accuracy of community detection after projection. For the rest of this paper, we use left and right to label to the two modes in our bipartite networks. The experimental approach used to generate the results from these methods is presented in Sect. 3.

2.1 Bipartite Network Models

The space of possible designs for random bipartite networks is enormous; network properties like edge density, vertex degree distribution, and correlation coefficients can be varied almost independently and may result in dramatically different structures and behaviour. We aim to use minimally assumptive models where we prescribe only the degree distribution and no vertex correlations beyond those necessary to impose community structure on the network. Even

in this space, there are a large number of intuitive degree distributions which could be implemented. Here we choose a generative model with a simple physical interpretation, described in detail by [18].

Vertices are introduced to the network at a fixed rate, along with edges which are more likely to connect to vertices which already have connections. A parameter m gives the relative importance of preferential attachment for the assignment of new edges. The cases we will study are: $m \rightarrow \infty$ (no preferential attachment) and finite m (giving strong preference for high degree vertices). In both cases, an interesting degree distribution still arises from the accumulation of edges on older vertices, even in the absence of preferential attachment. The main difference is that finite m (preferential attachment) yields a zeta distribution (or colloquially a power law) in the tail of the distribution, where the vertex degree k is large. Thus preferential attachment allows extremely well connected vertices to emerge, a common feature of complex self-organizing structures in nature and human-society [14]. This model accurately mirrors real-world phenomena, such as the distribution of page interactions on Facebook [6]. Setting $m \rightarrow \infty$ (no preferential attachment) produces a geometric degree distribution. Considering the finite m and $m \rightarrow \infty$ cases therefore allows for a comparison of how different types of heavy-tailed distribution affect the projection process.

Previous work by Weaver [18] gives the theoretical steady state degree distribution of randomly grown networks under different preferential attachment conditions (here adjusted for ease of computation) as:

$$\text{pmf}(k) = \frac{m + \delta}{m(\delta + 1) + 1} \frac{(m)(\frac{m}{\delta} + 2)}{(m + k)(\frac{m}{\delta} + 2)}$$

where we have used Pochhammer notation such that $(a)_b = a(a+1)\dots(a+b-1)$. Here δ is the number of edges added for each vertex that is added. Setting $\delta = m = 4$ here this simplifies (in the tail of large k) to $\text{pmf}_{\text{zeta}}(k) \sim k^{-3}$, that is, a zeta distribution representing preferential attachment. Setting $m \rightarrow \infty$ and $\delta = 4$ gives $\text{pmf}_{\text{geo}}(k) = \frac{1}{5} \left(\frac{4}{5}\right)^k$, that is, a geometric distribution from growth

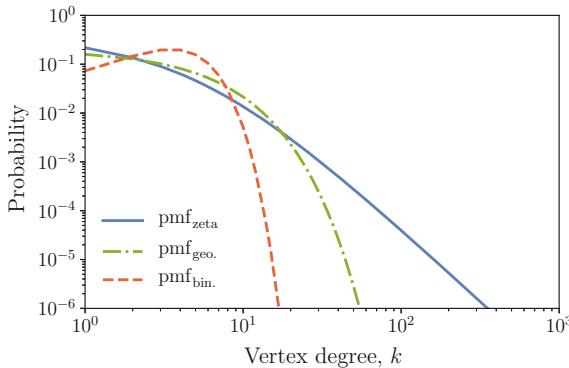


Fig. 1. Degree distributions used to generate bipartite networks. The distributions shown share identical means $\langle k \rangle = 4$, but vary in the weight of the tail as k increases.

without preferential attachment. In addition to these physically motivated models with and without preferential attachment, we provide an Erdős-Renyi bipartite graph for comparison, with degree distribution given by $\text{pmf}_{bin}(k) = \binom{\delta N}{k} N^{-k} \left(1 - \frac{1}{N}\right)^{\delta N - k}$. These degree distributions are plotted in Fig. 1.

We construct a random network for a chosen degree distribution by first selecting a number of vertices in the left- and right-modes, $N_l = N_r = N$ respectively. Then we assign each vertex a degree sampled from the chosen distribution. The same degree distribution is used to sample both modes and as such the two modes are indistinguishable from an analytic perspective. Edges are then added by joining pairs of randomly selected edge stubs. To add community structure, we follow the model of Guimerà et al. [8]. An underlying community structure is imposed by partitioning vertices into M equal-sized communities prior to the creation of edges. Each network edge has probability p of being constrained to connect two vertices in the same community. With complement probability $1 - p$ the edge connects any two vertices chosen irrespective of their community assignment. Notably, the fraction of all edges which join vertices belonging to the same community is not simply p , but can easily be shown to be $p + \frac{(1-p)}{M}$. A plurality of vertices have degree $k = 0$, a characteristic frequently mirrored in real-world citation networks [10]. These singletons are discarded before any further analysis. The giant components of sample networks drawn from this model, and their projections, can be seen in Fig. 2.

2.2 Unipartite Projection and Edge-Weighting Schemes

Unipartite projection creates an edge between each node pair of the chosen class that shares a neighbour in the other class. An important aspect of this process is how edge weights are assigned in the projection; approaches vary from simply recording presence/absence of a connection to weighting by the number of shared neighbours. Here we apply four weighting schemes to the projection process and test their influence on the accuracy of community detection in the unipartite network.

A *simple* weighting scheme assigns a weight w_{ij} to each edge e_{ij} in the projected network, where w_{ij} is exactly the number of mutual neighbours of nodes i and j in the bipartite network. Under simple weighting, we use the bipartite adjacency matrix B (where the rows correspond to the left-mode and the columns correspond to the right-mode) to define the unipartite adjacency matrix:

$$U_{\text{simple}} = B^\top B. \quad (1)$$

Note that U is symmetric, and has elements along the diagonal meaning that it encodes a directed network with self-connections. For the purposes of this manuscript, we do not allow self-connections, and set the diagonal elements to zero.

A *binary* weighting scheme works similarly to the simple scheme, except that edge weight is truncated at 1; the number of shared neighbours is disregarded, only the presence or absence of at least one shared neighbour is recorded as a 1 or 0 respectively, such that $U_{\text{binary}} = \text{Integer}(\text{Boolean}(U_{\text{simple}}))$.

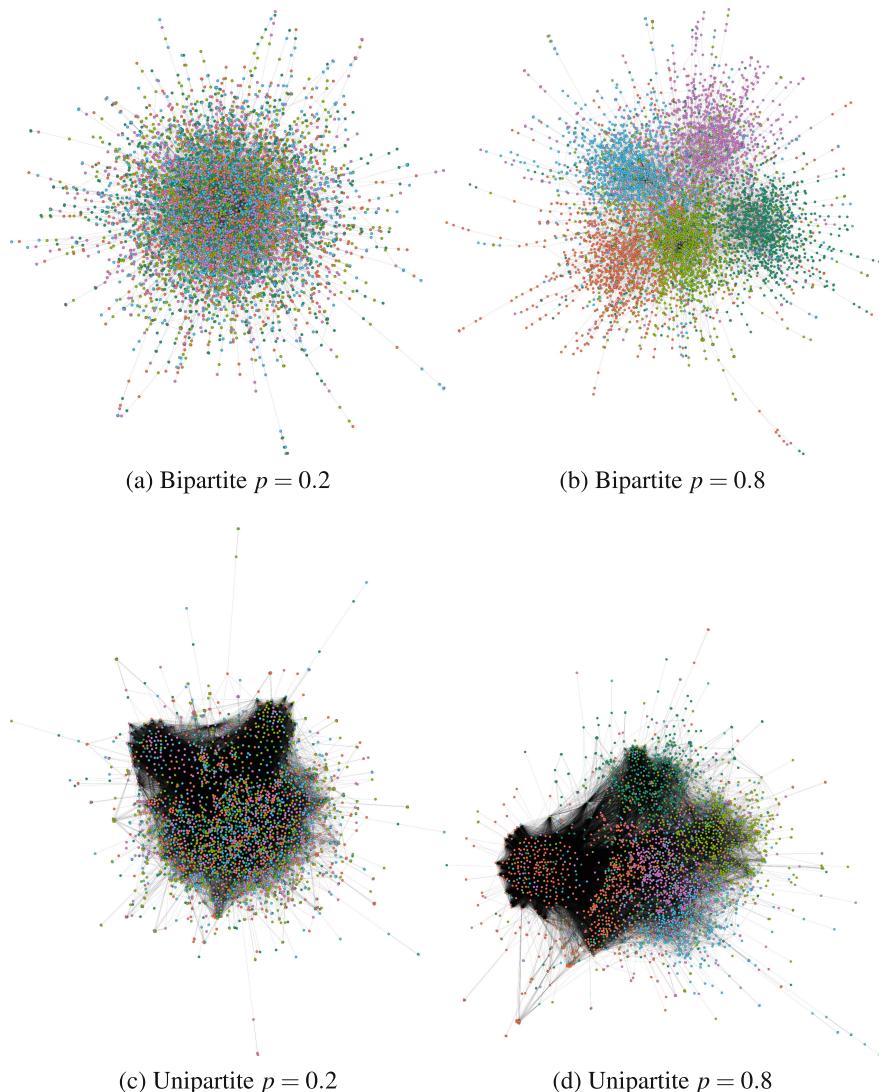


Fig. 2. Giant components of example bipartite networks generated by the model, and their unipartite projections: (a) and (c) use a community preference $p = 0.2$, (b) and (d) use $p = 0.8$. Each network degree sequence is sampled on 10^4 nodes. Node colour is determined by community assignment. Black areas indicate high edge density and highlight the increase locally under unipartite projection. Networks are visualised using Gephi with layout using the ForceAtlas2 algorithm [9].

A *hyperbolic* weighting was introduced by Newman [15] as a means of limiting the contribution of high-degree nodes in the bipartite network to the edge weights in the projected network. Consider that a bipartite node of degree k will create a total edge weight contribution of $\frac{1}{2}k(k - 1)$ under the simple weighting scheme;

then high- k nodes will have a large (and possibly disproportionate) influence on total edge weight in the projected network. The hyperbolic scheme rescales the edge-weight contribution from node i with degree k_i by $(k_i - 1)^{-1}$. Thus the edge weight contribution to the projected network becomes $\frac{1}{2}k_i$. Under hyperbolic weighting, the unipartite adjacency matrix is defined as:

$$U_{\text{hyper.}} = B^\top WB \text{ where } w_{ij} = \begin{cases} (k_i - 1)^{-1} & \text{if } i = j, \\ 0 & \text{otherwise.} \end{cases} \quad (2)$$

The *unary* weighting scheme is an extension of Newman's hyperbolic weighting to give each projected node a constant contribution to the total edge weight in the projected network, regardless of node degrees. The weighting scheme rescales the weight of each edge incident to node i by $2(k_i(k_i - 1))^{-1}$; each node has unit contribution to the total edge weight, independent of their degree. Under unary weighting, we define the unipartite adjacency matrix as:

$$U_{\text{unary}} = B^\top VB \text{ where } v_{ij} = \begin{cases} 2(k_i(k_i - 1))^{-1} & \text{if } i = j, \\ 0 & \text{otherwise.} \end{cases} \quad (3)$$

For the *hyperbolic* and *unary* weighting schemes, the matrices W and V are constructed such that the numbers of rows and columns are equal to the number of columns in B^\top .

2.3 Community Detection and Accuracy

We vary the value of p in the bipartite network models above to control the level of community structure in the constructed bipartite networks, where by design we have the true community label for each bipartite node. Then given a unipartite projection from a bipartite network we apply community detection to the unipartite network to recover community structure. We measure the level of accuracy in the returned community partition with respect to the known true partition.

For community detection we use the multilevel algorithm proposed by Blondel et al. [3]; this provides an efficient means to estimate the best community partition through modularity maximisation on the large and locally dense networks produced by our model.

To test the accuracy of the detected network communities we use adjusted Rand index to compare the true and predicted community partitions. The Rand index is defined as the ratio of all node pairs for which the two partitions agree with the total number of pairs. Here we use an adjusted form where the index is normalised by the expected level of agreement by chance, that is

$$\text{ARI} = \frac{\text{RI} - \mathbb{E}(\text{RI})}{\max(\text{RI}) - \mathbb{E}(\text{RI})}. \quad (4)$$

3 Results

All results displayed in this section are produced from averages taken across ensembles of 100 network instances. Each network consisted of $N = 10^6$ nodes (in each mode) with 4×10^6 edges between them. The community preference parameter was varied for $p \in \{0, 0.1, 0.2, \dots, 1\}$. For each value of p , networks were created for each degree distribution outlined by Fig. 1. As expected, each of the $M = 5$ communities contains an approximately equal number of edges and vertices from the left- and right-modes. Since the modes in each sampled network are, by design, indistinguishable, the choice of which mode to study is arbitrary. Our results focus on the projection onto the right-mode in each case. The multilevel community detection algorithm is applied to each projected network, and the resulting partition accuracy is evaluated by its adjusted Rand index and modularity.

A central consideration when gathering data from which to construct networks is sampling; if there is a real, underlying network, how thoroughly must we sample from one set of nodes in order to build a representative picture of the system as a whole? For example, if constructing a network of social media content views by sampling users from a given population, how many users are needed to give an accurate picture of the true network? Our constructed models of bipartite networks with different degree distributions allows us to answer this question. We consider the fraction of vertices in the right-mode in an underlying bipartite network that is reached by sampling an increasing fraction of the left-mode. Figure 3 shows the fraction of vertices contained in the sampled network and its giant component. For a geometric-tailed degree-distribution, and more so for the binomial distribution, we see a similar effect as [5], where a substantive sample is required to produce a sample network with a large component. Surprisingly this behaviour is not mirrored in the case of the long-tailed zeta distribution, where the existence of highly connected “hub” nodes means that a giant component emerges even for very small vertex samples.

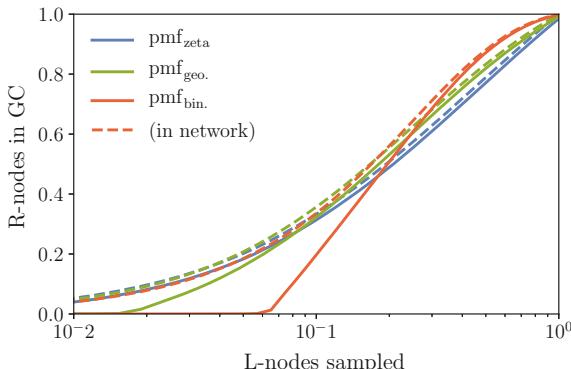


Fig. 3. Fraction of all R-nodes included in the sampled network and its giant component as a function of the fractional sample of L-nodes. Each network instance has 10^6 vertices in each of the L- and R-modes, 4×10^6 edges, and community preference $p = 0.5$.

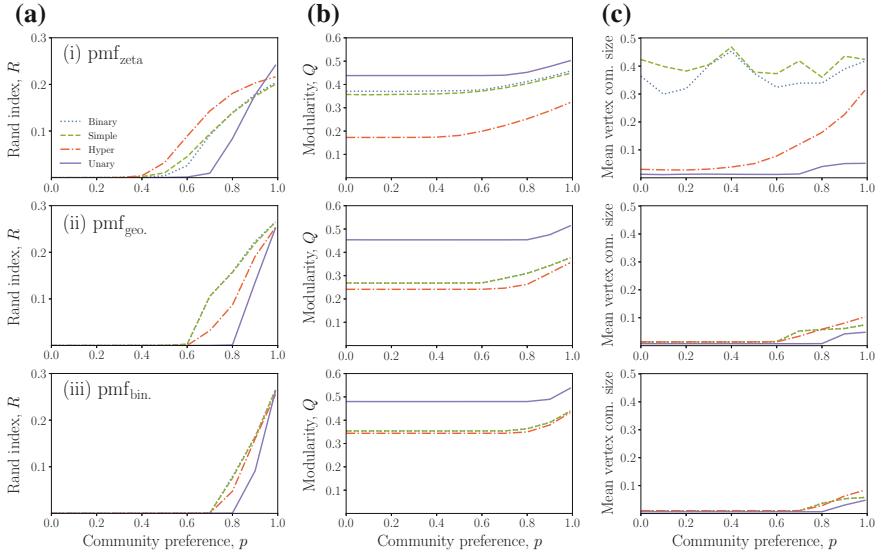


Fig. 4. Performance of community detection on unipartite projections. Left to right: (a) Adjusted Rand scores, (b) modularity, and (c) expected size of community, all plotted for unipartite projections of bipartite networks with varying levels of community structure. Top to bottom: (i) pmf_{zeta} , (ii) pmf_{geo} , and (iii) pmf_{bin} . bipartite degree distributions. Line colours and styles indicate different projection edge weighting schemes.

An understated computational challenge emerges in the projection of networks with heavy-tailed degree distributions; a vertex with k edges in the bipartite graph produces $\frac{1}{2}k(k-1)$ edges in a unipartite projection. The result is a dramatic increase in the edge density of the projected network. Our geometric degree distribution with 4×10^6 edges in the bipartite mode results in roughly 16×10^6 edges in the unipartite mode, or 120×10^6 edges if a long-tailed distribution is introduced. In the binomial case, there are only approximately 8×10^6 edges in the unipartite projection. The long-tailed degree distribution has increased the density of the network fifteen-fold compared to the binomial degree distribution. The number of projected edges is related to the second moment of the bipartite degree distribution; since the long-tailed distribution has a divergent second moment, it will increase arbitrarily with the size of the network under study, resulting in ever-more-dense unipartite projections.

Figure 4 shows the performance of community detection on the unipartite projections using three different metrics: adjusted Rand index (showing the agreement between node assignment in bipartite and unipartite partitions), unipartite modularity (showing the apparent community structure in the projection), and expected community size for a uniformly chosen node (expressed as a proportion of the total number of nodes in the network). Recovery of the underlying bipartite communities is affected by both degree distribution and projection weighting scheme. A common theme is that true communities can

only be recovered when there is a high level of imposed community structure in the bipartite network.

The adjusted Rand scores in Fig. 4a measure the similarity between communities detected in the unipartite network and the true communities, by counting the occurrence of ‘agreement’ between the partitions as to whether the nodes in each pair are assigned to the same or different communities. When the measure is close to zero, the detected unipartite communities are only as good as random community assignment. As it approaches unity, detected communities would perfectly recover the true partition. Figure 4a shows that for long-tailed zeta degree distributions, detected communities contain little information for $p < 0.4$, but become useful for further increases in p . The weighting scheme has a significant impact on performance, with hyperbolic weighting out-performing other schemes, and unary weighting performing much worse. The geometric-tailed and binomial degree distributions exhibit different behaviour under projection, with reasonable performance only achieved for extreme cases of strong bipartite community structure, above $p = 0.6$ and $p = 0.7$ respectively. Again the unary weighting scheme performs badly.

Some insights into why community detection on the unipartite projection may do a poor job in recovering bipartite communities are given by the modularity (Fig. 4b) and expected community size (Fig. 4c) results. Strangely, modularity in the unipartite projection is unaffected by the level of bipartite community structure for most values of p , with sensitivity only shown for high p . Furthermore, modularity can be relatively high (above 0.3) for many scenarios with low p . This indicates that the projected networks contain apparent community structure that is not present in the bipartite networks. The hyperbolic weighting scheme performs best in this regard, giving typically low modularity for low p , whereas the unary weighting scheme performs worst, giving high modularity for low p . Looking at the typical community sizes found in the unipartite networks, in most cases these are very small in the scenarios where adjusted Rand index indicates poor performance. At low p the small community sizes suggest that relatively high modularity is being found by partitioning into very many small communities. More meaningful communities emerge, as would be desired, for higher values of p . The striking behaviour of the binary and simple weighting schemes in Fig. 4c(i) shows that under the zeta degree distribution, average community size in the unipartite projection is large for all p . As the zeta distribution leads to nodes of very high degree, the unipartite projection is dominated by large cliques. The binary and simple weighting schemes have limited impact on the dominance of the hub nodes when forming communities, therefore communities will correspond to the neighbours of bipartite hub nodes in the unipartite projection.

4 Discussion

The Rand scores in Fig. 4 show that in some cases the true bipartite communities can be partially recovered through the unipartite projection process. However,

for this to be successful there must be strong community structure in the bipartite network (indicated by high p) to overcome the tendency of unipartite community detection to identify cliques created by high-degree bipartite nodes. In the absence of this condition, very little information about the true communities is found by modularity maximising community detection on the unipartite projection. This problem is exacerbated for degree distributions which do not have a long tail; highly connected vertices within a community are beneficial to the performance of community detection algorithms, and emerge only in long-tailed degree distributions.

The modularity scores found in the unipartite projections can be deceptive, with relatively high modularity being found in cases where there is no underlying bipartite community structure. In this case, the projection process is creating spurious community structure. In spite of this, there is some evidence that unipartite modularity does hold useful information; in the regime of high p , where community preference is strong, dense connections between cliques enable modularity optimisation to identify meaningful partitions. Considering the standard null-model with which network modularity is normally computed, unipartite projections violate the assumption that edges are independent; projection creates cliques, so that edges are not created independently. It is possible that an adjusted null-model can be devised to account for cliques and facilitate better community detection in projected networks.

Of the four weighting schemes studied here, the hyperbolic method performs most effectively with both geometric- and long-tailed zeta degree distributions. Given the Rand statistics in Fig. 4a, it is the most accurate once the underlying level of community structure is sufficiently strong. Hyperbolic weighting also performs surprisingly well at suppressing the inflated modularity measures, and producing meaningful community sizes when the bipartite community structure exists to be found.

Overall, the study here of community detection on the unipartite projection of bipartite networks with various levels of defined community structure shows that care needs to be taken around the application of this approach and the interpretation of results. In general, a more fruitful direction might be to develop better algorithms for community detection directly from bipartite networks; we note some recent efforts in this area [20]. However, the projection approach can be used in some circumstances. Community detection through modularity maximisation should be used sparingly on unipartite projections of networks that grow without an element of preferential attachment, as in these cases the detected modularity is uniformly very high regardless of the underlying community structure. On networks that experience preferential attachment during their growth process (which includes many real-world systems, especially social networks), the use of the hyperbolic weighting proposed by Newman [15] produces the most representative statistics of the true network topology. In any case, the modularity score of the unipartite projection by itself cannot be seen as representative of the community structure of the bipartite network due to the weaknesses of the typical modularity null model in this context. Future research

into a more suitable null model would be of benefit to the wider scientific community given the study of communities in projected networks across many subject areas.

Acknowledgements. TC is funded by an EPSRC Research Studentship (grant number EP/M506527/1). HW and IW acknowledge funding from ESRC (grant number ES/N012283/1) and NERC (grant number NE/P017436/1). We thank the two anonymous reviewers for their comments and the resulting improvements to this manuscript.

References

1. Barber, M.J.: Modularity and community detection in bipartite networks. *Phys. Rev. E* **76**, 066,102 (2007)
2. Beckett, S.J.: Improved community detection in weighted bipartite networks. *Royal Soc. Open Sci.* **3**(1) (2016)
3. Blondel, V.D., Guillaume, J.L., Lambiotte, R., Lefebvre, E.: Fast unfolding of communities in large networks. *J. Statist. Mech. Theory Exp.* **2008**(10), P10,008 (2008)
4. Bongiorno, C., London, A., Miccichè, S., Mantegna, : R.N.: Core of communities in bipartite networks. *Phys. Rev. E* **96**, 022,321 (2017)
5. Callaway, D.S., Hopcroft, J.E., Kleinberg, J.M., Newman, M.E.J., Strogatz, S.H.: Are randomly grown graphs really random? *Phys. Rev. E* **64**, 041,902 (2001)
6. Del Vicario, M., Zollo, F., Caldarelli, G., Scala, A., Quattrociocchi, W.: Mapping social dynamics on Facebook: the Brexit debate. *Soc. Netw.* **50**, 6–16 (2017)
7. Everett, M., Borgatti, S.: The dual-projection approach for two-mode networks. *Soc. Netw.* **35**(2), 204–210 (2013). Special Issue on Advances in Two-mode Social Networks
8. Guimerà, R., Sales-Pardo, M., Amaral, L.A.N.: Module identification in bipartite and directed networks. *Phys. Rev. E* **76**, 036,102 (2007)
9. Jacomy, M., Venturini, T., Heymann, S., Bastian, M.: ForceAtlas2, a continuous graph layout algorithm for handy network visualization designed for the Gephi software. *PLoS One* **9**(6), 1–12 (2014)
10. Larivière, V., Gingras, Y., Archambault, É.: The decline in the concentration of citations, 1900–2007. *J. Am. Soc. Informat. Sci. Technol.* **60**(4), 858–862 (2009)
11. Li, Y., You, C.: What is the difference of research collaboration network under different projections: topological measurement and analysis. *Physica A Statist. Mech. Appl.* **392**(15), 3248–3259 (2013)
12. Marquitti, F.M.D., Guimares, P.R., Pires, M.M., Bittencourt, L.F.: Modular: software for the autonomous computation of modularity in large network sets. *Ecography* **37**(3), 221–224 (2014)
13. Melamed, D.: Community structures in bipartite networks: a dual-projection approach. *PLoS One* **9**(5), 1–5 (2014)
14. Newman, M.: Power laws, Pareto distributions and Zipf's law. *Contemp. Phys.* **46**(5), 323–351 (2005)
15. Newman, M.E.J.: Scientific collaboration networks. ii. shortest paths, weighted networks, and centrality. *Phys. Rev. E* **64**, 016,132 (2001)
16. Schmidt, A.L., Zollo, F., Del Vicario, M., Bessi, A., Scala, A., Caldarelli, G., Stanley, H.E., Quattrociocchi, W.: Anatomy of news consumption on Facebook. *Proc. Nat. Acad. Sci.* **114**(12), 3035–3039 (2017)

17. Starbird, K.: Examining the alternative media ecosystem through the production of alternative narratives of mass shooting events on Twitter. In: Proceedings of International AAAI Conference on Web and Social Media(2017)
18. Weaver, I.S.: Preferential attachment in randomly grown networks. *Physica A Statist. Mech. Appl.* **439**, 85–92 (2015)
19. Williams, M., Cioroianu, I., Williams, H.: Different news for different views: political news-sharing communities on social media through the UK general election in 2015. In: Proceedings Workshop on News and Public Opinion (NECO) at International AAAI Conference on Web and Social Media (2016)
20. Zhou, C., Feng, L., Zhao, Q.: A novel community detection method in bipartite networks. *Physica A Statist. Mech. Appl.* **492**, 1679–1693 (2018)



Bayesian Complex Network Community Detection Using Nonparametric Topic Model

Ruimin Zhu^(✉) and Wenxin Jiang

Northwestern University, Evanston, IL 60201, USA

ruiminzhu2014@u.northwestern.edu, wjiang@northwestern.edu

Abstract. Network community detection is an important area of research. In this work, we propose a novel nonparametric probabilistic model for this task. We conduct random walks on the network and apply the Hierarchical Dirichlet Process topic model on the random walk data to explore the community structure of the network. Our work is among the very few endeavors in nonparametric probabilistic modeling in complex networks. Our proposed model is highly flexible. The nonparametric nature allows it to automatically detect the number of communities without prior knowledge. Our model is also quite powerful. It demonstrates significant improvements compared to other models in several experiments.

Keywords: Community detection · Bayesian modeling
Nonparametric topic model

1 Introduction

Complex networks usually exhibit community structure: nodes can be grouped such that within groups, nodes are densely connected while across groups, nodes are relatively sparsely connected. Given the vast existence of networks in human life and the importance of understanding their community structures, there is a rich set of literature on this task across various disciplines.

Random walks on networks reveal a lot information about the underlying structure of the networks and are thus used for community detection [39, 40, 42]. To illustrate, imagine a random walker visits the network. Given that some part of the network is densely connected, the path of the random walker tends to concentrate on local regions. Combining many random walks together can be an effective way of investigating the network topology. Furthermore, as a random walker chooses its next move solely based on its current neighborhood, this type of algorithm is relatively easy to implement.

Many community detection methods need prior knowledge to specify the number of communities. One way to handle this problem is by model selection, where we first fit several models, then conduct model selection to choose the

model that fits the data well and also has a relatively small model complexity. However, this approach is generally considered time-consuming. Nonparametric Bayesian models (BNP) address this issue in a very different way: they allow automatic model complexity adjustment, thus achieve both model flexibility and time efficiency.

Our model is based on random walks and nonparametric Bayesian modeling. We first conduct random walks on the network. We then fit the HDP topic model on the random walk data to reveal the underlying community structure. As corresponding to topic modeling in document data, here we treat nodes as words, random walks as documents, and clusters as topics. We use Stochastic Variational Inference [23] for fast model inference.

Our contributions are three-folds. First, the probabilistic nature of our model makes it principled and easy to interpret. Second, the nonparametric modeling approach makes our model flexible and require minimum prior knowledge about the network. Also, our model can adjust its complexity when new data become available, which makes it suitable for online learning to account for dynamic evolution of networks. Third, to the best of our knowledge, our work is among the very few that apply nonparametric topic models in social network research, especially in community detection.

The rest of the paper is organized as follows. In Sect. 2 we introduce the related works. In Sect. 3, we describe our method. In Sect. 4, we outline the experimental studies and compare our method to others. In Sect. 5, we provide analysis why our model outperforms others. In Sect. 6, we close with a conclusion and a discussion of possible future works.

2 Previous Work

Social network community detection has received intense research interest in the past decades. Due to the complexity of the task, there is no universal solution for community detection [27, 36]. In this section, we first present an overview of some major categories of techniques in community detection. We then further review works that are more closely related to our method.

One group of algorithms approaches the task by dividing the network into clusters such that the links within clusters are denser than that across clusters [3, 15, 26]. Other dissimilarity measures such as k-median, k-center, and k-clustering [4, 12, 31] can also be used. Partitional clustering algorithms divide networks into parts to minimize/maximize such measures. Hierarchical clustering algorithms [15, 16] iteratively partition the network into clusters at different granularities. The process generates a dendrogram and can be cut at a certain level to balance the tradeoff between granularity and predefined similarity measure. Spectral clustering methods [11, 14] are based on matrix eigen-decomposition. Another category of algorithms, e.g., the Girvan-Newman algorithm [18], performs community detection by removing edges between clusters iteratively to separate them. Optimization based methods such as modularity maximization algorithms [10, 20, 33, 34] devise proxies to approximate the quality of network communities, and optimize them to perform clustering. Clique-based

methods [13, 35] build upon clique–fully connected subgraph–detection in networks. Generative models assign probabilistic generative processes to networks. The most prominent one is the stochastic block model [24] and its various variants [1, 2, 25, 37].

Pons et al. [40] introduce a measure of similarity between nodes based on random walks. Rosvall and Bergstrom [42] utilize the probability flow of random walks on a network as a proxy for information flow to reveal community structure. Our work is distinct from theirs in that in our model, random walks are served as inputs for a probabilistic model. Recently, there have been work that consider network as a set of random walks and apply NLP techniques to explore the network structure [7, 19]. Perozzi et al. [39] conduct random walks on a network and treat them as sentences, later they used NLP techniques to map the nodes into a Euclidean space for various tasks including clustering analysis. While their work is a marriage between random walks and deep learning, ours is a marriage between random walks and topic model. What’s more, we directly study the community detection problem.

Another strand of research relates to our work is topic modeling. LDA topic model proposed by Blei et al. [5] has been applied in many domains such as document modeling [5], image processing [45], and information retrieval [48]. Zhang et al. [50] develop a method for network community detection based on LDA. In their model, a social interaction profile–neighbor(s) and second-order neighbor(s) of a node—is treated as a document and LDA is fitted to detect the topic structure of those social profiles. Compared to Zhang et al. [50], our model is more sophisticated. On the data side, we use random walks to collect more topological information about the network. On the probabilistic model side, HDP topic model is more powerful than LDA as the former embodies two advantageous merits—hierarchical and nonparametric.

Nonparametric Bayesian models demonstrate their power in a variety of tasks [22, 23, 46]. In social network research, Morup and Schmidt [32, 43] formulate a non-parametric Bayesian community generative model for social network analysis. Kim et al. [28] propose the hierarchical Dirichlet process relational model which allows nodes to have mixed membership in an unbounded set of communities. Guo [21] introduces a series of Bayesian nonparametric statistical models for community detection. Blundell and Teh [6] propose an efficient Bayesian nonparametric model for discovering hierarchical community structure in social networks. While these works directly assign generative models on the network itself, we apply the generative model on the random walk data induced by the network.

Note that Gerlach et al. [17] recently show that there is an equivalence between topic models and SBMs. Peixoto and Rosvall [38] also show a connection between dynamics on networks (e.g. random walks) and SBMs. Our current work helps strengthen these connections.

In relation to these works, our work essentially combines the data generation component of the random walk approach and the inferential component of the HDP topic model. We, therefore, name our model RW-HDP. This combination allows us to accomplish more than what previous methods can do.

3 Random Walk and Hierarchical Dirichlet Process

Table 1. Notation

Symbol	Network community detection	Document topic modeling
V	number of nodes	size of the vocabulary
D	number of random walks	number of documents
L	expected random walk length	expected document length
N_d	length of the d^{th} random walk	length of the d^{th} document
w_d	the d^{th} random walk	the d^{th} document
w_{dn}	n^{th} node in the d^{th} random walk	n^{th} word in the d^{th} document
z_{dn}	community assignment of w_{dn}	topic assignment of w_{dn}

In this section, we present RW-HDP for network community detection. We first introduce related terminologies and notations. We then describe the data generation process using random walks. Finally, we introduce the Hierarchical Dirichlet Process topic model.

3.1 Notations

Formally, let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be an undirected network, where \mathcal{V} is the set of nodes; $\mathcal{E} = \{e_{ij}\}$ is the set of edges and e_{ij} is the weight between node i and j . $V = |\mathcal{V}|$ is the number of nodes. As we treat each node as a word, V is also the vocabulary size. We assume networks are undirected but our model can be easily generalized to directed ones. Throughout the paper, we use terminologies in network community detection and document topic modeling interchangeably. Our notation is partly summarized in Table 1.

3.2 Random Walk Data Generation

The path of each random walker is treated as a document, and the collection of these paths is treated as the corpus. To carry out a random walk, we need to specify three factors: the starting point, the length, and the transition matrix.

The starting points are sampled independently uniformly from the nodes set \mathcal{V} . The lengths of random walks are sampled independently from Poisson distribution parametrized by L , the expected length of a random walk. For the transition matrix $P = (p_{ij})_{V \times V}$, we take the edge weights into consideration, i.e., $p_{ij} = \frac{e_{ij}}{\sum_j e_{ij}}$. A random walker at node i randomly chooses one of node i 's neighbors to visit in the next step with probability proportional to the edge weight between node i and its neighbors.

3.3 Hierarchical Dirichlet Process Topic Model

Here, we introduce the Hierarchical Dirichlet Process topic model [46] and show how to apply it for network community detection.

Hierarchical Dirichlet Process (HDP) topic model is a mixture model with an unbounded number of mixtures (topics). HDP topic model couples a set of document-level Dirichlet Process via a single top-level Dirichlet Process. In the top-level, the base measure H is a symmetric Dirichlet over the vocabulary simplex. It can be derived using stick-breaking construction [23] as follows.

1. Draw an infinite number of topics, $\phi_k \sim Dirichlet(\eta), k = 1, 2, \dots$
2. Draw corpus breaking proportions, $v_k \sim Beta(1, \gamma), k = 1, 2, \dots$
3. For each document:
 - (a) Draw document-level topic indexes, $c_{di} \sim Multinomial(\sigma(v)), i = 1, 2, \dots$
 - (b) Draw document breaking proportions, $\pi_{di} \sim Beta(1, \alpha), i = 1, 2, \dots$
 - (c) For each word:
 - (i) Draw topic assignment $z_{dn} \sim Multinomial(\sigma(\pi_d))$.
 - (ii) Draw word $w_{dn} \sim Multinomial(\phi_{c_{di}, z_{dn}})$.

Note that $\phi_k = p(w|z = k) = [p(w_1|z = k), \dots, p(w_V|z = k)]$ specifies the word distribution in the k^{th} topic. By stick-construction, the corpus level distribution of topics is given by:

$$p(z = 1) = \sigma_1(v) = v_1, p(z = k) = \sigma_k(v) = v_k(1 - \sum_{j=1}^{k-1} \sigma_j(v)), k = 2, 3, \dots \quad (1)$$

thus the overall probability of word i belonging to topic k is given by

$$p(z = k|w_i) \propto p(w_i|k)p(z = k) = \phi_{ki}v_k(1 - \sum_{j=1}^{k-1} \sigma_j(v)). \quad (2)$$

In HDP, variables such as ϕ_k and v_k are corpus or global level variables which govern the distributions of the observations (words) across all documents. Variables such as z_{dn} and π_{di} are document or local level variables which only determine the distributions of the observations in a particular document. The scopes of variables are shown in Fig. 1.

We apply Stochastic Variational Inference (SVI) [23] for our model inference. SVI combines stochastic sampling and Variational Bayesian Inference for fast posterior approximation¹.

In the HDP topic model, each observation is assumed to belong to a topic which might vary when the same word appears in different documents or even when it appears multiple times in the same document. For the network community detection task, to assign a node to a single community, we employ the

¹ Other constructions of the HDP topic model and the details of the Stochastic Variational Inference for the model can be found in [23].

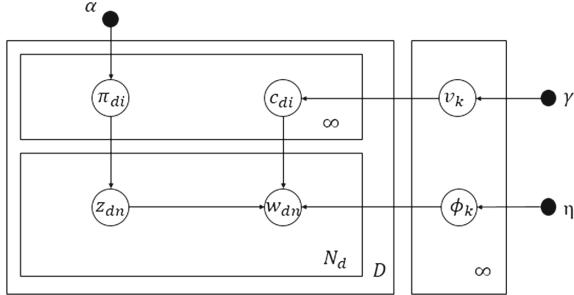


Fig. 1. HDP Topic Model. α, γ, η are hyperparameters controlling the prior distribution. ϕ, v are global level variables. z, π are local level variable.

Bayesian rule. After obtaining the probability $p(w|z)$ of a node appearing in a given community, and the community weight $p(z)$, by Bayesian theorem, the conditional probability of node belonging to a community is given by

$$p(z|w) = \frac{p(w|z)p(z)}{p(w)} \propto p(w|z)p(z). \quad (3)$$

We then assign the node to the community with the largest conditional probability.

4 Experiments

In this section, we evaluate and compare the performances of our model with three baseline models. We do the evaluation on five networks.

4.1 Models

We compare our model against three baseline models with each of them shares one characteristic with our model. RW-HDP consists of three major characteristics: random walk, nonparametric Bayesian, and topic model. We, therefore, choose the following three models to compare with:

1. SIP2-LDA [50], a topic based community detection model. In this model, each node is associated with a social interaction profile (SIP2), which only takes a node's immediate and secondary neighbors into consideration. Those social interaction profiles are treated as documents for community detection using Latent Dirichlet Allocation topic model.
2. Walktrap [40], a random walk based community detection model. It defines node-to-node distance and community-to-community distance based on properties of random walks, such as the transition probability between any pair of nodes within t steps. Later, it merges communities iteratively to get a hierarchical tree of partition. Finally, it cuts the tree to get the best partition.

3. BCD [32], a nonparametric Bayesian network generative model. The generative process is: first, a cluster assignment is generated using Dirichlet Process; then, within-cluster and between-cluster link probabilities are generated; finally, links between nodes are generated according to the within- and between-cluster link probabilities.

4.2 Choice of Hyperparameters

In RW-HDP, there are three types of hyperparameters: corpus, HDP topic model, and Stochastic Variational Inference hyperparameters. Here, we briefly explain the principles and intuitions behind the choice of hyperparameters.

The corpus hyperparameters are average random walk length L and number of random walks D . If L is too large, a document will contain words from many topics, which by intuition is not a well-written document and it indeed makes it hard for the topic model to detect any meaningful topic assignment. L cannot be too small either. An extreme case is $L = 1$, where it is impossible to capture any dependency between nodes. In our experiments, we set L to be around 100. The same principle applies to corpus size D . We set D to be approximately five times the number of nodes.

The HDP topic model has an infinite number of topics which is infeasible to implement in practice. Instead, we do truncations at both the corpus level and the document level. We fit K and T breaking points as the topic choices at the corpus level and document level respectively. Although we truncate the number of topics to be finite numbers, the model still behaves as if it has an infinite capacity as the training progresses, the posterior distribution always only concentrates on a small fraction of topics [23]. T is set to be much smaller than K as there might be hundreds of topics in the corpus, but a single document usually only covers a small set of these topics.

In Stochastic Variational Inference, hyperparameters include batch size and the number of epochs. The choice of batch size is a trade-off between speed and noise. We set the batch size to be a small number (less than 50) since the model is not sensitive to noise and speed is one of its priorities. We stop training when there is no significant dropping in perplexity scores.

4.3 Data

We conduct experiments on five medium-sized networks:

Table 2. Network statistics

Statistics	Yeast	GSE	ca-GrQc	ca-CondMat	US powergrid
Type	Biology	Biology	co-authorship	co-authorship	Engineer
Nodes	1540	9112	5242	16264	4941
Edges	8703	244928	14478	47594	6594

1. yeast: a yeast protein complex interaction network [49].
2. GSE: a breast cancer gene co-expression network [8,9].
3. ca-GrQc: Arxiv General Relativity and Quantum Cosmology collaboration network [29].
4. ca-CondMat: Arxiv Condense Matter Physics collaboration network [29].
5. US powergrid: the high-voltage power grid in the Western States of the United States of America [47].

The statistics of networks are summarized in Table 2.

4.4 Evaluation Metrics

We use community scoring functions for model performance evaluation. A good community tends to be densely connected internally and sparsely connected with other parts of the network. The community scoring functions quantify this intuition in different aspects [30].

Given a set of nodes S (a community of the network $\mathcal{G} = (\mathcal{V}, \mathcal{E})$), let $(S, S(\mathcal{E}))$ be the subgraph induced by S , where $S(\mathcal{E}) = \{(u, v) \in \mathcal{E} : u \in S, v \in S\}$. Let n denote the number of nodes in the original network. Let n_S denote the number of nodes in set S , and m_S the number of edges in set $S(\mathcal{E})$. Let c_S denote the number of edges with one end in S , and the other outside of S : $c_S = |\{(u, v) : u \in S, v \notin S\}|$. We use the following four community scoring functions:

1. Internal density: $D = \frac{2m_S}{n_S(n_S-1)}$. It sores the community structure based on its internal connectivity. A larger internal density usually means a better community structure [41].
2. Cut Ratio: $CR = \frac{c_S}{n_S(n-n_S)}$. It quantifies the community structure based on its external connectivity. A smaller cut ratio usually means a better community structure [15].
3. Conductance: $C = \frac{c_S}{2m_S+c_S}$. It measures the fraction of edge that points outside the cluster. A smaller conductance usually means a better community structure [44].
4. Modularity: $Q = \sum_{i=1}^m (e_{ii} - a_i^2)$, where m is the number of communities, e_{ij} the fraction of edges with one end in community i and the other in community j , $a_i = \sum_j e_{ij}$ [34].

For the two topic based models: SIP2-LDA and RW-HDP, we also compare their perplexity scores on the testing corpus. Perplexity is defined as:

$$\text{perplexity}(D_{test}) = \exp^{-\frac{\sum_{d=1}^M \log p(w_d)}{\sum_{d=1}^M N_d}},$$

where D_{test} is the testing corpus, which is either random walks in RW-HDP model, or randomly sampled social profiles in SIP2-LDA model. A smaller perplexity score corresponds to a better topic model.

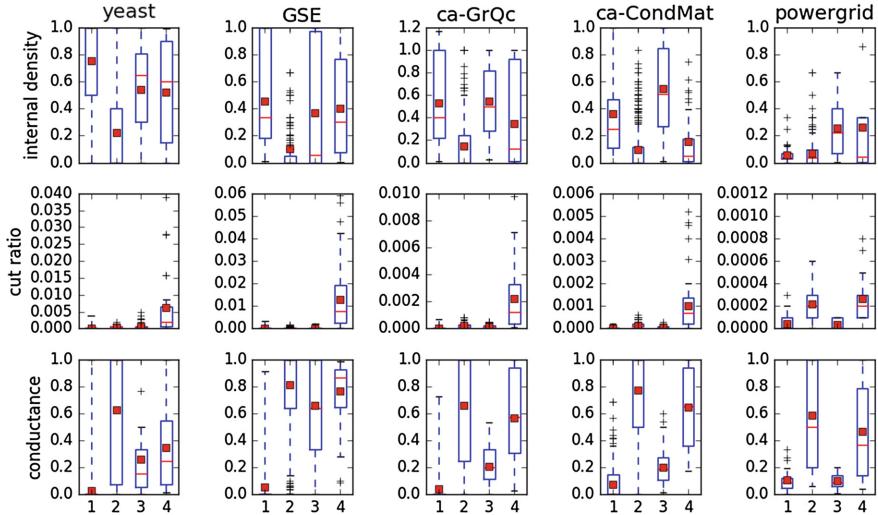


Fig. 2. Boxplots of internal density, cut ratio and conductance. 1: RW-HDP, 2: SIP2-LDA, 3: Walktrap, 4: BCD.

Table 3. Modularity

Model	Yeast	GSE	ca-GrQc	ca-CondMat	US powergrid
RW-HDP	0.7605	0.5967	0.7848	0.7588	0.9087
SIP2-LDA	0.6995	0.5881	0.7479	0.6615	0.7775
Walktrap	0.6968	0.6014	0.7430	0.7238	0.8953
BCD	0.6452	0.2017	0.5378	0.5041	0.4802

4.5 Results and Comparison

Figure 2 shows the comparison of four models on three community scoring functions. Table 3 shows the comparison on modularity score. Table 4 shows perplexity scores of two topic based models on their testing data. Our model outperforms other models almost in every aspect and on every dataset.

5 Discussion

RW-HDP outperforms SIP2-LDA because random walks are a much better way of exploring the topological structure of the network compared to the social

Table 4. Perplexity

Model	Yeast	GSE	ca-GrQc	ca-CondMat	US powergrid
RW-HDP	62.26	1124.51	504.16	1262.18	235.46
SIP2-LDA	279.95	1664.80	2902.81	41920.72	7197.49

interaction profile used in the latter. Also, HDP topic model has a larger model capacity than LDA.

Both RW-HDP and Walktrap utilize random walks for the community detection task. Walktrap adopts a greedy optimization scheme to maximize modularity score. Greedy algorithms can't guarantee optimal solutions. Furthermore, they could hurt the model's performance on other metrics.

As for BCD, though it is also a nonparametric Bayesian model, hierarchically, it has a relatively shallow depth, which could be the reason why it lacks the power to reveal the underlying community structure.

6 Conclusion

In this work, we present the RW-HDP model for network community detection. RW-HDP combines random walks and the Hierarchical Dirichlet Process topic model. We first conduct random walks on the network and treat them as documents. Later, we fit the Hierarchical Dirichlet Process topic model to reveal community structure. Our nonparametric view of the community detection task enables us to determine the number of communities without any prior knowledge. Our work is a new endeavor in nonparametric Bayesian modeling in social network research. Experiments on a variety of benchmarks show that our model outperforms others.

In future work, we plan to investigate overlapping community detection by allowing soft assignment of communities. We also want to explore other choices of topic models for the task. Furthermore, we want to extend the model to do node embeddings by using the conditional distribution of a node belonging to each community.

References

1. Airoldi, E.M., Blei, D.M., Fienberg, S.E., Xing, E.P.: Mixed membership stochastic blockmodels. *J. Mach. Learn. Res.* **9**(Sep), 1981–2014 (2008)
2. Ball, B., Karrer, B., Newman, M.E.: Efficient and principled method for detecting communities in networks. *Phys. Rev. E* **84**(3), 036,103 (2011)
3. Barnes, E.R.: An algorithm for partitioning the nodes of a graph. *SIAM J. Algebraic Discr. Methods* **3**(4), 541–550 (1982)
4. Bezdek, J.C.: Objective function clustering. In: Pattern recognition with fuzzy objective function algorithms, pp. 43–93. Springer (1981)
5. Blei, D.M., Ng, A.Y., Jordan, M.I.: Latent dirichlet allocation. *J. Mach. Learn. Res.* **3**(Jan), 993–1022 (2003)
6. Blundell, C., Teh, Y.W.: Bayesian hierarchical community discovery. In: Advances in Neural Information Processing Systems, pp. 1601–1609 (2013)
7. Bojchevski, A., Shchur, O., Zügner, D., Günnemann, S.: Netgan: Generating graphs via random walks. arXiv preprint [arXiv:1803.00816](https://arxiv.org/abs/1803.00816) (2018)
8. Chen, D.T., Nasir, A., Culhane, A., Venkataramu, C., Fulp, W., Rubio, R., Wang, T., Agrawal, D., McCarthy, S.M., Gruidl, M., et al.: Proliferative genes dominate malignancy-risk gene signature in histologically-normal breast tissue. *Breast Cancer Res. Treatment* **119**(2), 335 (2010)

9. Chen, Y., Xu, D.: Understanding protein dispensability through machine-learning analysis of high-throughput data. *Bioinformatics* **21**(5), 575–581 (2004)
10. Clauset, A., Newman, M.E., Moore, C.: Finding community structure in very large networks. *Phys. Rev. E* **70**(6), 066,111 (2004)
11. Donath, W.E., Hoffman, A.J.: Lower bounds for the partitioning of graphs. In: Selected Papers of Alan J Hoffman: With Commentary, pp. 437–442. World Scientific (2003)
12. Dunn, J.C.: A fuzzy relative of the isodata process and its use in detecting compact well-separated clusters (1973)
13. Everett, M.G., Borgatti, S.P.: Analyzing clique overlap. *Connections* **21**(1), 49–61 (1998)
14. Fiedler, M.: Algebraic connectivity of graphs. *Czechoslovak Math. J.* **23**(2), 298–305 (1973)
15. Fortunato, S.: Community detection in graphs. *Phys. Reports* **486**(3–5), 75–174 (2010)
16. Friedman, J., Hastie, T., Tibshirani, R.: The Elements of Statistical Learning, vol. 1. Springer series in statistics New York, NY, USA (2001)
17. Gerlach, M., Peixoto, T.P., Altmann, E.G.: A network approach to topic models. *Sci. Advanc.* **4**(7), eaaq1360 (2018)
18. Girvan, M., Newman, M.E.: Community structure in social and biological networks. *Proc. Nat. Acad. Sci.* **99**(12), 7821–7826 (2002)
19. Grover, A., Leskovec, J.: Node2vec: Scalable feature learning for networks. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 855–864. ACM (2016)
20. Guimera, R., Amaral, L.A.N.: Functional cartography of complex metabolic networks. *Nature* **433**(7028), 895 (2005)
21. Guo, J., Wilson, A.G., Nordman, D.J.: Bayesian nonparametric models for community detection. *Technometrics* **55**(4), 390–402 (2013)
22. Hjort, N.L., Holmes, C., Müller, P., Walker, S.G.: Bayesian Nonparametrics, vol. 28. Cambridge University Press (2010)
23. Hoffman, M.D., Blei, D.M., Wang, C., Paisley, J.: Stochastic variational inference. *J. Mach. Learn. Res.* **14**(1), 1303–1347 (2013)
24. Holland, P.W., Laskey, K.B., Leinhardt, S.: Stochastic blockmodels: first steps. *Soc. Netw.* **5**(2), 109–137 (1983)
25. Karrer, B., Newman, M.E.: Stochastic block models and community structure in networks. *Phys Rev. E* **83**(1), 016,107 (2011)
26. Kernighan, B.W., Lin, S.: An efficient heuristic procedure for partitioning graphs. *Bell Syst. Tech. J.* **49**(2), 291–307 (1970)
27. Khan, B.S., Niazi, M.A.: Network Community Detection: A Review and Visual Survey. arXiv preprint [arXiv:1708.00977](https://arxiv.org/abs/1708.00977) (2017)
28. Kim, D.I., Gopalan, P.K., Blei, D., Sudderth, E.: Efficient online inference for Bayesian nonparametric relational models. In: Advances in Neural Information Processing Systems, pp. 962–970 (2013)
29. Leskovec, J., Kleinberg, J., Faloutsos, C.: Graph evolution: densification and shrinking diameters. *ACM Trans. Knowl. Disc. Data (TKDD)* **1**(1), 2 (2007)
30. Leskovec, J., McAuley, J.J.: Learning to discover social circles in ego networks. In: Advances in Neural Information Processing Systems, pp. 539–547 (2012)
31. MacQueen, J., et al.: Some methods for classification and analysis of multivariate observations. In: Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, vol. 1, pp. 281–297. Oakland, CA, USA (1967)

32. Mørup, M., Schmidt, M.N.: Bayesian community detection. *Neural Computat.* **24**(9), 2434–2456 (2012)
33. Newman, M.E.: Fast algorithm for detecting community structure in networks. *Phys. Rev. E* **69**(6), 066,133 (2004)
34. Newman, M.E.: Modularity and community structure in networks. *Proc. Nat. Acad. Sci.* **103**(23), 8577–8582 (2006)
35. Palla, G., Derényi, I., Farkas, I., Vicsek, T.: Uncovering the overlapping community structure of complex networks in nature and society. *Nature* **435**(7043), 814 (2005)
36. Peel, L., Larremore, D.B., Clauset, A.: The ground truth about metadata and community detection in networks. *Sci. Advanc.* **3**(5), e1602,548 (2017)
37. Peixoto, T.P.: Hierarchical block structures and high-resolution model selection in large networks. *Phys. Rev. X* **4**(1), 011,047 (2014)
38. Peixoto, T.P., Rosvall, M.: Modelling sequences and temporal networks with dynamic community structures. *Nature Commun.* **8**(1), 582 (2017)
39. Perozzi, B., Al-Rfou, R., Skiena, S.: Deepwalk: Online learning of social representations. In: Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 701–710. ACM (2014)
40. Pons, P., Latapy, M.: Computing communities in large networks using random walks. In: International Symposium on Computer and Information Sciences, pp. 284–293. Springer (2005)
41. Radicchi, F., Castellano, C., Cecconi, F., Loreto, V., Parisi, D.: Defining and identifying communities in networks. *Proc. Nat. Acad. Sci. USA* **101**(9), 2658–2663 (2004)
42. Rosvall, M., Bergstrom, C.T.: Maps of random walks on complex networks reveal community structure. *Proc. Nat. Acad. Sci. USA* **105**(4), 1118–1123 (2008)
43. Schmidt, M.N., Morup, M.: Nonparametric bayesian modeling of complex networks: an introduction. *IEEE Signal Process. Mag.* **30**(3), 110–128 (2013)
44. Shi, J., Malik, J.: Normalized cuts and image segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.* **22**(8), 888–905 (2000)
45. Sudderth, E.B., Torralba, A., Freeman, W.T., Willsky, A.S.: Learning hierarchical models of scenes, objects, and parts. In: Tenth IEEE International Conference on Computer Vision, 2005. ICCV 2005, vol. 2, pp. 1331–1338. IEEE (2005)
46. Teh, Y.W., Jordan, M.I., Beal, M.J., Blei, D.M.: Sharing clusters among related groups: Hierarchical dirichlet processes. In: Advances in Neural Information Processing Systems, pp. 1385–1392 (2005)
47. Watts, D.J., Strogatz, S.H.: Collective dynamics of ‘small-world’ networks. *Nature* **393**(6684), 440 (1998)
48. Wei, X., Croft, W.B.: LDA-based document models for ad-hoc retrieval. In: Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 178–185. ACM (2006)
49. Yu, H., Braun, P., Yıldırım, M.A., Lemmens, I., Venkatesan, K., Sahalie, J., Hirozane-Kishikawa, T., Gebreab, F., Li, N., Simonis, N., et al.: High-quality binary protein interaction map of the yeast interactome network. *Science* **322**(5898), 104–110 (2008)
50. Zhang, H., Qiu, B., Giles, C.L., Foley, H.C., Yen, J.: An LDA-based community structure discovery approach for large-scale social networks. In: Intelligence and Security Informatics, 2007 IEEE, pp. 200–207. IEEE (2007)



Detecting Latent Terrorist Communities

Testing a Gower's Similarity-Based Clustering Algorithm for Multi-partite Networks

Gian Maria Campedelli^{1,2(✉)}, Iain Cruickshank², and Kathleen M. Carley²

¹ Università Cattolica del Sacro Cuore, L.go Gemelli 1, 20123 Milan, Italy
gianmaria.campedelli@unicatt.it

² Carnegie Mellon University, 5000 Forbes Avenue, Pittsburgh, PA 15213, USA
icruicks@andrew.cmu.edu, kathleen.carley@cs.cmu.edu

Abstract. Finding hidden patterns represents a key task in terrorism research. In light of this, the present work seeks to test an innovative clustering algorithm designed for multi-partite networks to find communities of terrorist groups active worldwide from 1997 to 2016. This algorithm uses Gower's coefficient of similarity as the similarity measure to cluster perpetrators. Data include information on weapons, tactics, targets, and active regions. We show how different dimensional weighting schemes lead to different types of grouping, and we therefore concentrate on the outcomes of the unweighted algorithm to highlight interesting patterns naturally emerging from the data. We highlight that groups belonging to different ideologies actually share very common behaviors. Finally, future work directions are discussed.

Keywords: Multi-partite networks · Unsupervised learning
Community detection · Terrorism

1 Introduction

In recent years, a slow but constant shift towards multidisciplinary dialogue between academic fields (e.g. criminology, political science, sociology, statistics, computer science) has directed the attention of the scientific community towards terrorism as a quantitatively measurable social phenomenon. Data on terrorism often involves different types of information: events, organizations, perpetrators, and tactics to name the most prominent. Since terrorism is a multifaceted and extremely complex issue, each of these different dimensions helped and may still help in understanding specific dynamics. Despite the progress made, Sageman [20] highlighted how terrorism research faces a stagnation which is mainly caused by the still limited availability of primary source information that are kept private by governments, leading to speculations with little empirical grounding in academia. Additionally, Sageman claims that this lack of data has not only

affected the solidity of academic results only but, also, the results achieved by the intelligence community which has the data but lacks methodological rigor. Although Sageman points out a relevant factual problem, our belief is that still a lot can be done in the effort of making open access data meaningful. In light of this, this work seeks to employ public event data on terrorism events from 1997 to 2016 to derive dynamic meta-networks on terrorist groups and subsequently test the performance of a new algorithm for community detection in multi-partite networks using Gower's Similarity Coefficient. The aims are to analyze how different weighting specifications of the algorithm affect the latent communities¹ and understand whether, besides ideology, purely operational and behavioral patterns can shed light on terrorist groups, highlighting hidden communities that may include actors which are actually considered distant or separate those actors that are usually seen as very similar. Furthermore, community detection can provide insights that may be relevant for policy strategies: if clear and distinct profiles exist, then different counter strategies should be deployed. The article is organized as follows: in the next section it presents related work on both network analysis and terrorism and previous clustering approaches to the phenomenon. In the third section, it will describe the data source and structure that will be employed in our analysis. The methodology section will then explain the structure of the Gower's Similarity Coefficient-based algorithm for finding communities in multi-partite networks. Results will then be presented, highlighting relevant findings. In the discussion section, results are reviewed with a particular focus on future work directions.

2 Related Work

The availability of open access data (combined with the diffusion of statistical software or data science oriented programming languages) has contributed to a change of perspective towards quantitative and computational approaches to the study of terrorism. From an applied network science standpoint, network analysis for the study of terrorism has been employed in several specific subdomains. A classic application aims at understanding and highlighting internal dynamics and roles within terrorist groups [1, 11, 12]. Shifting from the relational information gathered and structured to investigate roles and key players, scholars have also tested and simulated the strength and resilience of terrorist networks. Some of the works in these subdomains relied on mathematics to reveal network topologies and possible effective strategies to destabilize terrorist networks [5]. Furthermore, researchers integrated spatial and temporal dynamics to simulate the evolution of these networks, relying on the conceptual assumption that time and space are relevant features when aiming to analyze the phenomenon from the evolutionary standpoint [16, 17]. With the explosion of social media, the attention of researchers has been attracted to the possible consequences of criminal behavior in cyberspace. Indeed, a recent stream of research has started to focus on detecting terrorist and radical behaviors on these social media platforms,

¹ Throughout the paper, communities and clusters are used as synonyms.

such as Twitter [2, 10]. Finally, in the fourth and last subdomain, researchers are trying to use event data to reconstruct multi-mode or multiplex networks in order to predict future terrorist attacks, locations, and tactics. This is the most recent and underdeveloped field in which network science is being applied to terrorism [4, 7]. In another line of research, cluster analysis has not been extensively applied to the analysis of terrorism. In one of the first attempts at using cluster analysis to group terrorist organizations, Chenoweth and Lowham [6] used data on groups which targeted American citizens to explore alternative ways to conceive terrorist typologies. Qi et al. [18] used both social network analysis and unsupervised learning to group extremist web pages using an hierarchical multi-membership clustering algorithm based on the similarity score of these pages. Finally, Lautenschlager et al. [14] developed the Group Profiling Automation for Crime and Terrorism (GPACT) prototype that generates terrorist group profiling via a multi-step methodology that also includes clustering of terrorist events.

3 Data

The data used in this work come from the Global Terrorism Database (henceforth GTD) [13]. GTD includes information on terrorist attacks from 1970 to 2016. Information on worldwide attacks is retrieved from open sources and each event is required to meet certain criteria to be included in the dataset and labelled as a terrorist. Additionally, events which meet these criteria but have uncertainty as to whether they should be considered terrorist events are included in the dataset but mapped with the “doubter” variable. For our analysis, we used data from 1997 to 2016 on worldwide events (and related perpetrators), excluding all the attacks which were of doubtful terrorist nature. This methodological choice led from 106,114 events to a total of 88,513, and was the only pre-processing of the data performed. The meta-networks which have been created and employed for our study relied on six main terrorist dimensions, namely: Events ($N = 88,513$), Groups ($N = 1,494$), Targets ($N = 22$), Weapons ($N = 13$), Tactics ($N = 9$) and operating Regions ($N = 9$). Since a terrorist group can attack many different targets, use many different weapons, and operate in many different regions, and vice-versa, these data naturally form many-to-many relationships, and can therefore be easily modeled as networks. In addition to this information which represent the basis of this work, other variables extracted from the GTD will be employed to detect and assess behavioral patterns of terrorist groups belonging to the same clusters *ex post*. This information will include group based attributes regarding terrorist activity such as ideology, success rate, suicide rate, fatality rate, casualty rate, multiplot rate, international rate and number of targeted countries. The ideology of each group has been mapped using existing information present in two open access data sets (Big Allied and Dangerous 1 and an extraction of Big Allied and Dangerous 2) when that information was available within those sources, and by exception from other qualitative open access information sources. Finally, seven ideology categories were created: (i) Islamic/Jihadist groups, (ii) Left Wing/Anarchist, (iii) Right Wing/Racist,

(iv) Ethno-Nationalist, (v) Other/Unknown, (vi) Religious (Islam excluded), (vii) Animal-rights/Environmentalist. A given group may belong to more than one category at a time (e.g.: the Popular Front for the Liberation of Palestine which contains at the same time elements of Marxism and Nationalism). The success rate is given by the ratio between the successful attacks and the total number of events attributed to a given group. The suicide rate maps the ratio of suicide attacks over the total number of events plotted by the same group. Fatality and casualty rates are the ratios of attacks with at least one dead victim (fatality) or one wounded victim (casualty) person out of the total number of events. Finally, the international rate is simply the ratio between attacks with some international features (e.g. logistic organization) and the total number of attacks. All these variables seek to enrich the knowledge associated to each group and to understand whether the identified clusters highlight certain unexpected behaviors (Table 1).

Table 1. Group-based attributes on terrorist activity—descriptive statistics

	Mean	St. Dev.	Median	Min	Max
Events	60.24	1271.77	2.00	1.00	48,537
Success rate	0.90	0.24	1.00	0.00	1.00
Suicide rate	0.03	0.14	0.00	0.00	1.00
Fatality rate	2.75	8.55	0.50	0.00	170
Casualty rate	7.88	23.34	1.67	0.00	385.29
Multiplot rate	0.14	0.28	0.00	0.00	1.00
International rate	0.29	0.41	0.00	0.00	1.00
Targeted countries	1.46	4.57	1.00	1.00	163

4 Methodology

Since the variables of Targets, Weapons, Tactics, and Regions form a many-to-many relationship with Groups, we first model this data as a multi-partite network with each partition joined to Groups. Indeed, we define:

$$\mathfrak{G}^N := \langle (V_1, V_2, \dots, V_n), (E_{1,2}, E_{1,3}, \dots, E_{m,n}), (W_{E_{1,2}}, \dots, W_{E_{m,n}}) \rangle \quad (1)$$

as a multi-partite graph that contains N partitions describing relations between different sets of nodes V_m and V_n : these relations are formalized as edges $E_{m,n}$ that are weighted by $W \in \mathbb{Z}_{\geq 0}$ and each mode in the multi-partite network is represented as $G_{m,n} := \langle (V_m, V_n), E_{m,n}, W_{E_{m,n}} \rangle$. With this data structure we then employ Gower's Coefficient of Similarity [8] to place the terror organizations² in

² Throughout the work, Group, terror organization and organization are used as synonyms.

a latent space, whereby we can create a latent network of the organizations and assign these organizations to clusters based upon the multi-partite network. We use Gower's Similarity Coefficient defined as:

$$S_{ij} = \frac{\sum_{k=1}^r w_{ijk} S_{ij}^{(k)}}{\sum_{k=1}^v w_{ijk}} \quad (2)$$

where S_{ij} is the similarity between Groups i and j on a variable (i.e. Targets, Weapons, etc.), k and v is the total number of variables and w_{ijk} is the weight of the similarity between Group i and Group j for metric k . $S_{ij}^{(k)}$ is then dually defined as:

$$S_{ij}^{(k)} : \begin{cases} 1, & \text{if } (x_{ik} = x_{jk}) \neq \emptyset \\ 0, & \text{otherwise} \end{cases} \quad (3)$$

if the feature, k , is categorical (to include binary) for node i and j 's responses, x_{ik} , x_{jk} , and:

$$S_{ij}^{(k)} : \frac{|x_{ik} - x_{jk}|}{r_k} \quad (4)$$

where r_k is the range of x_k , if k is numerical. Since we are interested in how the different variables, or modes of the multi-partite network, affect the possible latent network and clusters of the terrorist organizations, the weighting term will take values of:

$$w_{ijk} = \frac{N}{\sum_n^N \delta(k, n')} \quad (5)$$

where N is the total number of n modes (in this case, $N = 4$), and $\delta(k, n')$ is an indicator function that returns 1 if k is within one of the specified important nodes, n' . Thus, we can use this weighting term to explore what happens when we consider certain modes, like Region or Tactics as more important to cluster formation than others, which allows us to investigate whether certain analytic theories regarding terrorist Group similarities are present in the data. Following the completion of the weighted pairwise Gower's Coefficient of Similarity calculation we are then left with an $N \times N$ affinity matrix that contains the pairwise similarities between each terrorist group and every other terrorist group. To cluster this affinity matrix into sub groups and create a latent network of the terrorist groups, we use k-NN network modularity maximization [19]. k-NN network modularity maximization takes an affinity or distance matrix and creates a k-NN graph where each node connects to its k nearest neighbors (Fig. 1). We applied k-NN network modularity maximization because the matrix is fully connected, thus impeding the use of Louvain, and spectral clustering is proved to be more efficient for sparser graphs [15]. Then, this graph is clustered using the Louvain method of modularity maximization [3]. The algorithm iterates over certain values of k and then selects the corresponding latent network and sub group assignments that maximize the modularity of the latent network. The following is a quick visual depiction of the k-NN network modularity maximization

procedure. In summary, our method for finding latent networks and groups of terrorist organizations, which allows for testing of analytic theories, is structured as follows:

1. Determine which modes (i.e. Tactics, Weapons, etc.) will be significant and construct the weighting matrix appropriately;
2. Calculate the Gower's Coefficient of Similarity between each Group and every other Group, using the weighting matrix, to form an affinity matrix;
3. Find latent graphs and sub-groups in the affinity matrix using the k-NN network modularity maximization procedure;
4. Compare the different clustering outputs and latent networks for the different modal weighting schemes to better understand their impact on terrorist groups

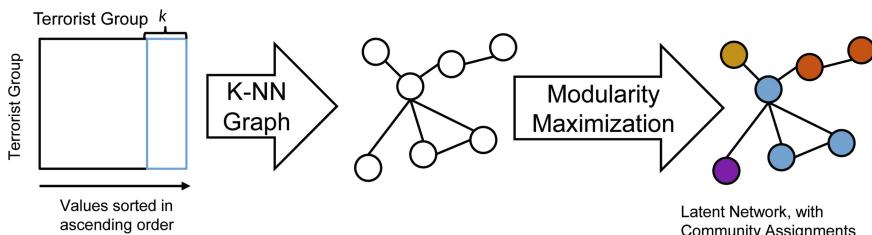


Fig. 1. The k-NN network modularity maximization procedure

Based on this algorithm, we test the relation between the different groupings emerging from the different weighting processes. The comparison will be based on the scores of two popular metrics for evaluating cluster similarity: the Adjusted Rand Index (ARI) and the Adjusted Mutual Information (AMI). A total of six models have been run. The first one assigns the same weight to each dimension (i.e.: no weights), the second one weights Region as the most important dimension, the third does the same with Tactic, and so on. In the last model, we used the "Ideology" attribute, which is generally seen as the main discriminant between terrorist groups, to test if it provides clustering assignments similar to the other models. This procedure seeks to test two working hypotheses:

- H1: Weighting differently a given terrorist dimension would lead to results that are considerably variable across different weighting schemes;
- H2: Weighting by ideology would lead to extremely different results compared with the other schemes, eventually posing the risk of missing relevant hidden patterns and feature clusters that arise regardless of ideology itself. Our intuition is that operational and behavioral characteristics are more fitting in explaining clustering rather than relying on mere political or religious motivations.

5 Results

The models run with different weighting schemes yield interesting results for both working hypotheses. Firstly, considering H1, both ARI and AMI measures vary with different ranges when we decide to weight differently a particular mode. Indeed, the ARI ranges from a minimum of 0.13 to a maximum of 0.36, indicating that the highest pairwise similarity is given when comparing models where Tactics and Weapons are considered more important. The AMI, although with different absolute values (the range is between 0.33 and 0.55), confirms this same results.

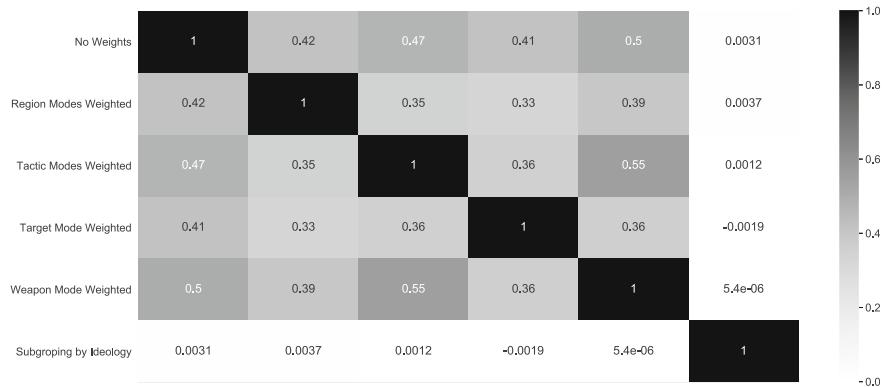


Fig. 2. Adjusted mutual information of different weighting schemes

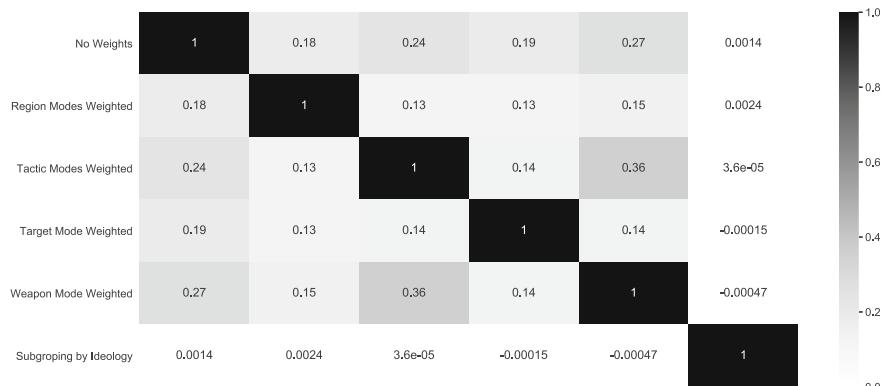


Fig. 3. Adjusted rand index of different weighting schemes

Therefore, we argue that applying different weights poses the risk of biasing outcomes that may naturally emerge using the algorithm treating all features as equal. Secondly, regarding H2, clustering on ideology provided communities that are completely different from all the other previous weighting schemes. As shown in Figs. 2 and 3, ideologically-driven clusters possess very little similarity (mostly around 0) in both AMI and ARI to our behavioral-based clustering. This confirms the hypothesis that, though most research and public debate usually distinguishes terror groups based on their ideologies and/or motivations, it may be more useful to look at operational and behavioral characteristics in order to find meaningful terrorist clusters. Investigating further, we now focus on the resulting clusters of the “No weights” model, analyzing the distribution across clusters of the variables described in the Data section, namely Success Ratio, Suicide Rate, Fatalities Rate, International Rate and the seven different mapped Ideologies. Our algorithm clustered the terrorist groups in the dataset in 37 distinct terrorist communities. The size of communities ranged from a minimum of 3 groups to a maximum of 159 groups per community (Fig. 5). Table 2 provides the descriptive statistics of each of the variables employed for the ex-post evaluation across clusters.

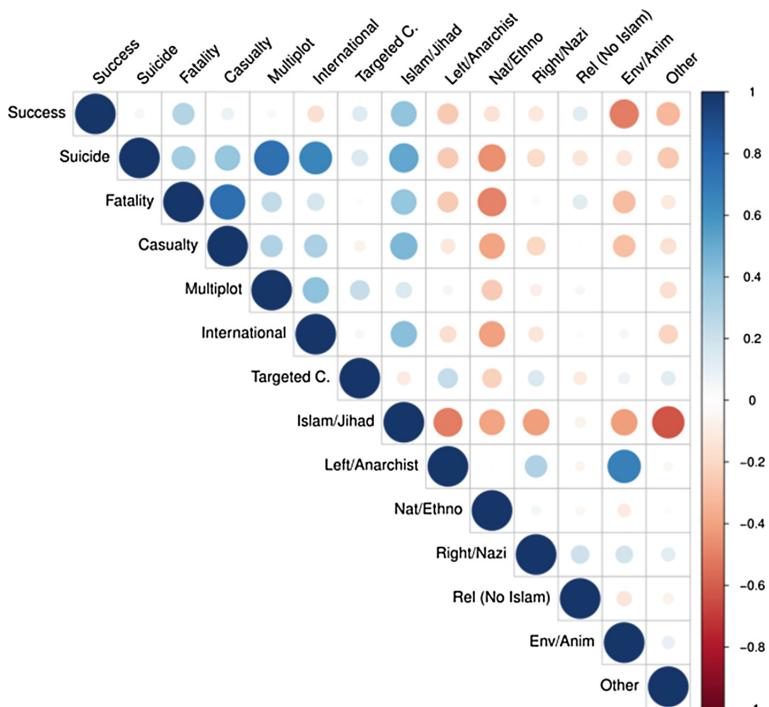


Fig. 4. Pearson’s correlation matrix of variables and ideologies across groups

Table 2. Variables distribution across resulting clusters. ANOVA tests revealed statistically significant differences across groups for all variables (Prob>F = 0.000)

C	Succ rate	Suic rate	Fatal rate	Casual rate	Multi rate	Intl rate	Targ count	Islam	Left	Nat	Right	Rel	Env	Oth
0	0.88	0.06	4.25	14.61	0.09	0.29	1.19	0.37	0.14	0.36	0.04	0.01	0.01	0.19
1	0.92	0.17	4.49	20.77	0.13	0.55	1.36	0.64	0.18	0.00	0.00	0.00	0.00	0.18
2	0.96	0.04	1.92	4.66	0.12	0.19	3.73	0.23	0.31	0.44	0.01	0.01	0.01	0.08
3	0.92	0.06	1.65	5.88	0.06	0.16	1.04	0.33	0.31	0.37	0.02	0.00	0.02	0.10
4	0.90	0.03	4.39	19.59	0.16	0.27	1.12	0.42	0.12	0.38	0.02	0.05	0.00	0.14
5	0.97	0.07	3.12	6.10	0.18	0.25	1.02	0.44	0.02	0.39	0.00	0.02	0.00	0.17
6	0.86	0.03	2.46	4.57	0.13	0.17	1.07	0.29	0.10	0.54	0.00	0.05	0.00	0.17
7	0.96	0.03	1.92	6.95	0.11	0.34	1.88	0.28	0.10	0.48	0.00	0.00	0.00	0.30
8	0.93	0.01	2.37	8.38	0.08	0.27	1.08	0.23	0.17	0.52	0.03	0.03	0.00	0.08
9	0.89	0.03	2.45	8.06	0.17	0.21	1.19	0.41	0.15	0.44	0.04	0.01	0.00	0.07
10	0.74	0.11	2.32	5.96	0.16	0.35	1.05	0.31	0.03	0.59	0.03	0.00	0.00	0.10
11	0.95	0.03	1.01	2.74	0.11	0.31	1.91	0.43	0.06	0.45	0.02	0.02	0.00	0.09
12	0.96	0.02	2.57	5.45	0.20	0.33	1.23	0.29	0.12	0.46	0.04	0.12	0.00	0.10
13	0.93	0.07	2.68	9.91	0.19	0.19	1.09	0.53	0.09	0.31	0.00	0.13	0.00	0.06
14	0.88	0.01	1.10	5.20	0.16	0.40	2.03	0.19	0.39	0.28	0.00	0.03	0.03	0.14
15	0.99	0.00	2.37	4.78	0.08	0.21	1.11	0.15	0.26	0.48	0.07	0.04	0.00	0.11
16	0.95	0.06	2.93	7.16	0.19	0.36	1.68	0.58	0.14	0.28	0.02	0.00	0.00	0.08
17	0.89	0.02	1.57	7.03	0.19	0.59	1.28	0.28	0.16	0.52	0.04	0.12	0.00	0.16
18	0.87	0.01	1.74	4.30	0.18	0.43	1.40	0.05	0.26	0.58	0.16	0.07	0.02	0.07
19	0.89	0.01	0.72	1.85	0.20	0.21	1.29	0.13	0.48	0.27	0.06	0.02	0.02	0.06
20	1.00	0.04	2.16	6.34	0.00	0.71	1.08	0.92	0.00	0.23	0.00	0.00	0.00	0.00
21	0.90	0.06	6.53	18.50	0.21	0.28	1.03	0.48	0.24	0.28	0.00	0.00	0.00	0.07
22	0.95	0.03	3.07	6.30	0.09	0.23	1.17	0.30	0.11	0.52	0.02	0.02	0.00	0.10
23	0.96	0.00	4.79	7.12	0.16	0.12	1.06	0.11	0.11	0.33	0.00	0.11	0.00	0.33
24	0.86	0.02	2.14	4.46	0.15	0.36	1.76	0.36	0.12	0.38	0.05	0.07	0.00	0.17
25	0.78	0.04	1.31	4.93	0.19	0.41	1.39	0.13	0.30	0.41	0.00	0.00	0.04	0.22
26	0.95	0.01	7.03	11.19	0.11	0.27	1.72	0.32	0.09	0.34	0.08	0.01	0.01	0.22
27	0.83	0.00	1.17	4.60	0.24	0.38	1.46	0.08	0.43	0.32	0.04	0.04	0.06	0.13
28	0.66	0.00	0.83	1.32	0.02	0.30	1.14	0.00	0.24	0.33	0.05	0.00	0.05	0.33
29	0.90	0.09	4.80	13.40	0.06	0.58	1.22	0.89	0.00	0.11	0.00	0.11	0.00	0.00
30	1.00	0.00	1.67	3.00	0.00	0.00	1.00	1.00	0.00	0.33	0.00	0.00	0.00	0.00
31	0.88	0.01	2.21	5.52	0.09	0.04	2.35	0.05	0.20	0.25	0.15	0.05	0.00	0.40
32	0.91	0.00	2.29	10.29	0.29	0.14	2.00	0.14	0.14	0.29	0.00	0.00	0.00	0.43
33	0.80	0.00	0.54	11.38	0.14	0.26	1.00	0.50	0.25	1.00	0.00	0.00	0.00	0.00
34	0.92	0.78	4.80	14.72	0.63	1.00	2.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00
35	0.75	0.00	0.00	3.00	0.00	0.50	1.00	0.00	0.00	0.50	0.00	0.00	0.00	0.50
36	1.00	0.00	0.02	0.03	0.15	0.00	1.00	0.17	0.00	0.83	0.00	0.00	0.00	0.17
M	0.90	0.03	2.74	7.88	0.13	0.29	1.45	0.30	0.18	0.40	0.03	0.03	0.01	0.13

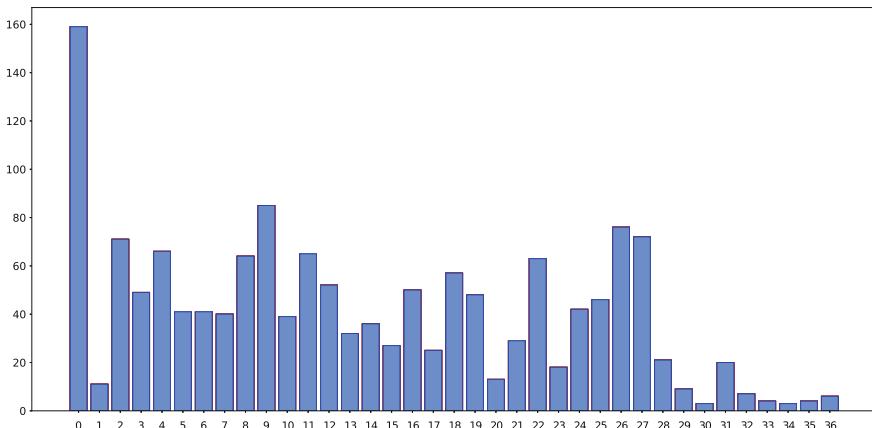


Fig. 5. Number of terrorist groups in each cluster

There are several significant results taken from this model. First of all, while Success Ratio is extremely high in each community (the average is 0.90), the community that mostly deviates from this value (i.e.: cluster 28, Success Ratio = 0.66) includes a total of 21 terror groups and none of these groups is Islamic/Jihadist.

There is only another cluster with no Islamic/Jihadist groups in it (cluster 35), and although it is a small community (only 4 groups in it), the Success Ratio is significantly low in that case too (0.75). This seems to indicate that, generally, Islamic/Jihadist groups tend to have good operational performance in their attacks. As further proof, it is worth to note that the clusters with the highest percentage of Islamic/Jihadist groups (namely, clusters 1, 16, 20, 30, 34) show Success Rate values always higher than the average, specifically in the range from 0.92 to 1.

Secondly, when focusing on the Fatalities Rate, data highlight that the clusters that yield higher values generally include several types of ideologies. Two major examples are cluster 0 (Fatalities Rate = 4.25) and Cluster 4 (4.39). In the first case, out of a total of 159 terror groups, 37.11% are Islamic/Jihadist and 35.85% are Ethnic/Nationalist. In the second case, out of a total of 66 groups, 42.42% are Islamic/Jihadist and 37.88% are Nationalists. This demonstrates how the fact of being able to carry out particularly devastating attacks is not a feature that is specifically related to a single ideology.

Thirdly, Pearson correlation (Fig. 4) revealed interesting relations between ideologies and behavioral characteristics, looking at data from a more general perspective. Listing some: multiplot rates are extremely correlated with suicide and international operations. Additionally there is no strong relationship between the degree of severity of an attack (fatality rate) and the extent to which this attack is internationally plotted or motivated. This result indicates that both domestic and transnational terrorism are able to inflict high-magnitude

terror. Our further hypothesis is that, however, the distribution of highly fatal attacks per geopolitical terror type (i.e.: domestic or transnational) is not equally distributed across regions in the world. In fact, it may be that being transnational or domestic is not a discriminant feature *per se*, but it is intrinsically related to the operating geographic context. Moreover, it is worth to note how there is a positive correlation between the shares of environmental/animalist groups and leftists/anarchist organizations: two ideological types of terrorism which are overlapping in some cases.

6 Discussion and Future Work

This work has applied a novel clustering algorithm for multi-partite networks based on Gower's coefficient of similarity to define latent communities of terror groups at a global scale, using data for the period 1997–2016. Besides the innovative application, this work has presented multiple models based on different weighting schemes and demonstrated how (1) weighting more specific features may lead to substantially different results and (2) giving more importance to ideology will ultimately hide common behavioral patterns that groups shared regardless of their motivations. Finally, this work has presented the results of the algorithm in the “No Weight” case, analyzing the most relevant outcomes in terms of attribute variables across the behaviorally-detected clusters. This exploratory application calls for future work. Specifically, further directions could involve the use of machine learning algorithms developing feature spaces adding contextual, operational, and temporal information to evaluate if it is possible to train a classifier that correctly predicts the cluster to which each group is assigned. This analysis would eventually confirm the meaningfulness of the resulting communities. Using a simple ex-post descriptive analysis, we have demonstrated that our algorithms yield outcomes that give a certain patterned structure to the data. Thus, a further investigation of the data using supervised learning may strongly corroborate our findings. Furthermore, our application seeks to be compared to other existing clustering algorithms designed for multi-mode matrices, e.g. the Infinite Relational Model [9], in order to evaluate the stability of the results when other methodological architectures are applied.

References

1. Belli, R., Freilich, J.D., Chermak, S.M., Boyd, K.A.: Exploring the crime-terror nexus in the United States: a social network analysis of a Hezbollah network involved in trade diversion. *Dyn. Asymm. Confl.* **8**(3), 263–281 (2015). <https://doi.org/10.1080/17467586.2015.1104420>
2. Benigni, M.C., Joseph, K., Carley, K.M.: Online extremism and the communities that sustain it: detecting the ISIS supporting community on Twitter. *PLOS ONE* **12**(12), e0181405 (2017). <https://doi.org/10.1371/journal.pone.0181405>
3. Blondel, V.D., Guillaume, J.L., Lambiotte, R., Lefebvre, E.: Fast unfolding of communities in large networks. *J. Stat. Mech. Theory Exp.* (10), P10,008 (2008). <https://doi.org/10.1088/1742-5468/2008/10/P10008>

4. Campedelli, G.M., Cruickshank, I., Carley, K.M.: Complex networks for terrorist target prediction. In: Thomson, R., Dancy, C., Hyder, A., Bisgin, H. (eds.) Social, Cultural, and Behavioral Modeling, vol. 10899, pp. 348–353. Springer International Publishing, Cham (2018)
5. Carley, K.M.: Destabilization of covert networks. Comput. Math. Org. Theory **12**(1), 51–66 (2006). <https://doi.org/10.1007/s10588-006-7083-y>
6. Chenoweth, E., Lowham, E.: On classifying terrorism: a potential contribution of cluster analysis for academics and policy-makers. Def. Sec. Anal. **23**(4), 345–357 (2007). <https://doi.org/10.1080/14751790701752402>
7. Desmarais, B.A., Cranmer, S.J.: Forecasting the locational dynamics of transnational terrorism: a network analytic approach. Sec. Inf. **2**(1), 8 (2013). <https://doi.org/10.1186/2190-8532-2-8>
8. Gower, J.C.: A general coefficient of similarity and some of its properties, **27**
9. Kemp, C., Tenenbaum, J., Griffiths, T., Yamada, T., Ueda, : L.: Learning systems of concepts with an infinite relational model. AAAI **3** (2006)
10. Klausen, J.: Tweeting the Jihad : social media networks of western foreign fighters in Syria and Iraq. Stud. Confl. Terror. **38**(1), 1–22 (2015). <https://doi.org/10.1080/1057610X.2014.974948>
11. Koschade, S.: A social network analysis of jemaah islamiyah: the applications to counterterrorism and intelligence. Stud. Confl. Terror. **29**(6), 559–575 (2006). <https://doi.org/10.1080/10576100600798418>
12. Krebs, V.: Mapping networks of terrorist cells. Connections **24**(3), 43–52 (2002)
13. LaFree, G., Dugan, L.: Introducing the global terrorism database. Terror. Polit. Viol. **19**(2), 181–204 (2007). <https://doi.org/10.1080/09546550701246817>
14. Lautenschlager, J., Ruvinsky, A., Warfield, I., Kettler, B.: Group profiling automation for crime and terrorism (GPACT). Proc. Manuf. **3**, 3933–3940 (2015). <https://doi.org/10.1016/j.promfg.2015.07.922>
15. von Luxburg, U.: A tutorial on spectral clustering (2007). arXiv:0711.0189 [cs], <http://arxiv.org/abs/0711.0189>
16. Medina, R., Hepner, G.: In: In: Karawan, I.A., McCormack, W., Reynolds, S.E. (eds.), Values and Violence. Geospatial Analysis of Dynamic Terrorist Networks, vol. 4, pp. 151–167. Springer, Netherlands, Dordrecht (2009). https://doi.org/10.1007/978-1-4020-8660-1_10
17. Moon, I.C., Carley, K.M.: Modeling and simulating terrorist networks in social and geospatial dimensions. IEEE Intell. Syst. **22**(5), 40–49 (2007). <https://doi.org/10.1109/MIS.2007.4338493>
18. Qi, X., Christensen, K., Duval, R., Fuller, E., Spahiu, A., Wu, Q., Zhang, C.Q.: A hierarchical algorithm for clustering extremist web pages. In: 2010 International Conference on Advances in Social Networks Analysis and Mining, pp. 458–463 (2010). <https://doi.org/10.1109/ASONAM.2010.81>
19. Ruan, J.: A fully automated method for discovering community structures in high dimensional data, pp. 968–973. IEEE (2009). <https://doi.org/10.1109/ICDM.2009.141>
20. Sageman, M.: The stagnation in terrorism research. Terror. Polit. Viol. **26**(4), 565–580 (2014). <http://dx.doi.org/10.1080/09546553.2014.895649>



GLaSS: Semi-supervised Graph Labelling with Markov Random Walks to Absorption

Max Glonek^{1(✉)}, Jonathan Tuke¹, Lewis Mitchell¹, and Nigel Bean^{1,2}

¹ School of Mathematical Sciences, University of Adelaide, Adelaide, SA 5005, Australia

max.glonek@adelaide.edu.au

² ARC Centre of Excellence for Mathematical and Statistical Frontiers, University of Adelaide, Adelaide, SA 5005, Australia

Abstract. Graph labelling is a key activity of network science, with broad practical applications, and close relations to other network science tasks, such as community detection and clustering. While a large body of work exists on both unsupervised and supervised labelling algorithms, the class of random walk-based supervised algorithms requires further exploration, particularly given their relevance to social and political networks. This work proposes a new semi-supervised graph labelling method, the GLaSS method, that exactly calculates absorption probabilities for random walks on connected graphs, whereas previous methods rely on simulation and approximation. The proposed method models graphs exactly as a discrete time Markov chain, treating labelled nodes as absorbing states. The method is applied to a series of undirected graphs of roll call voting data from the United States House of Representatives. The GLaSS method is compared to existing supervised and unsupervised methods, demonstrating strong and consistent performance when estimating the labels of unlabelled nodes in graphs.

Keywords: Community detection · Graph labelling · Random walk · Markov chain · Political networks

1 Introduction

Graph labelling is concerned with the problem of estimating the labels of one or more nodes within a graph, where an association between the graph's structure and the distribution of labels is assumed to exist. Many graph labelling algorithms exist, both supervised [2, 7, 13] and unsupervised [10, 14]. In both cases, a graph comprises u unlabelled and ℓ labelled nodes, and the algorithms seek to estimate the labels of the unlabelled nodes. While a diverse range of graph labelling methods exist [4], this work focuses on the class of dynamical and statistical inference methods that use random walks.

In unsupervised algorithms, the graph is organised into clusters, without consideration of the labelled nodes. Once clustered, labels for unlabelled nodes in the graph can be estimated based on the clusters to which labelled nodes belong. However, cases may arise where an identified cluster contains no labelled nodes, or where a cluster contains multiple nodes with different labels, creating uncertainty as to how labels should be estimated for nodes in such clusters.

The Walktrap algorithm is one commonly used random walk-based unsupervised graph labelling method [10]. Walktrap searches for densely connected subgraphs by simulating short random walks on a graph, reasoning that short walks are more likely to remain in the same cluster than to leave it. Walktrap quantifies the similarity between nodes using a distance metric, then recursively merges identified clusters based on short random walks, providing a hard classification for each node. Because Walktrap does not use information about labelled nodes, there is no generally accepted method for estimating the labels for unlabelled nodes based on the clusters it identifies.

Unlike unsupervised algorithms, supervised algorithms utilise the information contained in labelled nodes when estimating the labels of unlabelled nodes. A common approach is to treat labelled nodes as absorbing states and unlabelled nodes as transient states in a discrete time Markov chain (DTMC), and estimate the absorption probabilities or expected times to absorption for all transient states in the chain. Labels for each unlabelled state can then be estimated using the approximate probabilities or times. However, while supervised methods use both labelled nodes and the graph's structure to estimate labels, they only approximate absorption probabilities and times, rather than calculating them exactly.

The Rendezvous algorithm [2] labels nodes in a semi-supervised setting by constructing a simplified, “rendezvous” graph, where edges are drawn from an unlabelled node to only its M nearest neighbours. M is chosen to be as small as possible while ensuring that each unlabelled node in the rendezvous graph is connected to at least one labelled node. Once the rendezvous graph has been constructed, edge weights are calculated using a Euclidean distance metric, and absorption probabilities are calculated using the eigenvalues and eigenvectors of the rendezvous graph's transition matrix. Absorption probabilities for nodes in the rendezvous graph are then used to estimate the label of nodes in the full graph.

Another semi-supervised graph labelling method seeks to label nodes in a binary setting according to expected time to absorption, rather than absorption probability [7]. This “Censored Time” method simulates step-limited random walks over a graph, recording the number of steps taken for all walks that are absorbed before being terminated by the step limit. The censored times to absorption for absorbed walks are used to approximate the conditional expected time to absorption in each labelled node in the graph. A hard classification is used to estimate labels according to the lowest censored conditional time to absorption.

This work proposes a new semi-supervised graph labelling method, the Graph Labelling Semi-Supervised (GLaSS) method, using random walks to absorption. The method models a graph as a DTMC, where transient states correspond to unlabelled nodes, and absorbing states correspond to labelled nodes. The transition matrix P , for the DTMC, is formed from the graph's weighted adjacency matrix by normalising the weighted out-degree of each node in the network. From careful construction of P , the probability of absorption in each absorbing state can be calculated exactly, and these probabilities can then be used to estimate the label for every node corresponding to a transient state in the DTMC.

By calculating exact absorption probabilities and expected times to absorption, the GLaSS method provides better label estimates than contemporary supervised methods, which rely on approximations of these quantities. By utilising the information contained in labelled nodes in the graph, GLaSS also provides a clear method for estimating the label of unlabelled nodes using quantities that are meaningful and interpretable, unlike unsupervised random walk methods.

The GLaSS method is formally introduced in Sect. 2. Section 3 describes the data analysed, and a full description of all analyses performed is presented in Sect. 4. Conclusions and areas for further work are discussed in Sect. 5.

2 Method

Consider an undirected graph $G = (V, E)$ comprising n nodes, $V = \{v_1, \dots, v_n\}$, connected by a set of positive real-weighted edges E . Define the weighed adjacency matrix $A = [a_{i,j}]$, where $a_{i,j} = a_{j,i}$ records the weight of the edge connecting v_i and v_j , and $a_{i,j} = 0$ if no edge connects v_i and v_j . Suppose the first u nodes in G are unlabelled, and the remaining ℓ nodes in G are labelled, where $n = u + \ell$, and construct the sets $U = \{1, \dots, u\}$ and $L = \{u+1, \dots, n\}$ to index the unlabelled and labelled nodes of G , respectively. Arrange A as

$$A = \begin{bmatrix} A_{U,U} & A_{U,L} \\ A_{L,U} & A_{L,L} \end{bmatrix}$$

where $A_{J,K}$ describes the weighted edges connecting nodes indexed by J to nodes indexed by K .

Consider a random walk on G , described by a discrete time Markov chain (DTMC) where all unlabelled nodes map to transient states and all labelled nodes map to absorbing states. Let X_t denote the state of the chain at time t . Calculate the transition probabilities for the DTMC using A , where

$$p_{i,j} = P(X_{t+1} = j \mid X_t = i) = \frac{a_{i,j}}{\sum_{k=1}^n a_{i,k}} \quad (1)$$

is the probability that the DTMC is in state j at the next time step, given that the DTMC is currently in state i . Construct the transition matrix

$$P = [p_{i,j}] = \begin{bmatrix} P_{U,U} & P_{U,L} \\ P_{L,U} & P_{L,L} \end{bmatrix} = \begin{bmatrix} R & S \\ 0 & I_\ell \end{bmatrix}. \quad (2)$$

The $u \times u$ matrix R governs transitions between transient states, the $u \times \ell$ matrix S governs transitions from transient states to absorbing states, 0 is an $\ell \times u$ zero matrix, and I_ℓ is the $\ell \times \ell$ identity matrix.

2.1 DTMC Absorption Probabilities

Let $h_{i,j}$ be the probability that the DTMC is eventually absorbed in state j , given that the chain starts in state i . Define the matrix of absorption probabilities $H = [h_{i,j}]$. H is restricted to have u rows and ℓ columns, corresponding to the u transient states and ℓ absorbing states of the DTMC, respectively. Then H can be calculated as

$$H = (I_u - R)^{-1} S \quad (3)$$

where I_u is the $u \times u$ identity matrix, and R and S are as above [6].

2.2 Semi-supervised Graph Labelling

Given a graph G and the matrix of absorption probabilities H , let Y_i be the label of an unlabelled node v_i , and let y_j be the label of a labelled node v_j . The distribution over Y_i can be directly derived from H , for all $i \in U$, as follows:

$$P(Y_i = k) = \sum_{j=u+1}^n h_{i,j} \mathbb{1}(y_j = k) \quad (4)$$

where $\mathbb{1}$ is an indicator function, taking value 1 if its argument is true, and 0 otherwise.

2.3 DTMC Expected Times to Absorption

Let t_i be the expected number of time steps before the DTMC is absorbed in any absorbing state, given that the chain starts in state i . Define the vector of expected times to absorption $\mathbf{t} = (t_1, \dots, t_u)^T$, where the u elements of \mathbf{t} correspond to the u transient states of the DTMC. Then \mathbf{t} can be calculated as

$$\mathbf{t} = (I_u - R)^{-1} \mathbf{c} \quad (5)$$

where \mathbf{c} is a column vector of length u whose entries are all 1, and I_u and R are as above [6].

2.4 The Graph Labelling Semi-supervised (GLaSS) Method

Consider a graph G , with u unlabelled nodes and ℓ labelled nodes, and suppose that all labelled nodes have one of two labels; either K_1 or K_2 . From the weighted adjacency matrix A , construct the transition matrix P , as in (1). Using P , calculate the vector of expected times to absorption \mathbf{t} , as in (5). The expected times to absorption may, optionally, be used as a filtering criterion; nodes with

a large expected time to absorption, relative to the distribution of t_i over all nodes in the graph, may be excluded from further analysis.

Once nodes have been optionally filtered using \mathbf{t} , calculate the matrix of absorption probabilities H , by (3), and calculate $P(Y_i = K_1)$ and $P(Y_i = K_2)$ for all $i \in U$, as in (4). Because, by the Law of Total Probability, $P(Y_i = K_1) + P(Y_i = K_2) = 1$, only one probability is required to proceed. Consider $P(Y_i = K_1)$ for all i , and implement a binary classifier with some threshold α . If $P(Y_i = K_1) \geq \alpha$, estimate the label for node v_i as K_1 ; otherwise, if $P(Y_i = K_1) < \alpha$, estimate the label for node v_i as K_2 . Choose α to maximise the binary classifier's discrimination between K_1 and K_2 .

Using this method, it is possible to estimate the label for every unlabelled node in G . As a graph labelling method in a semi-supervised setting, the method is called the GLaSS method.

3 Data

Validating the GLaSS method requires graphs with a clear community structure and known labels for all nodes. To simulate a graph with few known labels, only a small subset of all known labels will be used by GLaSS, with remaining labels withheld to simulate “unlabelled” nodes in the graph. All labels estimated by GLaSS can be compared to actual, withheld labels, to assess performance. Therefore, United States roll call voting data is used to validate the GLaSS method.

In the United States House of Representatives (the House), parliamentary procedure occasionally gives rise to roll call votes. In a roll call vote, the vote of every member of the House is recorded, making it possible to see which members of the House voted the same way. Roll call voting data can be modelled as an undirected graph, where each node represents a member of the House, and a positive integer-weighted edge records the number of times respective members voted the same way.

The results of roll call votes in the House for the meetings of eight separate Congresses, between 1953 and 1997¹, have been collected for analysis [9], and modelled as eight separate undirected graphs. For simplicity, in each Congress, the following rules are applied:

1. Only “yea” and “nay” votes are considered.
2. Votes are disregarded if cast by the Speaker of the House^{2,3}.
3. Only members whose party affiliation is Democrat or Republican are considered.
4. In cases where a member’s party affiliation changes during a meeting of Congress, their party affiliation at the time they were elected is used.

¹ Each meeting of Congress begins on January 3 and runs for a period of two years.

² Conventionally, the Speaker of the House participates in very few votes.

³ The 101st Congress had two speakers, both of whose votes are disregarded in these analyses.

5. In rare cases, a member of the House does not sit for the entire meeting of Congress, and their seat is taken by a new member. In these cases, the voting records of both members are retained.⁴

Because the party affiliation of each member is known, all nodes in each graph are labelled. For random walks on each graph, only the labels of nodes corresponding to the Majority Leader and the Minority Leader are retained (one Democrat and one Republican), thus all other nodes in each graph are “unlabelled”. Choice of Congresses is informed by recent work examining partisanship trends in the House [1], ensuring variation in partisanship and which party is in Majority. All graphs are either fully connected or nearly fully connected, and a detailed summary of each graph is contained in Table 1.

Table 1. Years covered, total number of members (nodes), democrats, republicans, and votes for each congress. Congresses where the number of democrats is shown in **bold** had a democrat majority leader, and congresses where the number of republicans is shown in **bold** had a republican majority leader.

Congress	Years	Total members	Democrats	Republicans	Votes
83rd	1953–55	439	218	221	147
86th	1959–61	441	285	156	180
89th	1965–67	441	299	142	394
92nd	1971–73	442	257	185	649
95th	1977–79	440	293	147	1540
98th	1983–85	438	271	167	906
101st	1989–91	441	262	179	879
104th	1995–97	443	208	235	1321

4 Results

Each Congress is modelled as a graph, and each graph is analysed using the GLaSS method, as described in Sect. 2.4. Expected time to absorption is calculated for each “unlabelled” node in each graph; the mean and variance of t_i for each graph are given in Table 2. Based on the distribution of t_i for each graph, no filtering is required, and labels are estimated for all “unlabelled” nodes in each Congress.

As each graph contains only two labelled nodes (one Democrat, one Republican), only the probability of being absorbed in the Democrat state of the corresponding DTMC is considered. Histograms of absorption probabilities for the 83rd, 86th, 89th, and 92nd Congresses are shown in Fig. 1, and histograms for the 95th, 98th, 101st, and 104th Congresses are shown in Fig. 2. In all Congresses, Democrat and Republican members are clearly separated, though some overlap between clusters exists.

⁴ Consequently, while the House has 435 seats, each graph has more than 435 nodes.

Using the binary classifier in GLaSS, a threshold α_k is chosen for the k th Congress. If $P(Y_i = Democrat) \geq \alpha_k$, then member i is labelled a Democrat; otherwise, member i is labelled a Republican. Estimated labels are compared to the true party affiliation for all “unlabelled” nodes. By varying α_k across the range of absorption probabilities calculated for each respective Congress, a ROC curve is derived. ROC curves for all eight Congresses are displayed in Fig. 3, and the AUC for each Congress is given in Table 2.

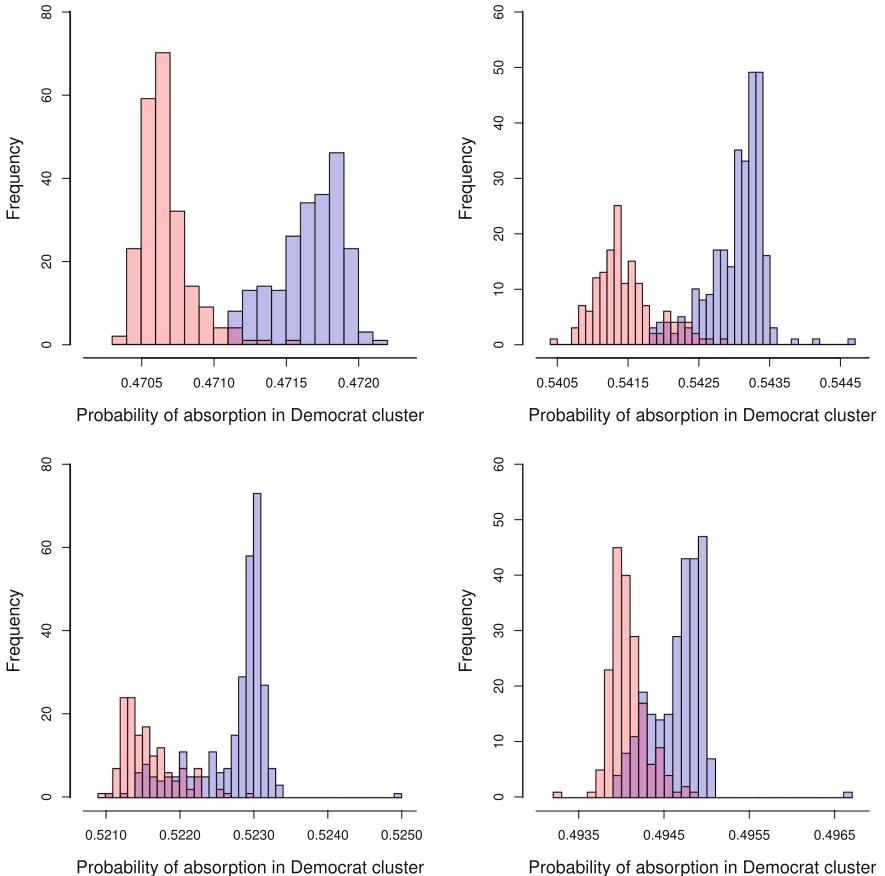


Fig. 1. Clockwise from top right: 86th, 92nd, 89th, and 83rd congresses. Histograms show the probability of absorption in the democrat cluster for each congress. Red bars show republican members, and blue bars show democrat members. The range of absorption probabilities for each congress is narrow, but clusters are clearly separated. Note there is more overlap between clusters in the 89th and 92nd congresses, corresponding to increased bipartisanship [1].

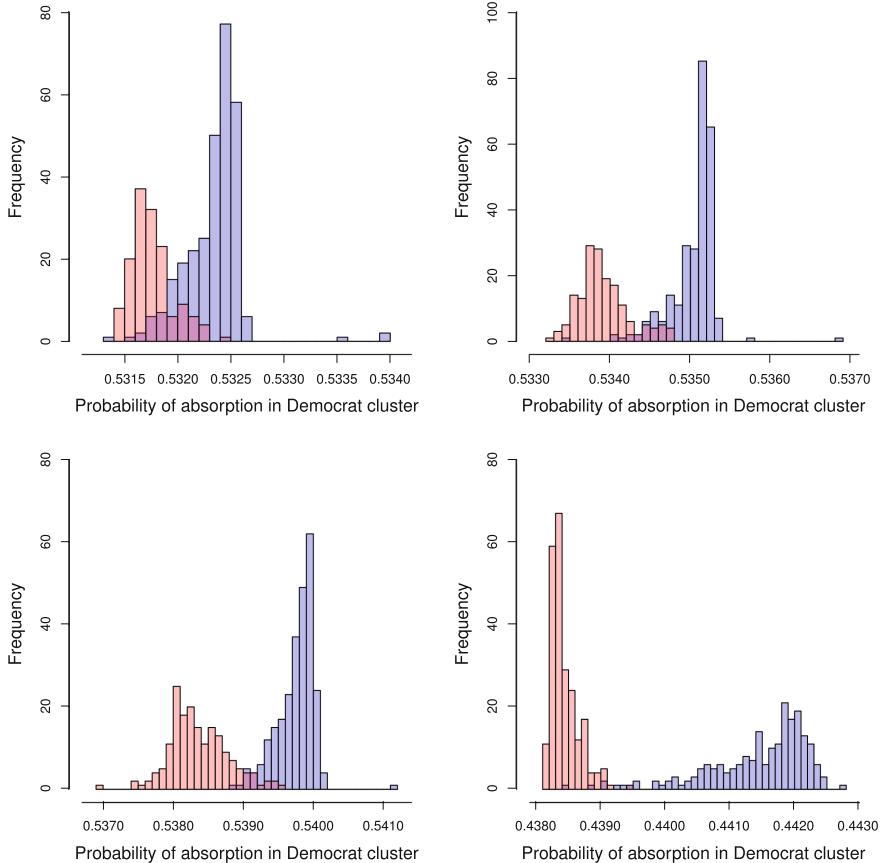


Fig. 2. Clockwise from top right: 98th, 104th, 101st, and 95th congresses. Histograms show the probability of absorption in the democrat cluster for each congress. Red bars show republican members, and blue bars show democrat members. The range of absorption probabilities for each congress is narrow, but clusters are clearly separated. Clusters become more separated over time, corresponding to an increase in partisanship within the house [1].

4.1 Comparison to Other Methods

The GLaSS method is compared to two alternative random walk-based graph labelling methods. The first method, the Walktrap algorithm [10], is an unsupervised method. Walktrap searches for densely connected subgraphs by simulating random walks on a graph, reasoning that short random walks are more likely to stay in the same cluster than to leave it. Because each Congress has two clearly defined clusters (Democrats and Republicans), the Walktrap algorithm is successful, in the first instance, if it places the Majority Leader and Minority Leader in different clusters, and if only two clusters are identified. If the Walktrap algorithm is successful in separating the Majority and Minority Leaders, the label for

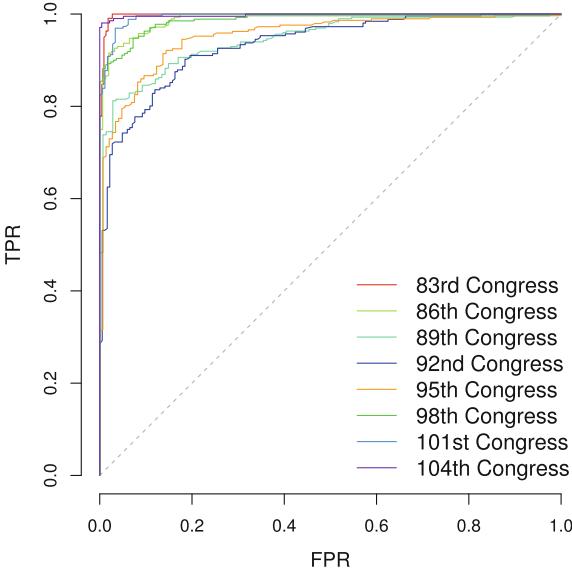


Fig. 3. ROC curves showing the performance of the binary classifier in GLaSS for each Congress, as α_k is varied. As expected, the three most partisan congresses analysed (89th, 92nd, and 95th) [1] show the weakest performance.

each member is estimated to be the same as the label of the Leader in that member’s cluster. All analysis is conducted using a popular default implementation of the Walktrap algorithm [8].

The second method (Censored Time) is semi-supervised, and estimates the expected time to absorption, conditional on being absorbed in each labelled state [7]. Censored Time simulates step-limited random walks on a graph, where a walk is terminated if it is not absorbed before reaching the step limit. For walks that are absorbed, the censored conditional time to absorption is recorded, and these are used to estimate the conditional expected time to absorption for each labelled state. For a graph with two labels, Censored Time labels nodes according to the state with the smaller estimated conditional expected time to absorption. For each graph, the exact expected time to absorption is calculated for all nodes, as specified in Sect. 2.3, and the ceiling of the mean expected time to absorption is adopted as the step limit for Censored Time. For each “unlabelled” state in each graph, 1000 step-limited random walks are simulated, to estimate the conditional expected time to absorption for each labelled state.

To compare the performance of Walktrap, Censored Time, and GLaSS, an F1 score is calculated for each method and each Congress. For each Congress, the value of α_k is chosen to maximise GLaSS’s F1 score. F1 scores for all three methods and all eight Congresses are given in Table 2. From the F1 scores, it is clear that GLaSS outperforms Censored Time for all Congresses, and equals or surpasses Walktrap in most cases. Walktrap provides comparable performance to

GLaSS for the most partisan Congresses (101st and 104th), but its performance decreases with decreasing partisanship, and it fails for two Congresses (83rd and 89th), by identifying more than two clusters. The GLaSS method exceeds, or effectively matches, the performance of Walktrap and Censored Time for all Congresses, while also showing greater resilience to decreasing separation of clusters caused by decreases in partisanship [1].

Table 2. F1 scores for walktrap and censored time, and the maximal F1 score for GLaSS (highest scores shown in **bold**). Additionally, α_k gives the range of cutoffs that yield the maximal F1 score using GLaSS. AUC gives the area under the curve for the ROC curves for GLaSS. The mean and variance of the expected time to absorption (see Sect. 2.3) for each Congress are given in μ_t and σ_t , respectively.

Congress	F1 score			GLaSS			
	Walktrap	Censored time	GLaSS	α_k	AUC	μ_t	σ_t
83rd	- ^a	0.5068	0.9864	(0.471117, 0.471149)	0.9978	191.77	0.0037
86th	0.8479	0.5779	0.9561	(0.542170, 0.542193)	0.9899	203.47	0.0062
89th	- ^a	0.5957	0.9122	(0.521823, 0.521827)	0.9464	214.97	0.0014
92nd	0.7975	0.5259	0.8876	(0.494203, 0.494204)	0.9338	208.33	0.0046
95th	0.8333	0.5558	0.9293	(0.531896, 0.531899)	0.9528	218.68	0.0014
98th	0.8907	0.5602	0.9531	(0.534291, 0.534311)	0.9857	216.22	0.0044
101st	0.9738	0.5980	0.9736	(0.539083, 0.539084)	0.9949	223.42	0.0007
104th	0.9878	0.5667	0.9878	(0.439130, 0.439215)	0.9979	223.20	0.0030

^a More than two clusters identified

5 Discussion

Graph labelling is a fundamental task within network science, with diverse applications. This work proposes a new semi-supervised graph labelling method, the GLaSS method, using random walks to absorption. The GLaSS method has been used to analyse a series of undirected graphs, showing very strong performance when estimating the labels of unlabelled nodes. The GLaSS method represents a compelling alternative to existing supervised and unsupervised random walk methods. The key features of the GLaSS method are that, unlike other supervised methods, it calculates exact absorption probabilities and expected times to absorption, and, unlike unsupervised methods, it provides a clear method for the labelling of unlabelled nodes based on identified clusters.

Results show the GLaSS method meets or exceeds the performance of the supervised and unsupervised methods to which it is compared, as measured using F1 score. ROC curves and AUC for each graph analysed also show that the GLaSS method shows consistently very strong performance. Future work will

extend this work to examine the performance of the GLaSS method for graphs of varying size, connectedness, density, and with different numbers of known labels. Extending the GLaSS method can be generalised to label graphs with more than two clusters, and graphs with fewer labelled nodes than clusters, is of particular interest. Future work will also explore the use of expected time to absorption as a filtering criterion for nodes, particularly in cases where the number of clusters exceeds the number of known labels.

In an applied setting, future work will also use GLaSS to further explore social, political, and other networks. Online and social-media networks are of particular interest, with a growing body of work examining the structure, dynamics, and polarisation of online social networks [3,5,11,12]. Future applied work with GLaSS will examine these characteristics for new and existing graphs.

Acknowledgements. The authors thank Data to Decisions CRC and the ARC Centre of Excellence for Mathematical and Statistical Frontiers for their financial support.

References

1. Andris, C., Lee, D., Hamilton, M.J., Martino, M., Gunning, C.E., Selden, J.A.: The rise of partisanship and super-cooperators in the US house of representatives. *PLoS One* **10**(4), e0123507 (2015)
2. Azran, A.: The rendezvous algorithm: multiclass semi-supervised learning with markov random walks. In: Proceedings of the 24th International Conference on Machine Learning (ICML), pp. 49–56 (2007)
3. Fish, B., Huang, Y., Reyzin, L.: Recovering social networks by observing votes. In: Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems, pp. 376–384 (2016)
4. Fortunato, S.: Community detection in graphs. *Phys. Rep.* **486**(3–5), 75–174 (2010)
5. Garimella, K., Weber, I.: A long-term analysis of polarization on Twitter. [arXiv:1703.02769](https://arxiv.org/abs/1703.02769) (2017)
6. Grinstead, C.M., Snell, J.L.: Introduction to probability. Amer. Math. Soc. (2012)
7. Hassan, A., Radev, D.: Identifying text polarity using random walks. In: Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL), pp. 395–403 (2010)
8. Csardi, G., Nepusz, T.: The igraph software package for complex network research. *InterJ. Complex Syst.* 1695 (2006). <https://igraph.org>. Accessed 28 Aug 2018
9. Lewis, J.B., Poole, K., Rosenthal, H., Boche, A., Rudkin, A., Sonnet, L.: Voteview: congressional roll-call votes database (2018). <https://voteview.com/data>. Accessed 21 Aug 2018
10. Pons, P., Latapy, M.: Computing communities in large networks using random walks. In: International Symposium on Computer and Information Sciences, pp. 284–293 (2005)
11. Rizouli, M.A., Graham, T., Zhang, R., Zhang, Y., Ackland, R., Xie, L.: #debatenight: the role and influence of socialbots on twitter during the 1st us presidential debate. [arXiv:1802.09808](https://arxiv.org/abs/1802.09808) (2018)
12. Shai, S., Stanley, N., Granell, C., Taylor, D., Mucha, P.J.: Case studies in network community detection. [arXiv:1705.02305](https://arxiv.org/abs/1705.02305) (2017)

13. Talukdar, P.P., Reisinger, J., Paşa, M., Ravichandran, D., Bhagat, R., Pereira, F.: Weakly-supervised acquisition of labelled class instances using graph random walks. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing, pp. 582–590 (2008)
14. Zhou, H., Lipkowsky, R.: Network brownian motion: A new method to measure vertex-vertex proximity and to identify communities and subcommunities. In: International Conference on Computational Science (ICCS), pp. 1062–1069 (2004)



Semi-supervised Overlapping Community Finding Based on Label Propagation with Pairwise Constraints

Elham Alghamdi[✉] and Derek Greene

University College Dublin, Dublin, Ireland

elham.alghamdi@ucdconnect.ie, derek.greene@ucd.ie

Abstract. Algorithms for detecting communities in complex networks are generally unsupervised, relying solely on the structure of the network. However, these methods can often fail to uncover meaningful groupings that reflect the underlying communities in the data, particularly when those structures are highly overlapping. One way to improve the usefulness of these algorithms is by incorporating additional background information, which can be used as a source of constraints to direct the community detection process. In this work, we explore the potential of semi-supervised strategies to improve algorithms for finding overlapping communities in networks. Specifically, we propose a new method, based on label propagation, for finding communities using a limited number of pairwise constraints. Evaluations on synthetic and real-world datasets demonstrate the potential of this approach for uncovering meaningful community structures in cases where each node can potentially belong to more than one community.

Keywords: Overlapping community finding · Semi-supervised learning

1 Introduction

In many real-world application involving machine learning, the tasks do not neatly correspond to the standard distinction between supervised and unsupervised learning. Rather, a limited degree of background knowledge or user annotation time will be available. Tasks such as community detection can potentially benefit from the introduction of “lightweight” supervision originating from domain experts or crowdsourced annotations, where this knowledge might be encoded as constraints indicating that a pair of nodes should always be assigned to the same community or should never be assigned to the same community. For instance, we might be interested in grouping users on a social media platform such as Twitter, based primarily on their follower connections, in order to discover communities of individuals with shared ideologies. To improve our ability to achieve this, and go beyond simply looking at connections, we could present pairs of user profiles to a human annotator (the “oracle”), to ask whether two users should be assigned to the same community or different communities. By

harnessing this kind of knowledge, we can potentially uncover communities of nodes which are difficult to identify with unsupervised methods.

Initial work in community detection focused on the development of algorithms to produce disjoint groups [3]. However, in many real-world networks we observe pervasive overlap, where nodes belong to many highly-overlapping groups [1]. More recently, overlapping community finding algorithms have been developed for application to these networks [1, 14]. However, this work has focused only on the unsupervised case. In contrast, work on semi-supervised community finding continues to focus on cases where communities are strictly required to be disjoint [17].

In this paper, we propose a semi-supervised method for overlapping community finding based on a label propagation strategy, which has previously been applied in a purely unsupervised context [21]. The proposed method, referred to as Pairwise Constrained SLPA (PC-SLPA), involves a speaker-listener information propagation process. To encode external supervision, we use pairwise constraints to influence the community finding process. Since the choice of constraints in semi-supervised learning has been shown to be highly important [15], we further propose a strategy for selecting constraint pairs for which an oracle should be queried. This strategy is specifically designed for the case where communities overlap in a network. The experiments described later in Sect. 4, which involve synthetic and real networks, show that the introduction of a relatively small number of constraints with PC-SLPA can improve our ability to correctly uncover the underlying communities.

2 Related Work

2.1 Community Finding

Finding non-overlapping communities. Algorithms in this context can be broadly grouped into three types. (1) *Hierarchical algorithms* construct a tree of communities based on the network topology. These can be one of two types: divisive algorithms [7] or agglomerative algorithms [4]. (2) *Modularity-based algorithms* optimize the well-known modularity objective function to uncover communities in a network [20]. (3) *Other algorithms* which include those based on label propagation approaches [21], spectral methods that make use of the eigenvectors of a graph’s adjacency matrix, and methods based on statistical modeling [6].

Finding overlapping communities. Existing algorithms in this context can be classified into four main categories. (1) *Node seeding and local expansion algorithms* detect communities by starting from a node or a small group of nodes, then expanding them into a community using some fitness function. OSLOM [13] is an example of such an algorithm, which expands communities based on a fitness function measuring the statistical significance of communities with respect to random variations. (2) *Clique expansion methods* use a group of fully-connected nodes, called a clique, as the starting point for building larger

communities. Greedy Clique Expansion (GCE) [14] is an example of this type of algorithm. (3) *Link clustering algorithms* detect communities by splitting the network edges rather than the nodes [2]. (4) *Label propagation algorithms* attempt to group each node into a community based on its neighboring nodes' affinities.

Speaker-listener label propagation. A representative example of this strategy is the Speaker Label Propagation Algorithm (SLPA) [21]. Here every node is associated with a corresponding *memory* to store the frequencies of labels received from other nodes. Each node can take the role of either a *listener* or a *speaker*, and the roles are switched based on the state of the node – *i.e.* whether a node is providing information or consuming it. In the listener state, a node accepts labels from its neighbors, based on certain rules. In the speaker state, the node chooses a label from its own memory according to certain rules and sends it to neighboring listener nodes. Initially each node is assigned its own unique label. Then an iterative evaluation stage is repeatedly applied:

1. Randomly select one node as a listener.
2. Each neighbor of the listener randomly chooses a label from its own memory with a probability proportional to the frequency of occurrence of this label, and sends the label to the listener.
3. The listener chooses the most popular label among the received labels, and then adds it to its own memory.

A subsequent post-processing stage converts each node's memory into a probability distribution of labels. If the probability of the frequency of a certain label is less than a user-specified threshold, the label is removed from a node's memory. After this thresholding step, all nodes having the same label are grouped into one community. Nodes that have more than one label naturally belong to multiple communities

2.2 Semi-supervised Learning in Community Finding

Several forms of prior knowledge have been used to guide community detection. The most widely-used strategy has been that of *pairwise constraints* involving “must-link” and “cannot-link” relations. These relations indicate that either two nodes must be in the same community or must be in different communities. Such constraints have been implemented in several algorithms, including a modularity-based method [17], a spectral analysis method [9, 23], and methods based on matrix factorization [23]. Instead of constraints, some authors have proposed the use of *node labels* to encode prior knowledge for community detection [15]. In [18], the authors propose a method that uses a semi-supervised label propagation algorithm based on *node labels* and *negative information*, where a node is deemed to not belong to a specific community.

The vast majority of semi-supervised algorithms in this area aim solely at detecting disjoint communities, whereas many real-world social networks contain overlapping structures [1]. In [5], a small set of nodes called *seed nodes* was used,

whose affinities to a community is provided as prior knowledge to infer the rest of the nodes affinities in the network. However, to the best of our knowledge, no work has been done in the context of finding overlapping communities using supervision encoded as pairwise constraints.

3 Methods

3.1 Pairwise Constraints for Overlapping Communities

Before describing the proposed methods for semi-supervised community finding, we firstly discuss the issue of selecting appropriate pairwise constraints for networks containing overlapping communities.

Given a network that contains a set of nodes V , semi-supervised pairwise constraints typically take two possible forms:

1. A *must-link constraint* specifies that two nodes should be in the same community. Let C_{ML} be the must-link constraint set: $\forall v_i, v_j \in V$ where $i \neq j$, $(v_i, v_j) \in C_{ML}$ indicates that two nodes v_i and v_j must be assigned to the same community.
2. A *cannot-link constraint* specifies that two nodes should be in different communities. Let C_{CL} be the cannot-link constraint set: $\forall v_i, v_j \in V$ where $i \neq j$, $(v_i, v_j) \in C_{CL}$ indicates that v_i and v_j must be assigned to separate communities.

These constraints are provided by the oracle, typically an individual expert or committee of annotators. The simplest approach for selecting pairwise constraints to present to the oracle is to naïvely select a pair of nodes (v_i, v_j) at random, and query the oracle about whether the pair share a must-link or cannot-link relationship. This process is typically repeated until some supervision budget is exhausted.

In non-overlapping community finding, must-link constraints have a *transitive property*, such that a third must-link relationship can be inferred from two other associated must-link constraint pairs. So, if $(v_i, v_j) \in C_{ML}$, and $(v_i, v_k) \in C_{ML}$, then we can also infer that $(v_j, v_k) \in C_{ML}$ (see Fig. 1(a)).

However, incorporating constraints into the context of overlapping communities is more challenging. This is because the transitive property does not hold here (see the second example in Fig. 1). Specifically, if $(v_i, v_j) \in C_{ML}$, and $(v_i, v_k) \in C_{ML}$, there are two possible scenarios for the pair (v_j, v_k) . It can be the case that either $(v_j, v_k) \in C_{ML}$ or $(v_j, v_k) \in C_{CL}$. This is because an overlapping node v_j can have a must-link constraint with both v_i and v_k , yet these two nodes could belong to two different communities. However, it is also possible that all three nodes are in fact in the same community. Unless we explicitly inform the algorithm about whether a must-link or cannot-link constraint exists for the pair (v_j, v_k) , the algorithm cannot reliably distinguish between the two cases.

If the network has highly-overlapping communities (*i.e.* each node typically belongs to many communities), then this problematic situation will occur more

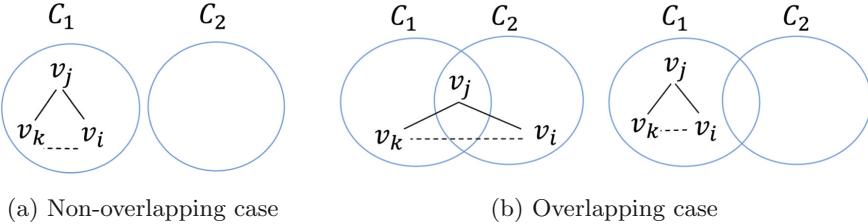


Fig. 1. In the non-overlapping case (a), the transitive property allows us to infer a third must-link constraint from two existing must-link constraints. However, this does not automatically apply in the overlapping case (b), where two possible situations exist.

frequently. Therefore, if we attempt to incorporate pairwise constraints into overlapping community finding without taking this situation into account, the quality of the resulting communities can potentially decrease, even as more constraints are added. Next we introduce a strategy to resolve this issue.

3.2 Semi-supervised Overlapping Community Finding

We now propose a new semi-supervised label propagation procedure for finding overlapping communities, which consists of two distinct phases:

1. Select and pre-process constraints, to resolve the problem of the lack of the transitive property for must-link constraints.
2. Apply label propagation-based community finding, in a manner that takes into account information provided by the selected constraints.

Phase 1: Selecting and pre-processing constraints. After selecting an initial set of pairwise constraints by querying an oracle, we can view the set of pairwise constraints as a new graph, where an edge exists between two nodes from the original network if they share a pairwise constraint (either must-link or cannot-link). Then we look for all possible *forbidden triads* among the nodes involved in the must-link set. Given three nodes A, B, C, a *forbidden triad* (sometimes referred to as an *open triad*) occurs when A is connected to B and C, but no edge exists between B and C. In our pre-processing step, we look for such cases — *i.e.* where we do not know whether a must-link or cannot-link exists between a pair of nodes B and C. To control the size of the constraints set, we greedily expand it until we reach a pre-defined maximum size. The complete constraint selection strategy can be summarized as follows (see also Fig. 2):

1. Select a small random set of both must-link and cannot-link constraints.
2. Find all possible forbidden triads in the must-link set, to identify pairs to query the oracle about their relationship.
3. For each resulting pair, if their relationship is must-link, then add the pair to the must-link set. Otherwise, add the pair to the cannot-link set.
4. Repeat all steps until the maximum number of selected constraints is reached.

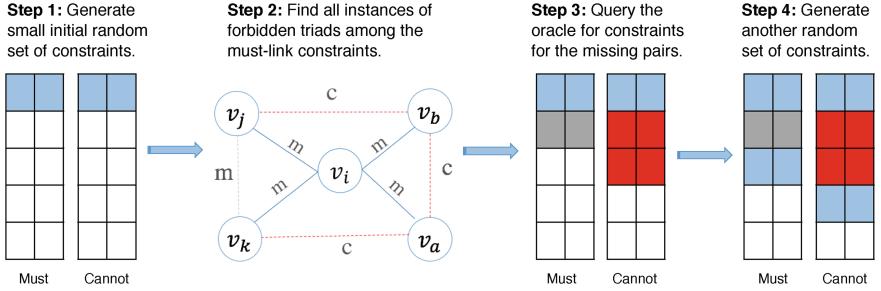


Fig. 2. An illustration of all steps in the overlapping constraint selection process.

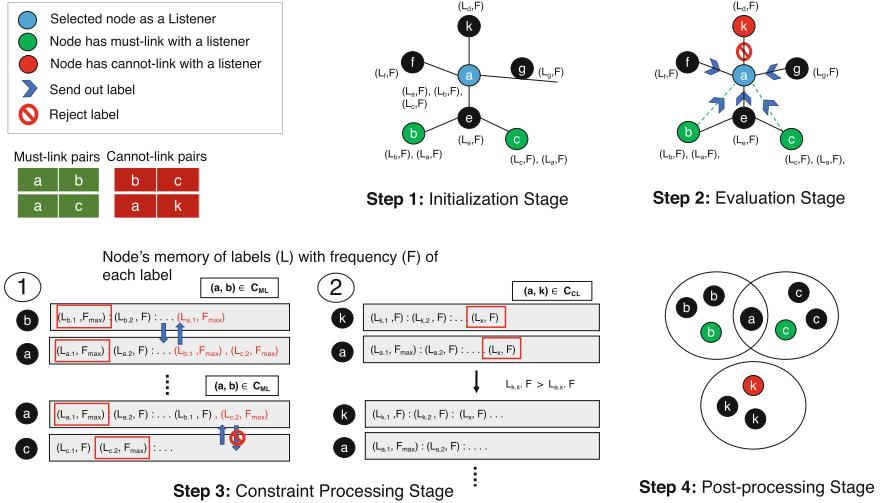


Fig. 3. An illustration of Steps 1–4 involved in the PC-SLPA algorithm.

At the end of this process, the pairwise constraints are ready to be supplied to the community detection algorithm, which we describe next.

Phase 2: Pairwise Constrained SLPA (PC-SLPA). We incorporate the selected pairwise constraints as follows (see also Fig. 3):

1. In the *initialization step*:
 - Give a unique label to each node in the network.
 - For each pair of nodes having a must-link relationship, the two nodes exchange labels (*i.e.* update each node's memory with the other node's label).
2. The *evaluation step* broadly follows a similar process as unsupervised SLPA (see Sect. 2.1). However, we account for the pairwise constraints as follows:
 - Randomly select one node as a listener, and identify the set of speakers (*i.e.* the neighbours of the listener).
 - Augment the set of speakers by adding all nodes that have must-link relationship with the listener and removing all nodes that have cannot

- link relationship with the listener. Then each speaker sends out a label according to the rule defined in standard SLPA.
- (c) Each listener accepts the sent labels, unless sent by a node which shares a cannot-link constraint with the listener. Since all nodes that hold the same label will be grouped together as a community at the end of the process, this avoids grouping together pairs of nodes having a cannot-link relationship.
3. The *constraint processing step* considers both sets of pairwise constraints:
- (a) For each must-link pair, compare the memories of the two nodes to ensure they both share the same highest occurrence frequency label. If they do not, both nodes exchange their most frequently-occurring labels with each other under a condition that each node does not have a cannot link relationship with any node assigned to that label.
 - (b) For each cannot-link pair, compare the memories of both nodes. If both nodes have a common label, remove this label from the node that has the lowest label occurrence frequency.
4. In the *post-processing step*, convert each node's memory into a probability distribution of labels. If the node's probability of a certain label is less than a threshold $r \in [0, 1]$, the label is removed from the node's memory. Then all nodes having the same label are grouped into one community. Nodes that have more than one label correspond to overlapping cases which belong to multiple communities.

4 Evaluation

4.1 Experimental Setup

We now evaluate the performance of PC-SLPA to determine the extent to which introducing varying levels of constraints can improve community detection.

Data. Firstly, we evaluate on synthetic data created using the widely-used LFR generator [12], which can produce networks with properties similar to real-world networks, with overlapping ground truth communities. The selection of network parameters shown in Table 1(a) is based on those used to evaluate the original algorithm SLPA [21] and other works in the literature. We generate two different groups of synthetic networks with different sizes, each containing small and large communities and mixing parameter μ varies from 0.1 to 0.3. Small communities have 10–50 nodes, while large communities have 20–100 nodes. Each group consists of 16 networks with different combinations of the parameter O_m , which controls the number of communities per node. For the first network in each set, all nodes belong to two communities ($O_m = 2$). For each successive network, this parameter value is incremented by 1 until $O_m = 8$ is reached.

Secondly, we consider three real-world networks which have previously been used in the community finding literature [16]: (1) a co-purchasing network from Amazon.com; (2) a friendship network from YouTube; (3) a scientific collaboration network from DBLP. These networks contain annotated ground truth overlapping communities. For each network, we include only the 5,000 largest such

communities, as per [22]. We then perform filtering as per [10] – the remaining communities are ranked based on their internal densities and the bottom quartile is discarded, along with any duplicate communities. Finally, as an additional step, we eliminated extremely small communities. For the Amazon and YouTube networks, communities of size < 5 nodes are removed, while for the DBLP network communities with < 10 nodes are removed. Details of the resulting networks are listed in Table 1(b).

Baselines. To the best of our knowledge, no work has been conducted in the literature regarding pairwise constrained algorithms for finding overlapping communities. Therefore, for the sake of comparison, the PC-SLPA results are compared with outputs of the following popular unsupervised overlapping community detection algorithms: SLPA [21], OSLOM [13], MOSES [19], and COPRA [8]. For OSLOM and MOSES, we use the default parameters recommended by the original authors. For COPRA, we use the settings recommended in [21]. To evaluate the performance of these algorithms relative to the ground truth groupings, we use the overlapping form of Normalized Mutual Information (NMI) [11]. Since SLPA and COPRA are non-deterministic, we average the NMI values over 20 runs.

Experiments. We conducted two experiments in our evaluation. The first aims to assess the performance of the unsupervised algorithms, which provides a baseline for evaluating the performance of our proposed method. For both SLPA and PC-SLPA we use the default parameters values $T = 100$ and $r \in [0, 1]$, as suggested in [14]. The second experiment evaluates the performance of PC-SLPA with increasing numbers of constraints, from 1% to 5% of the total number of possible pairs in each network. Since the initial pairwise constraints are selected at random, we repeat the semi-supervised process for 20 runs and average the resulting NMI scores.

Table 1. The first table lists parameters used for the generation of LFR synthetic networks. The second table summarizes details of the real-world networks.

Parameter	Description	Value	Parameter	Description	Value
N	Number of nodes	1000-5000	t_1	Degree exponent	2
k	Average degree	10	t_2	Community exponent	1
K_{max}	Max degree	50	μ	Mixing parameter	0.1-0.3
C_{min}	Min community size	10/20	O_m	Communities per node	1-8
C_{max}	Max community size	50/100			

Real-world Networks	Amazon	YouTube	DBLP
#Nodes - # Edges - #Communities	7411 - 21214 - 876	6426 - 23226 - 31	7233 - 33045 - 613
Max community size	27	31	38
Min community size	5	5	10
Max communities per node	4	11	8
#Overlapping nodes	1394(18%)	865(13%)	214 (3.3%)

4.2 Results and Discussion

Synthetic networks. For most of the 32 networks, PC-SLPA achieves consistently higher NMI scores than the standard SLPA algorithm, except where $\mu = 0.1$. Here PC-SLPA attains lower NMI values than SLPA, until the number of constraints increases towards 5%. In general, as the percentage of pairwise constraints being used increases, the accuracy of PC-SLPA improves significantly.

When evaluating on LFR-generated networks, different factors can affect algorithm performance, such as the mixing parameters, and the size of both networks and embedded communities. The larger the value of μ , the poorer the communities detected algorithms due to the weaker intra-community connectivity. As we see from Fig. 4, the performance of SLPA drops as μ increases from 0.1 to 0.3. However, PC-SLPA shows more stability with higher values of μ . For instance, in the case of small networks of big communities with $\mu = 0.3$, the NMI score of the standard SLPA is 0.82 at $O_m = 2$ and drops to 0.50 when O_m increased to 8. In contrast, the PC-SPLA algorithm shows a more moderate decrease in accuracy as the value of O_m increases. As for the size of network, both algorithms show better performance when the network increases from 1,000 to 5,000 nodes, with PC-SLPA achieving the best performance on the networks with larger communities.

When comparing PC-SLPA to the baseline algorithms, we observe that COPRA and MOSES show the lowest performance on all synthetic networks. As for OSLOM, it shows slightly better performance than PC-SLPA for networks with a low level of community overlap. However, as the number of communities per node increases, PC-SLPA starts to out-perform all of the baseline algorithms, indicating that it is effective in highly-overlapping contexts.

Table 2 summarizes the performance of all algorithms as win-loss records. Each table entry shows the number of wins of an algorithm (on the rows) over another algorithm (on the columns). To compare two algorithms, we subtract the sum of wins and losses from the total number of synthetic networks. The last column reports rank scores based on the total number of “wins” by each algorithm across all synthetic networks. According to the total number of wins, we rank the highest number as the best algorithm. Then, we order the algorithms from best to worst. As we can see from Table 2, PC-SLPA with 5% pairwise constraints is the top-ranked algorithm, and performs better than the competing benchmark algorithms on 90% of the networks. OSLOM is the next best alternative, followed by SLPA.

Table 2. Win-loss table of NMI performance for all algorithms on 32 synthetic networks.

		Loser					Rank-score	
		OSLOM	MOSES	COPRA	SLPA	PC-SLPA	Total wins	Ranking
Winner	PC-SLPA	22	32	32	29	0	115/128 (90%)	1
	OSLOM	0	32	32	22	10	96/128 (75%)	2
	SLPA	10	32	32	0	3	77/128 (60%)	3
	MOSES	0	0	16	0	0	22/128 (12.5%)	4
	COPRA	0	16	0	0	0	10/128 (12.5%)	4

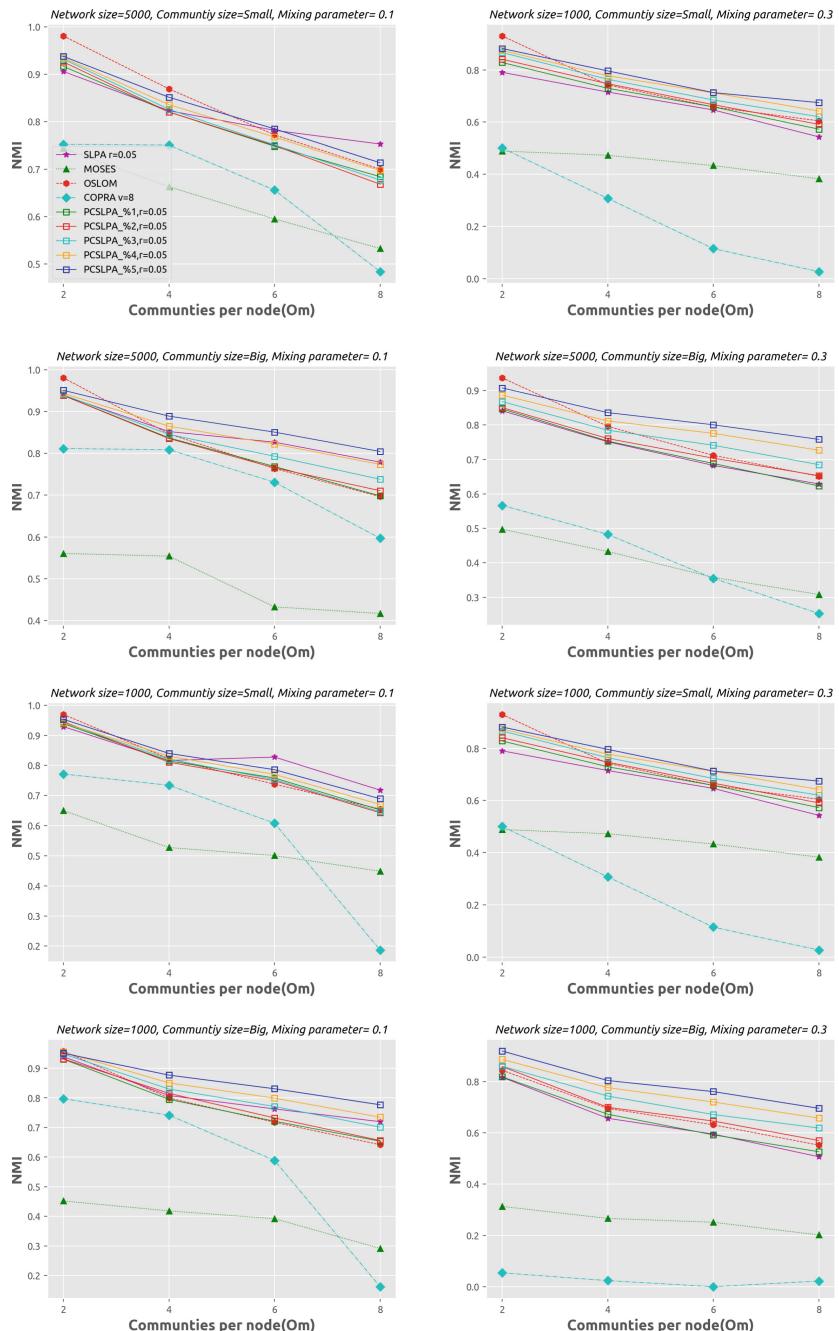


Fig. 4. Performance of all algorithms on 32 synthetic networks, containing both small and large communities, with mixing parameter $\mu \in [0.1, 0.3]$. NMI values are plotted against the number of communities per node (O_m), with 4 networks in each plot.

Table 3. NMI scores of all algorithms on three real-world networks.

	<i>OSLOM</i>	<i>MOSES</i>	<i>COPRA</i>	<i>SLPA</i>	<i>PCSLPA%1</i>	<i>PCSLPA%2</i>	<i>PCSLPA%3</i>	<i>PCSLPA%4</i>	<i>PCSLPA%5</i>
Amazon	0.9668	0.9084	0.96228	0.9568	0.9612	0.9650	0.9678	0.9709	0.9723
YouTube	0.4490	0.4209	0.1907	0.6296	0.6011	0.6130	0.6241	0.6338	0.6439
DBLP	0.8485	0.7707	0.9136	0.8972	0.9059	0.9156	0.9231	0.9278	0.9326

Real-world networks. Next we discuss our experiments on the three real-world networks. We compare the NMI performance of our proposed semi-supervised method with increasing numbers of pairwise constraints, relative to the benchmark algorithms. For the non-deterministic algorithms, 20 runs were executed and NMI scores were averaged. From Table 3, we see that PC-SLPA algorithm achieves high NMI scores (>0.9) on the Amazon and DBLP networks. However, PC-SLPA shows moderate performance on YouTube network, which may be due to the poor separation between the ground truth groups in this network. The addition of $<4\%$ of constraints does not yield an improvement over the unsupervised approach. The effect of high inter-community overlap is far more pronounced in the cases of the OSLOM, MOSES, and COPRA algorithms. Overall, PC-SLPA outperforms the four alternative algorithms in most cases on these networks, with small but consistent increases as the number of provided constraints is increased from 1% to 5%. We would expect this trend to continue as more constraints are added, although it may be impractical to generate larger numbers of constraints in real-world scenarios.

5 Conclusion

We have proposed a new algorithm, PC-SLPA, for detecting overlapping communities, based on the use of a label propagation strategy that is informed by the addition of external information encoded as pairwise constraints. We explored the nuances around the selection of constraints, which are specific to contexts where the communities in the data naturally overlap. Based on extensive experiments, the results show that overlapping community finding algorithms with constraints can considerably out-perform their unconstrained counterparts on both synthetic and real-world networks. As one might expect, their performance improves with increasing number of pairwise constraints. In general, the results show the potential of using semi-supervised strategies for finding overlapping communities. In future work we plan to apply ideas from active learning in order to reduce the annotation burden on the oracle, while maintaining the effectiveness of community detection.

References

1. Ahn, Y.Y., Bagrow, J.P., Lehmann, S.: Link communities reveal multiscale complexity in networks. *Nature* **466**(7307), 761–764 (2010)
2. Amelio, A., Pizzuti, C.: Overlapping community discovery methods: a survey. *Social Networks: Analysis and Case Studies*, pp. 105–125. Springer, Berlin (2014)

3. Blondel, V., Guillaume, J., Lambiotte, R., Lefebvre, E.: Fast unfolding of communities in large networks. *J. Stat. Mech.* **10008**, P10008 (2008)
4. Clauset, A., Newman, M.E., Moore, C.: Finding community structure in very large networks. *Phys. Rev. E* **70**(6), 066,111 (2004)
5. Dreier, J., Kuinke, P., Przybylski, R., Reidl, F., Rossmanith, P., Sikdar, S.: Overlapping communities in social networks. arXiv preprint [arXiv:1412.4973](https://arxiv.org/abs/1412.4973) (2014)
6. Fortunato, S.: Community detection in graphs. *Phys. Rep.* **486**(3–5), 75–174 (2010)
7. Girvan, M., Newman, M.E.: Community structure in social and biological networks. *PNAS* **99**(12), 7821–7826 (2002)
8. Gregory, S.: Finding overlapping communities in networks by label propagation. *New J. Phys.* **12**(10), 103,018 (2010)
9. Habashi, S., Ghanem, N.M., Ismail, M.A.: Enhanced community detection in social networks using active spectral clustering. In: Proceedings of the 31st Annual ACM Symposium on Applied Computing, pp. 1178–1181 (2016)
10. Harenberg, S., et al.: Community detection in large-scale networks: a survey and empirical evaluation. *Wiley Interdiscip. Rev. Comput. Stat.* **6**(6), 426–439 (2014)
11. Lancichinetti, A., Fortunato, S., Kertész, J.: Detecting the overlapping and hierarchical community structure in complex networks. *New J. Phys.* **11**(3), 033,015 (2009)
12. Lancichinetti, A., Fortunato, S., Radicchi, F.: Benchmark graphs for testing community detection algorithms. *Phys. Rev. E* **78**(4), 046,110 (2008)
13. Lancichinetti, A., Radicchi, F., Ramasco, J., Fortunato, S., Ben-Jacob, E.: Finding statistically significant communities in networks. *PLoS ONE* **6**(4), e18,961 (2011)
14. Lee, C., Reid, F., McDaid, A., Hurley, N.: Detecting highly overlapping community structure by greedy clique expansion. In: Workshop on Social Network Mining and Analysis (2010)
15. Leng, M., Yao, Y., Cheng, J., Lv, W., Chen, X.: Active semi-supervised community detection algorithm with label propagation. In: International Conference on Database Systems for Advanced Applications, pp. 324–338. Springer, Berlin (2013)
16. Leskovec, J., Krevl, A.: SNAP datasets: stanford – large network dataset collection (2015)
17. Li, L., Du, M., Liu, G., Hu, X., Wu, G.: Extremal optimization-based semi-supervised algorithm with conflict pairwise constraints for community detection. In: Proceedings of the ASONAM2014, pp. 180–187 (2014)
18. Liu, D., Duan, D., Sui, S., Song, G.: Effective semi-supervised community detection using negative information. *Math. Probl. Eng.* **2015**, 8 (2015)
19. McDaid, A., Hurley, N.: Detecting highly overlapping communities with model-based overlapping seed expansion. In: Proceedings of the ASONAM2010, pp. 112–119 (2010)
20. Newman, M.E.: Modularity and community structure in networks. *Proc. Natl. Acad. Sci.* **103**(23), 8577–8582 (2006)
21. Xie, J., Szymanski, B.K., Liu, X.: SLPA: uncovering overlapping communities in social networks via a speaker-listener interaction dynamic process. In: Proceedings of the IEEE 11th International Conference on Data Mining Workshops, pp. 344–349 (2011)
22. Yang, J., Leskovec, J.: Defining and evaluating network communities based on ground-truth. *Knowl. Inf. Syst.* **42**(1), 181–213 (2015)
23. Zhang, Z.Y.: Community structure detection in complex networks with partial background information. *EPL (Europhys. Lett.)* **101**(4), 48,005 (2013)



Entropy in Network Community as an Indicator of Language Structure in Emoji Usage: A Twitter Study Across Various Thematic Datasets

Ryan Hartman¹, S. M. Mahdi Seyednezhad¹, Diego Pinheiro², Josemar Faustino¹, and Ronaldo Menezes^{3(✉)}

¹ Department of Computer Engineering and Sciences, Florida Institute of Technology, Melbourne, USA

`rhartman2014@my.fit.edu, sseyednezhad2013@my.fit.edu, cruzj2012@my.fit.edu`

² Department of Internal Medicine, University of California, Davis, USA
`pinsilva@ucdavis.edu`

³ BioComplex Laboratory, Department of Computer Science, University of Exeter, Exeter, UK
`r.menezes@exeter.ac.uk`

Abstract. Emojis are emerging as an alternative way to interact and communicate online, and their large-scale adoption has the potential to reveal distinct patterns of human communication and social interactions. In this work, we investigate the hypothesis that emojis are a kind of language. By building networks of emoji co-occurrence, we examine the diversity of the community structure of such networks with regards to predefined categories of emojis. Using four different techniques of community detection, we validate our hypothesis on six Twitter datasets: five from specific topics and one random dataset. Our results demonstrate that the community structure of emojis is more diverse when they are used in non-random topics such as politics and sports, and that Stochastic Block Models appears to extract communities with higher diversity.

1 Introduction

Online social networks have attracted a significant number of users and have rapidly become people's main form of communication. In these social networks, users communicate their feelings and emotions mainly using short pieces of text. The message size limitation, either imposed by the platform or bolstered by the need to say more and type less, can be challenging to convey the right idea and can create misunderstandings. The need to convey meaning succinctly led to the usage of pictographical representations of emotions and ideas in the form of *emoticons* (e.g., “;-)”, “;p”). The large scale adoption of *emoticons* led to the emergence of modern Emojis, which are small images used with the intent of expressing emotions on ideas within text. Nowadays, emoji usage is widespread, and they can be found in many instant messaging apps as well as in social media sites such as Twitter, Instagram, Facebook, and many others.

Currently, more than half of the posts on Instagram contain emojis, and they attract a 17% higher interaction rate when compared to messages which do not include emojis [7]. In 2015, emojis were announced as the fastest growing communication method in the United Kingdom [2], and given the ever-increasing usage of emojis in social media, the assessment of emojis as a form of language deserves further investigation. However, this assessment cannot be made with classical linguistic techniques [21], as emojis inherently lack the features found in structured languages. Instead, the relationship of meaning and diversity of characters can be explored to asses the semantic patterns of the emoji usage phenomena.

One common way to analyze languages and their similarities is to probe the entropy of the words ordering in different families of the languages [9, 12, 14]. Furthermore, word maximum entropy approach is a reliable tool for linguistic disambiguation or part of speech text tagging [11, 19]. In this paper, we consider emojis as words of a hypothetical language in which the entropy of the words will give us the information on how diverse emojis are used.

In this work, we analyze emojis by investigating its semantics through the sequences formed by their co-occurrence in social media posts. In order to communicate and express their feelings, users may use a more diverse set of emojis such that emojis of different categories will appear together with a higher likelihood. This diversity analysis is possible because emojis are originally grouped into seven categories, *Smiley-People*, *Animals-Nature*, *Food-drink*, *Activities*, *Objects*, *Symbols*, and *Flags*; the categories are used to gauge diversity. By building networks of co-occurring emojis, we can unveil their community structure and subsequently assess the diversity of their community structure with respect to categories of emojis to quantify the semantics. In this work, we use the assessment of communities proposed by Hartman et al. [8] to examine the richness of emoji semantics through the diversity of communities from six Twitter datasets collected based on different topics (as described in Table 1) using four techniques of community detection (as described in Table 2). By analyzing the structural properties of such communities as well as their resemblance to available metadata, our work sheds light on the idea that emojis are a form of language.

The contribution of this paper stems from the use of our previously proposed assessment tool for communities detection (in the sense of their ability to capture organizations known *a priori*) [8] to argue that emojis are being used in a way that resembles language structure (entropy) [12].

This paper starts with a description of related works in Sect. 2. We follow in Sect. 3 with the description of the data used in this paper, how we create emoji directed co-occurrence networks, and the community detection algorithms we use in this work. In Sect. 4 we present our main results, concluding the paper with some final considerations in Sect. 5.

2 Related Work

Recent studies on emojis can be divided into two major approaches. In the first approach, researchers aim to understand the meaning of emojis. Barbieri et al. [1] investigated the meaning of Twitter emojis by examining the likelihood of the pairwise appearance and measuring how often emojis convey the same meaning. Novak et al. [15] drew a sentiment map of the 751 most frequently used emojis and found high frequency of usage associated with positive tweets. Wijeratne et al. [20] created a dictionary to make a machine readable sense inventory for emoji. In order to create octuples representing the meaning of the emoji, they used the Unicode, description, image, and keywords attached to the meaning of the emoji.

The second approach attempts to analyze the collective behavior of users based on emoji usage. Novak et al. [15] found that the inter-annotator agreement of tweets containing emojis were higher than the ones without emojis. More interestingly, they acknowledged that users normally use emojis at the end of tweets, and the rank of emojis did not change between different languages. Seyed-nezhad et al. [18] extracted a network of emojis based on their co-occurrence in tweets from two different datasets. They stated the emoji with the maximum edge betweenness could give us a hint about the underlying subject in which the tweets were collected. This work was generalized by Fede et al. [4] by experimenting with more datasets which contained directed networks. They concluded that important emojis are topic dependent. Lu et al. [13] created a network of emojis by point-wise mutual information (PMI). Their findings pointed to a strong correlation between social indicators and patterns of emoji usage.

Using networks of emojis, we can extract the structure of related emojis using community detection techniques. Community detection techniques aim to identify the building blocks of networks and their structural properties. It has been applied to networks of protein interaction, food web, genetic disorders, gene expression, and social networks [5]. However, the most efficient techniques for exploring communities may yield different results [8, 10]. Hence, recent works have used community detection techniques in a holistic approach [8], which includes a comparative analysis of multiple techniques as well as the resemblance of extracted communities to available metadata. This is the direction we pursue in this work.

3 Data and Methods

3.1 Data Collection and Curation

For this study, we collected tweets from Twitter based on different topics at different time periods. The goal is to cover a diverse set of topics, allowing us to examine the effect of such diversity on communities, extracted by state-of-the-art community detection techniques. Moreover, we add to the analysis a topic-free dataset. This data contains tweets randomly sampled from the Twitter feed, without the use of tracking keywords. The random data allows us to observe any possible bias due to using topic-based data. Table 1 shows further information about the datasets used in this work.

In order to show the network statistics are correlated with the emojis' frequency of usage, we calculated the Spearman's rank correlation between the frequency and weighted degree of the nodes. The highly correlated ranks suggest network characteristics such as weighted degree can explain the frequency of emoji usage.

Table 1. Six datasets collected from Twitter. The topics of the datasets covers several areas of interest. The Spearman's rank correlation is calculated for each dataset between the frequency and weight-degree of the nodes.

Label	Dataset	Characteristics	# Tweets (Millions)	% Containing emojis	Collection period	Spearman's rank correlation
D_1	$G\text{-}20$	Surnames of G-20 countries' leaders	10.6	7%	Aug. 24–Sep. 24, 2014	0.94
D_2	$Organ$	Organ transplantation terms	2.5	9%	Oct. 2015–Apr. 2017	0.85
D_3	$rioSports$	Sports in the 2016 Rio Olympics	1.8	1%	Aug. 05–Aug. 21, 2016	0.95
D_4	$rioTerms$	"Olympics" in different	5.8	1%	Aug. 05–Aug. 21, 2016	0.92
D_5	WWC	Women's World Cup 2015	10.7	1%	Jun. 06–Jul. 05, 2015	0.91
D_6	$randSample$	2 months samples from Twitter	168.5	<1%	Dec. 13, 2016–Jan. 31, 2017	0.97

In summary, we have data related to politics (D_1), health (D_2), sports (D_3 , D_4 , and D_5), as well as a random collection of tweets (D_6). The random sample D_6 has the lowest percentage of tweets containing emojis, while the organ transplantation D_2 collection has the greatest amount.

3.2 Network Construction

The main focus of this paper is on comparing prominent community detection techniques and the characteristics of the communities they uncover for a variety

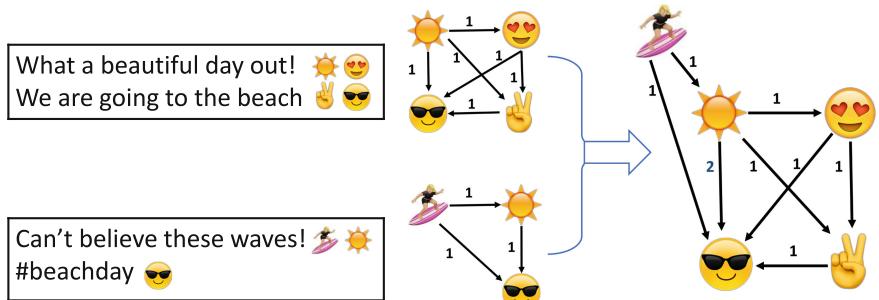


Fig. 1. Directed network of emojis. We create a connection from emoji to emoji in the order they appear in a tweet. This process is repeated for every tweet in the dataset. Then we accumulate all the sub-networks extracted from tweets into a main directed network of emojis.

of datasets. Differing from previous works [18], here we consider that the order of emojis appearing in a tweet is fundamental and hence better represented using directed links.

A directed network of emojis gives us an opportunity to study the collective usage of emojis on social media. Additionally, different sequences of emojis may reflect different feelings expressed by users. For example, someone tweeting “I loved ❤️ this place until that horrible 😞 incident happened”, the meaning is different from another tweet such as “This place is sometimes horrible 😞, but I love ❤️ it anyway!”. Note that the order of the emojis is related to the sentiment being expressed. In order to build the directed network, we connect each emoji to every subsequent one appearing in the same tweet. Figure 1 shows the process of making directed weighted links between emojis.

3.3 Community Detection Techniques and Evaluation Criteria

Since the emoji co-occurrence networks are weighted and directed, we used state-of-the-art techniques that support networks with these features [5]. Table 2 describes the selected techniques and their respective approach to identifying communities. For each emoji network that was constructed, all techniques are applied and the characteristics of the communities found are then analyzed. []

Table 2. Community detection techniques

Acronym	Name	Approach	Description
\mathcal{IM}	InfoMap	Bottom-up	Builds a map of information flow in the network using a random walk. Finding a community is equivalent to minimizing the flow representation by applying a compression technique [17]
\mathcal{BM}	Stochastic block models	Top-down	Applies maximum likelihood estimation to infer the latent block division in the empirical network. Such inference is equivalent to the entropy minimization of the network ensemble [16]
\mathcal{LP}	Label propagation	Hybrid	Based on belief propagation, where each node spreads its label to its neighbors. Convergence of labels uncovers the community structure [6]
\mathcal{LM}	Louvain modularity	Bottom-up	Works by optimizing network modularity, which is the tightness of node connectivity into modules/communities in the empirical network relative to a null model [3]

To begin, we examined the size characteristics of the communities found by these four techniques. Then, we apply an unsupervised evaluation by computing the communities’ conductance. The conductance C of a community k , measures the ratio between the intragroup and intergroup connectivity of the communities [5] and is computed as shown in Eq. 1.

$$C(k) = \frac{\sum_{i \in k, j \notin k} w_{ij}}{\sum_{i \in k, j} w_{ij}} , \quad (1)$$

where w_{ij} is the weight of the link connecting nodes i and j . In this sense, well-structured communities exhibit a higher volume of edges between nodes within a community compared to edges going to the outside of the community.

We also conduct a supervised evaluation using the idea of rank stability for community detection [8] which was proposed as a way to measure the homogeneity of a particular community using the attribute values of nodes within that community as follows:

$$E(n) = - \sum_{t=1}^L p_{kt}(n) \log_2[p_{kt}(n)] , \quad (2)$$

where $p_{kt}(n)$ is the proportion of nodes in community k that are associated with attribute t in their rank n , and L is the number of emoji categories. In this work, we only have one attribute ($n = 1$) which can be one of the six categories of emojis.

4 Results

The statistical and structural properties of extracted communities can vary depending on the community detection technique. For instance, the number of communities extracted on large-scale networks significantly vary depending on the community detection technique [8]. We organized our results in three parts. First, we characterize two structural properties of major importance, namely, community size and conductance. Then, we characterize the communities of emoji networks with regards to emoji categories known a priori to shed light in the context of emojis as language. Lastly, we present how these macroscopic characteristics are related and how a careful exploration of such relationship has the potential to help us gain insights on the structure and function of emojis.

Overall, emoji networks exhibit a well-defined community structure with regards to their size which is slightly shifted depending on the dataset (Fig. 2, left). Exceptionally, $\mathcal{L}\mathcal{P}$ appears to find communities with typically greater size, it identifies the least number of communities, whereas other techniques find ten times more communities; this result is consistent with previous work [8]. Although the distribution of nodes within communities is an important aspect when identifying groups of interrelated emojis, we also need to quantify the extent in which extracted communities exhibit desirable structural properties.

Despite the lack of a general definition of a community, the number of links running between nodes within the community (i.e., internal edges) should be larger than the number of links running from nodes within the community to nodes outside the community (i.e., external edges). Conductance extends such definition for weighted networks (Eq. 1). The conductance of communities vary depending on the technique and dataset (Fig. 2, middle). $\mathcal{I}\mathcal{M}$ and $\mathcal{L}\mathcal{M}$ show a

more similar conductance distribution when compared to \mathcal{BM} . Precisely, \mathcal{BM} has a higher likelihood of identifying communities with greater typical conductance. Similarly, \mathcal{IM} and \mathcal{LM} are likely to identify communities with moderate values of typical conductance. Lastly, \mathcal{LP} is likely to identify communities with lower conductance.

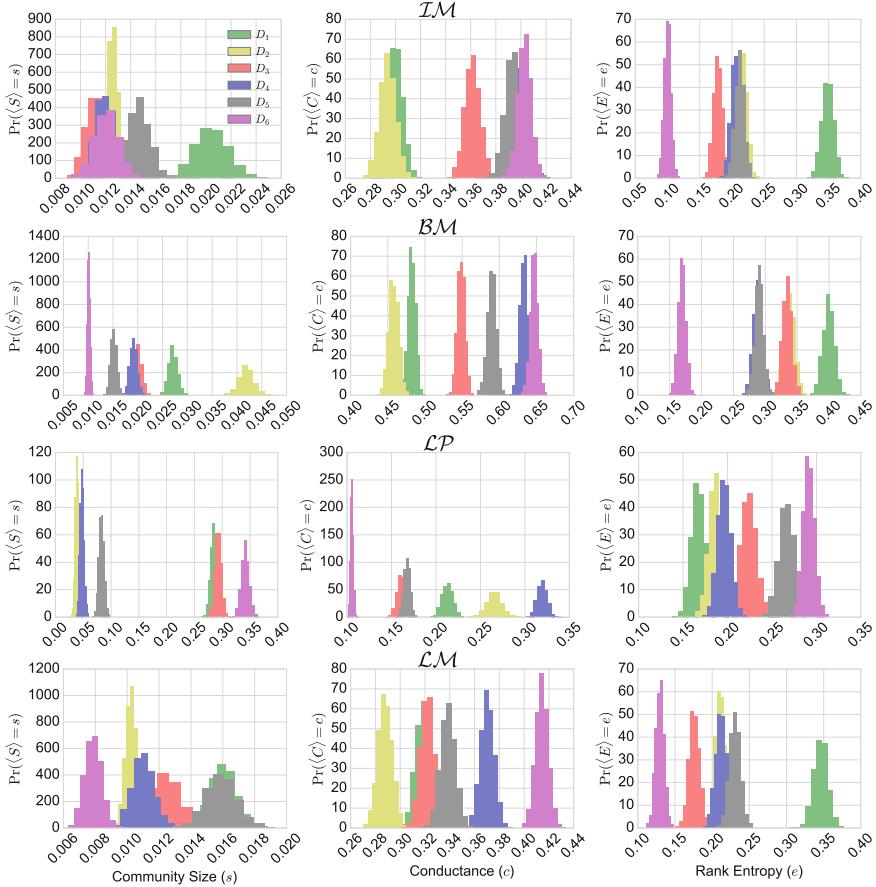


Fig. 2. Community size (S , left), conductance (C , middle), and rank entropy (E , right) of emoji networks characterized across six datasets of different thematics $D_1 \dots 6$ using four community detection techniques, namely, infomap \mathcal{IM} , block models \mathcal{BM} , label propagation \mathcal{LP} , and Louvain modularity \mathcal{LM} .

Besides analyzing the structural properties of communities, we are also interested on evaluating communities to gain insight on the usage of emojis as a language. Here, we apply an entropy based metric to explore the levels of meaning in emoji usage. Our assumptions are that the higher the diversity of emoji categories within a community, the higher the level of meaning conveyed by these emojis.

We carry out this analysis by assessing the extent in which emojis within a community resemble the official emoji categories, using the rank entropy of communities [8]. The rank entropy (Eq. 2) varies depending on the employed technique and underlying dataset (Fig. 2, right). Overall, the highest rank entropies are exhibited by communities extracted using \mathcal{BM} as well as communities extracted from dataset D_1 . Conversely, the lowest entropies are exhibited by communities extracted using \mathcal{LP} as well as communities extracted from the random dataset D_6 . \mathcal{LP} presents some exceptions to aforementioned statements.

We can characterize communities of emojis by their structural properties, such as community size and conductance, as well as by their resemblance with available metadata using the rank entropy. Besides the independent characterization of community size, conductance, and rank entropy, we can also examine the relationship between these characteristics and unveil additional properties of the extracted communities. Indeed, these characteristics are related to each other (Table 3). Overall, communities with greater size tend to be moderately associated with lower conductance and higher rank entropy; however, there is a lack of a significant association between conductance and rank entropy. After controlling for community size, conductance appears to be associated with rank entropy mainly depending on the underlying dataset. For instance, it is moderate in datasets D_1 and D_2 , and it is absent in the random dataset D_6 .

Table 3. Relationship between community size S , conductance C , and rank entropy E as measured by Pearson correlation. Community size S is negatively correlated with conductance C and positively correlated with rank entropy E . Even after controlling for community size, there is a lack of significant correlation between conductance and rank entropy in all techniques, except for label propagation. ($\hat{\rho}_{SE \cdot S} = 0.12$, $p > .1$). However, further looking such correlation in each dataset across multiple techniques, conductance is positively correlated with rank entropy, which is strongest in D_1 and weakest in the random dataset D_6 .

	\mathcal{IM}	\mathcal{BM}	\mathcal{LP}	\mathcal{LM}	$D1$	$D2$	$D3$	$D4$	$D5$	$D6$
$\hat{\rho}_{SC}$	-0.14	-0.12	-0.24	-0.05	-0.19	-0.14	-0.14	-0.17	-0.13	-0.19
$\hat{\rho}_{SE}$	0.41	0.37	0.56	0.46	0.36	0.31	0.36	0.35	0.34	0.31
$\hat{\rho}_{CE}$	-0.03	0.01	-0.03	-0.03	0.11	0.12	0.10	0.05	0.01	-0.05
$\hat{\rho}_{CE \cdot S}$	0.01	0.05	0.12	-0.02	0.20	0.18	0.16	0.12	0.06	0.01

5 Conclusion

Emojis are a *de facto* form of communication; their simplicity and ease of use were fundamental for their wide adoption. When we build emoji networks, we can exhibit the structural properties of its usage with significant detail. These networks show the function of emojis and how they relate to language. In this work, we investigated the hypothesis that emojis are a form of language by

building networks of emojis co-occurring on social media posts and subsequently analyzing the diversity of their community structure. To gain insights on how emojis are used, we compare the diversity of communities from specific topics such as politics and sports with that of randomly collected dataset. We found that users tend to communicate on social media using emojis of different categories. In this sense, the Stochastic Block Model would be more suitable way of doing community analysis on these networks because it is capable of finding more diverse (i.e., higher rank entropy) and well-formed communities (i.e., higher conductance). Yet, other possibilities to build emoji networks remain to be explored such as those based on risk ratio, pointwise mutual information and Φ -correlation.

Finding suitable datasets is quite hard because emojis are used in mostly personal communications. The usage on Twitter allowed us to perform this work but would be interesting to investigate the network of emojis on other social media such as Instagram to verify if the language characteristics we found here are also present. We have attempted to use another dataset from Reddit but unfortunately emoji is not widely used on Reddit.

Acknowledgements. Diego Pinheiro and Josemar Faustino would like to thank the Science Without Borders program (CAPES, Brazil) for financial support under grants 0624/14-4 and 1043-14-5, respectively. This material is based upon work supported by the National Science Foundation under Grant No. CNS 09-23050.

References

1. Barbieri, F., Ronzano, F., Saggion, H.: What does this emoji mean? a vector space skip-gram model for twitter emojis. In: Language Resources and Evaluation Conference. LREC, Portoroz, Slovenia (2016)
2. Doble, A.: UK's fastest growing language is... emoji. <http://www.bbc.co.uk/newsbeat/article/32793732/uk-s-fastest-growing-language-is-emoji> (2015)
3. Dugué, N., Perez, A.: Directed Louvain: maximizing modularity in directed networks. Technical Report, Université d'Orléans. <https://hal.archives-ouvertes.fr/hal-01231784> (2015)
4. Fede, H., Herrera, I., Seyednezhad, S.M., Menezes, R.: Representing emoji usage using directed networks: a twitter case study. In: International Workshop on Complex Networks and their Applications, pp. 829–842. Springer (2017). https://doi.org/10.1007/978-3-319-72150-7_67
5. Fortunato, S.: Community detection in graphs. Phys. Rep. **486**(3–5), 75–174 (2010). <http://dx.doi.org/10.1016/j.physrep.2009.11.002>
6. Gaiteri, C., Chen, M., Szymanski, B., Kuzmin, K., Xie, J., Lee, C., Blanche, T., Chaibub Neto, E., Huang, S.C., Grabowski, T., Madhyastha, T., Komashko, V.: Identifying robust communities and multi-community nodes by combining top-down and bottom-up approaches to clustering. Sci. Rep. **5**(1), 16,361 (2015). <https://doi.org/10.1038/srep16361>
7. Gottke, J.: Instagram emoji study: emojis lead to higher interactions. <https://www.quintly.com/blog/2017/01/instagram-emoji-study-higher-interactions/> (2017)

8. Hartman, R., Faustino, J., Pinheiro, D., Menezes, R.: Assessing the suitability of network community detection to available meta-data using rank stability, vol. 17, pp. 162–169. ACM Press, New York, New York, USA (2017). <https://doi.org/10.1145/3106426.3106493>
9. Kalimeri, M., Constantoudis, V., Papadimitriou, C., Karamanos, K., Diakonos, F.K., Papageorgiou, H.: Word-length entropies and correlations of natural language written texts. *J. Quant. Linguist.* **22**(2), 101–118 (2015)
10. Lancichinetti, A., Fortunato, S.: Community detection algorithms: a comparative analysis. *Phys. Rev. E* **80**(5), 056,117 (2009). <https://doi.org/10.1103/PhysRevE.80.056117>
11. Le-Hong, P., Roussanaly, A., Nguyen, T.M.H., Rossignol, M.: An empirical study of maximum entropy approach for part-of-speech tagging of vietnamese texts. In: *Traitement Automatique des Langues Naturelles-TALN 2010*, p. 12. <https://hal.inria.fr/inria-00526139> (2010)
12. Levitin, L.B., Reingold, Z.: Entropy of natural languages: theory and experiment. *Chaos Solitons Fract.* **4**(5), 709–743 (1994). [https://doi.org/10.1016/0960-0779\(94\)90079-5](https://doi.org/10.1016/0960-0779(94)90079-5)
13. Lu, X., Ai, W., Liu, X., Li, Q., Wang, N., Huang, G., Mei, Q.: Learning from the ubiquitous language: an empirical analysis of emoji usage of smartphone users. In: *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, pp. 770–780. ACM (2016). <http://dx.doi.org/10.1145/2971648.2971724>
14. Montemurro, M.A., Zanette, D.H.: Universal entropy of word ordering across linguistic families. *PLoS One* **6**(5), e19,875 (2011)
15. Novak, P.K., Smailović, J., Sluban, B., Mozetič, I.: Sentiment of emojis. *PloS One* **10**(12), e0144,296 (2015)
16. Peixoto, T.P.: Hierarchical block structures and high-resolution model selection in large networks. *Phys. Rev. X* **4**(1), 1–18 (2014). <http://dx.doi.org/10.1103/PhysRevX.4.011047>
17. Rosvall, M., Bergstrom, C.T.: Maps of random walks on complex networks reveal community structure. *Proc. Nat. Acad. Sci.* **105**(4), 1118–1123 (2008). <http://www.pnas.org/cgi/doi/10.1073/pnas.0706851105>
18. Seyednezhad, S.M., Menezes, R.: Understanding subject-based emoji usage using network science. In: *Workshop on Complex Networks Complenet*, pp. 151–159. Springer (2017). https://dx.doi.org/10.1007/978-3-319-54241-6_13
19. Suárez, A., Palomar, M.: A maximum entropy-based word sense disambiguation system. In: *Proceedings of the 19th International Conference on Computational Linguistics*, vol. 1, pp. 1–7. Association for Computational Linguistics (2002). <https://doi.org/10.3115/1072228.1072343>
20. Wijeratne, S., Balasuriya, L., Sheth, A., Doran, D.: Emojinet: building a machine readable sense inventory for emoji. In: *International Conference on Social Informatics*, pp. 527–541. Springer (2016). https://dx.doi.org/10.1007%2F978-3-319-47880-7_33
21. Winograd, T.: Understanding natural language. *Cognit. Psychol.* **3**(1), 1–191 (1972). [https://doi.org/10.1016/0010-0285\(72\)90002-3](https://doi.org/10.1016/0010-0285(72)90002-3), <http://www.sciencedirect.com/science/article/pii/0010028572900023>



TimeRank: A Random Walk Approach for Community Discovery in Dynamic Networks

Ilias Sarantopoulos^{1,2(✉)}, Dimitrios Papatheodorou^{2,4}, Dimitrios Vogiatzis^{1,3}, Grigoris Tzortzis¹, and Georgios Paliouras¹

¹ National Centre for Scientific Research (NCSR) “Demokritos”, Athens, Greece
isaranto@huaeb.gr, dimitrv@iit.demokritos.gr, gtzortzi@iit.demokritos.gr,
paliourg@iit.demokritos.gr

² Athens University of Economics and Business, Athens, Greece
dimitrispapatheodorou95@gmail.com

³ The American College of Greece, Athens, Greece

⁴ Aalto University, Espoo, Finland

Abstract. In this work we consider the problem of discovering communities in time evolving social networks. We propose TimeRank, an algorithm for dynamic networks, which uses random walks on a tensor representation to detect time-evolving communities. The proposed algorithm is based on an earlier work on community detection in multi-relational networks. Detection of dynamic communities can be done in two steps (segmentation of the network into time frames, detection of communities per time frame and tracking of communities across time frames). Alternatively it can be done in one step. TimeRank is a one step approach. We compared TimeRank with Non-Negative Tensor Factorisation and Group Evolution Discovery method on synthetic and real world data sets from Reddit.

1 Introduction

Community detection in networks represented as static graphs, is a well-studied problem [7], but real world networks demonstrate temporal activity incurring changes in their structure over time. Along with the network, the structure and the composition of the communities also change, i.e. they *evolve*. Given a network that is observed over a time period, it can be divided into equal T time intervals or timeframes, then the social network can be represented as a sequence of graphs $\{G_1, G_2, \dots, G_t\}$, where each graph includes the interactions between the nodes observed in the corresponding timeframe. A *dynamic community* is made of a chain of corresponding static communities along the time frames of the network. Some of the issues that are related to the problem of dynamic community detection, is the static community detection per time frame, the tracking of static communities along time frames, as well as the granularity of the segmentation.

The motivating idea in the current work is to map a temporal network to a weighted static network, where the weights capture temporal relations. Subsequently any existing algorithm for community detection can be applied on the static network, and finally the communities of the static network can be mapped back to the original temporal network.

The main contributions of this paper are:

- TimeRank, a random walk algorithm for dynamic community detection that is based on an earlier algorithm for community detection on multi-relational networks.
- Experimental investigation of the merits of TimeRank on synthetic and real world data sets and comparison with two other methods.

The rest of the paper is organised as follows; Sect. 2 describes the related work; Sect. 3 introduces the background method MutuRank, an algorithm for community discovery in multi-relational networks; Sect. 4 describes the proposed method TimeRank; experimental results are presented in Sect. 5; finally, conclusions are drawn in Sect. 6.

2 Related Work

Dynamic community discovery is represented by two main approaches: two step and one step methods. Two step methods tend to see community detection and tracking as two discrete procedures. During the first step, time is divided into possibly overlapping frames and a community detection method is applied to the static graph of each frame (usually the Louvain method is chosen [5]). During the second step, communities between neighbouring time-frames are compared and matched using a similarity measure and depending on the degree of similarity an event is assigned on each evolution between two consecutive time-frames (form, dissolve, merge, split, grow, shrink, continue). Mostly the Jaccard similarity is being used. These two-step approaches do not tend to focus on the first step (community detection) but rather on modelling the tracking of the evolution of the detected communities. Such methods are described in [6, 9–11, 20, 21]. One step approaches do not separate community detection from tracking of their evolution but perform both in one step [8, 23]. An interesting instance of one step methods, performs both community tracking and community prediction in a parameter free way and it is based on tensor decomposition and the minimum description length principle [3]. Also approaches for detection and prediction have been proposed that are based on link and content analysis [2].

The ***Group Evolution Discovery (GED)*** method is a two step method (i.e. community tracking) [6]. For the second step a new measure is introduced called inclusion, which measures both the quality and quantity of the inclusion of one group in another. $I(G_1, G_2) = \frac{|G_1 \cap G_2|}{|G_1|} \cdot \frac{\sum_{x \in (G_1 \cap G_2)} SP_{G_1}(x)}{\sum_{x \in (G_1)} SP_{G_1}(x)}$, where the first fraction measures the group quantity and the second the quality, and $SP_{G_1}(x)$ is the value of social position of x in group G_1 . Social position is a centrality measure.

For each of the matched communities we want to discover its evolution. Group evolution is a sequence of changes succeeding each other that happen in sequential timeframes. In [6] the authors extended the events proposed in [4, 19]. They identified seven type of rule-based events that describe the change in the state of a group or groups between two timeframes (T_i and T_{i+1}): continuing, shrinking, growing, splitting, merging, dissolving, forming.

Non-Negative Tensor Factorisation (NNTF). The interaction among users across time can be represented by a 3-dimensional tensor $T \in \mathbb{R}^{N \times N \times S}$, where N is the number of nodes/users of the network and S the number of network snapshots. Tensor factorisation techniques and PARAFAC in particular [12] has been used in the detection of dynamic communities [8]. In tensor factorisation the following optimization problem is solved: $\min_{\mathbf{A}, \mathbf{B}, \mathbf{C}} \|T - \mathbf{A}, \mathbf{B}, \mathbf{C}\|_F^2$, where matrices \mathbf{A} and \mathbf{B} are square of size $N \times k$ and associate a weight for the membership of each node to each community structure, while \mathbf{C} , which is of size $S \times k$, associates the communities with time intervals and k , which is provided as input, is the number of latent factors or communities that are discovered. In the case of undirected networks $\mathbf{A} = \mathbf{B}$. This way clustering and tracking is performed in one step, with matrix \mathbf{A} representing the clustering output and matrix \mathbf{C} the tracking output.

3 Background: MutuRank

We start by describing MutuRank, an algorithm for discovering community structure in multi-relational networks, which we later transform to fit the problem of dynamic community finding.

A multi-relational network (MRN), is a network where there can be many types of edges between any two nodes, for example in a social network an edge between two users can have multiple meanings: follow, like, retweet etc. A well known problem in MRNs is discovering community structure. The problem is similar to discovering communities in single-relational networks (SRNs). Because of the homogeneity of the nodes across the relations, we want to find a good k -way partition $\mathcal{P} = \{C_1, \dots, C_k\}$, where C_k is the k th community and $C_1 \cup \dots \cup C_k \subseteq V$ and V is the set of nodes that exist in the network, by considering all relations. The MRN is represented by a three dimensional $n \times n \times m$ affinity tensor, where n is the number of nodes and m is the number of relations. Let $\mathcal{A} = [a_{i,j,d}]$ be the affinity tensor and $a_{i,j,d} \in \mathbb{R}$ denote the relation strength between nodes i and j under the d th relation. Then $a_{i,j,d} = 1$ if i and j share an edge in d , otherwise $a_{i,j,d} = 0$.

In this context, co-ranking frameworks such as MutuRank [22] simultaneously rank the n nodes and the m relations in order to get a weighted importance p_i for each node i and q_d relation d and thus produce an SRN calculated as follows:

$$w_{i,j} = \sum_{d=1}^m q_d \cdot a_{i,j,d}, \quad (1)$$

where $\mathbf{p} = (p_1, p_2, \dots, p_n)$ and $\mathbf{q} = (q_1, q_2, \dots, q_m)$ are the weight/probability distribution vectors of nodes and relations respectively. In SRNs obtaining the distribution \mathbf{p} which models node importance can be achieved with well known algorithms like HITS [13] and Pagerank [18]. But in the case of MRNs nodes and relations yield mutual influence, thus there is the need to co-rank nodes and relations simultaneously, hence \mathbf{p} is implicitly used in the calculation of \mathbf{q} in Eq. (3) (see below).

MutuRank uses the mutual influence in order to rank relations and eventually transform the multi-relational network into a single-relational network (SRN), where there exists only one type of interaction between nodes, and communities can be detected using any community detection algorithm applicable to SRNs.

As we envision a random walk applied on a MRN, we need to construct two probability tensors $\mathcal{O} = [o_{i,j,d}]$ and $\mathcal{R} = [r_{i,j,d}]$ with respect to nodes and relations by normalizing the entries of \mathcal{A} as follows $o_{i,j,d} = \frac{a_{i,j,d}}{\sum_{l=1}^n a_{l,j,d}}$, $r_{i,j,d} = \frac{a_{i,j,d}}{\sum_{e=1}^m a_{i,j,e}}$

MutuRank also considers influence exerted by prior distributions of nodes and relations. To conclude, we have the following iterative equations for computing the ranking scores of nodes and relations simultaneously:

$$p_i^t = \alpha \sum_{j=1}^n \sum_{d=1}^m p_j^{t-1} \cdot o_{i,j,d} \cdot \text{Prob}^{t-1}[d|j] + (1 - \alpha)p_i^*, \quad (2)$$

$$q_d^t = \beta \sum_{i=1}^n \sum_{j=1}^n p_j^{t-1} \cdot r_{i,j,d} \cdot \text{Prob}^{t-1}[i|j] + (1 - \beta)q_d^* \quad (3)$$

where $\mathbf{p}^* = (p_1^*, p_2^*, \dots, p_n^*)$ and $\mathbf{q}^* = (q_1^*, q_2^*, \dots, q_m^*)$ are the prior distributions of nodes and relations respectively, and α, β are two parameters to balance the knowledge coming from network structure and the prior knowledge and are also called damping factors. Ideally prior distributions are obtained from a domain expert who can provide us with prior knowledge which quantifies the importance of nodes and relations. Such prior knowledge is very difficult to obtain so in most cases we can simply set $p_i^* = 1/n$, $1 \leq i \leq n$ and $q_d^* = 1/m$, $1 \leq d \leq m$. Incorporating the prior knowledge is helpful to deal with dangling nodes (that have zero out-degree) [18]. More precisely, given a dangling node i , $\forall 1 \leq j \leq n$, $a_{i,j,d} = 0$ leads to $\text{Prob}[i|j] = \text{Prob}[d|j] = 0$. Therefore, the weight of the sink node cannot be updated by the iterative process (Fig. 1).

In the case where we have one relation, so $m = 1$ and $p_i^* = 0$, $1 \leq i \leq n$, Eq. (2) becomes $p_i^t = \sum_{j=1}^n p_j^{t-1} \cdot \frac{1}{k_j}$, where k_j is the degree of node j . In this case we have exactly the same computation as in Pagerank [18], thus Pagerank can be regarded as a special case of MutuRank. Upon running the above iterative process we can transform the MRN into a SRN with the help of Eq. (1). Afterwards we can employ a clustering algorithm on the graph described by two-dimensional matrix w to extract community structure.

4 TimeRank

In this section we describe the proposed method, TimeRank, a random walk algorithm for dynamic community detection that is based on MutuRank.

Our intention is to transform the problem of finding communities in MRNs to that of dynamic community finding and employ a similar procedure to that of MutuRank to get the dynamic communities. To do that we use a tensor representation for our dynamic network. The first two dimensions refer to the nodes and the third dimension to the timeframes, in essence treating the timeframes as relations. A tensor “slice” for a particular timeframe represents the adjacency matrix of the network at that timeframe. The difference in our representation is that in each slice, node i is represented by a set of distinct nodes, one for each timeframe which we call node images. Let N be the numbers of nodes, and T the number of timeframes. Then node i will actually be a set of the different images of the node, one for each timeframe, $i = \{i_1, i_2, \dots, i_T\}$. This way the number of distinct nodes in our representation is $N \times T$. The dimensions of the new tensor are $(N \times T) \times (N \times T) \times (T)$.

$$\mathcal{A}[:, :, t] = \begin{pmatrix} a_{11t} & a_{12t} & a_{13t} & \dots & a_{1(N \times T)t} \\ a_{21t} & a_{22t} & a_{23t} & \dots & a_{2(N \times T)t} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{(N \times T)1t} & a_{(N \times T)2t} & a_{(N \times T)3t} & \dots & a_{(N \times T)(N \times T)t} \end{pmatrix}$$

Given the above representation the dynamic communities we want to discover will include node images from separate timeframes.

Inter-Timeframe Edges Community detection algorithms analyse the structure of the network, attempting to produce communities, which are sets of

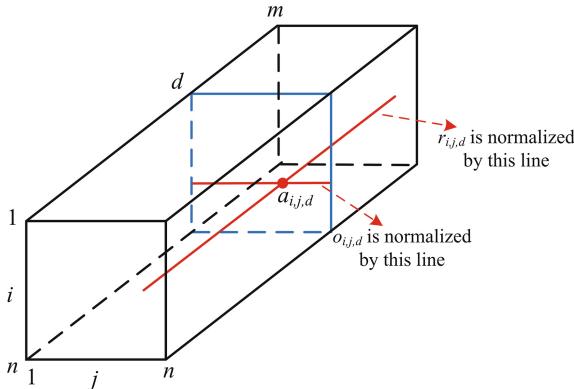


Fig. 1. Depiction of the normalisation of tensors \mathcal{O} and \mathcal{R} in MutuRank and TimeRank (source [22]).

nodes that are more densely connected with each other than the other nodes in the network. With the current representation it would not be possible for clusters from different timeframes to be placed in the same dynamic community, because neither the tensor “slices” for the timeframes nor the node images in these “slices” are connected between them in any way. In this context we introduce inter-timeframe edges, which connect a node with its image in other timeframes. Dynamic communities contain separate groups (timeframe clusters) within them, which originate from different timeframes. These groups belong to the same dynamic community because they share some common nodes. The reasoning behind adding the inter-timeframe edges is that by connecting each node with its image we bring these groups closer to each other, thus allowing dynamic patterns to be more easily unveiled.

Given node i which exists in timeframe t_1 of a dynamic network, we propose two ways of connecting the node with its occurring images in the other timeframes $\{i_2, \dots, i_T\}$. If a node does not appear in a specific timeframe, it will not be connected with its corresponding node images. Apart from the inter-timeframe edges there are also the original edges that exist in the graphs. Since each tensor slice represents a timeframe, it will comprise the edges that exist in the corresponding timeframe as they appear on the initial graph as well as the added inter-timeframe edges. The two variations of TimeRank rely on the following two representations:

- **One to All Connection (OAC)** In “One to All Connection” we connect each node i_t with all its occurrences in other timeframes. For each node i_t we add the following pairs of edges in the network: $\{(i_1, i_t), \dots, (i_{t-1}, i_t), (i_{t+1}, i_t), \dots, (i_T, i_t)\}$.
- **Next Occurrence Connection (NOC)** In “Next Occurrence Connection” we connect each node i_t with its images in the previous and next timeframes in which this node exists. For each node i_t we add the following pairs of edges in the network: $\{(i_{t-1}, i_t), (i_{t+1}, i_t)\}$.

Algorithm 1 TimeRank Algorithm

- 1: Create $(N \times T)$ adjacency matrices for each timeframe
 - 2: Add inter-time edges
 - 3: Compose tensor $\mathcal{A} \in \mathbb{R}^{(N \times T) \times (N \times T) \times (T)}$
 - 4: Apply MutuRank algorithm on \mathcal{A} and get ranking of timeframes
 - 5: Create time-weighted network with $(N \times T)$ nodes
 - 6: Perform clustering on this network and extract dynamic communities
-

In our implementation of TimeRank, in the final step we create an $N \times T$ matrix in the same fashion like MutuRank does, and we apply spectral clustering on that matrix to obtain the k dynamic communities, where the number of dynamic communities k is provided as input. In (1) Step 4 can be omitted, and

instead of running MutuRank we can use a uniform distribution for \mathbf{q} during step 5. In MutuRank [22] irreducibility is a widely-used assumption. We also utilise irreducibility in TimeRank as an important assumption in order to derive the existence of the two equilibrium distributions.

5 Experimentation

5.1 Data Description and Engineering

Synthetic Data We constructed two different networks with two different event types, composed from 1000 nodes that span over 5 timeframes. We used the Dynamic Benchmark Network Generator [11], which is based on [15]. We set the seed to 10 in order for our dataset to be easily reconstructed. Nodes have a mean degree of 20 and a maximum degree of 40. The networks begin at $t = 1$ with approximately 32 communities, with each of the communities having a size in the range [15, 50]. The above parameters are common in the two datasets, with each having additional parameters regarding the events it implements. Specifically in the *Expand/Contract* dataset 10 randomly selected communities expand or contract by 25% of their previous size, while in the *Hide/Appear* dataset 10% of the communities hide and reappear between timeframes.

Real World Data: Reddit. Reddit is a popular social network that has the interesting construct of the *subreddits* which are theme-defined sub-communities of the network, e.g. “cats”, “AskReddit”, “Philosophy”. The subreddits are explicitly used as the ground truth of our community tracking and the network is constructed through employing the users as nodes and their interactions/replies in the comment section as edges. We used the JSON comment dumps that can be found here¹, cleaned them, preprocessed them and sampled from them in order to have our final dataset based on the comments of the months 9/2010 and 10/2010. We removed inactive users and small, inactive or very large subreddits. The timeframe length we used was 1 week and the experiments were performed for 4 and 8 non-overlapping timeframes. Thus across a month there are 4 static graphs of communities that macroscopically compose a dynamic graph.

5.2 Evaluation Measures

To evaluate the results of the experiments we use a set of extrinsic metrics. These metrics require the actual community grouping to be known, commonly referred to as the ground truth. A detected community structure is considered good if it is close to the ground truth. There are different approaches in how to quantify the term “close”. We use set matching methods (Normalised Mutual Information [14]) and pairwise evaluation measures (Omega Index [16], BCubed [1]). Let $\mathcal{T} = \{T_1, T_2, \dots, T_k\}$ be the ground truth community structure of the network and $\mathcal{C} = \{C_1, C_2, \dots, C_n\}$ be the community structure detected by an algorithm. Evaluation in this work is done in the context of dynamic communities, i.e.

¹ <http://files.pushshift.io/reddit/comments>.

ground truth dynamic communities are compared to the dynamic communities generated by the tested algorithms. The omega index as described in [16] can be found in GitHub² and the Python Package Index³.

5.3 Results

Below there are the results of some interesting experiments using the synthetic and the Reddit data. The methods used were TimeRank-AOC (TR-AOC) and TimeRank-NOC (TR-NOC) with MutuRank and with uniform timeframe distributions (-U) for q (Table 2), Non Negative Tensor Factorisation (NNTF) and Group Evolution Discovery (GED). Overlapping versions were also tested, but showed no promising results. All code is available in GitHub.^{4,5}

Benchmark Dynamic Networks In the synthetic *Expand/Contract* dataset, from the existing 32 communities, 10 expand and another 10 contract. Both expansion and contraction happen with a rate of 25%. The results, as seen in Table 1 establish that TimeRank succeeds in tracking the dynamic communities. In the case of NOC performance is boosted with the use of MutuRank, while AOC performance is inferior to NOC. In general the connection of each image with all of its nodes in this case does not help MutuRank uncover a relative ranking for the timeframes. While NNTF performs satisfactorily, GED fails to track the communities, because it identifies some expanded or contracted communities as new ones, instead of adjusted versions of the ones it has previously encountered.

Table 1. Tables for Expand/Contract and Hide/Appear datasets

Method	Expand/Contract					Hide/Appear				
	NMI	Omega	Prec	Rec	F1	NMI	Omega	Prec	Rec	F1
TR-AOC-U	0.866	0.874	0.905	0.882	0.893	1.000	1.000	1.000	1.000	1.000
TR-NOC-U	0.908	0.919	0.944	0.921	0.933	0.880	0.890	0.912	0.963	0.937
TR-AOC	0.849	0.864	0.890	0.883	0.886	1.000	1.000	1.000	1.0000	1.000
TR-NOC	0.923	0.954	0.964	0.944	0.953	0.910	0.918	0.935	0.963	0.949
NNTF	0.805	0.8445	0.842	0.864	0.853	1.000	1.000	1.000	1.000	1.000
GED	0.464	0.659	0.924	0.572	0.707	0.531	0.700	0.901	0.662	0.763

² <https://github.com/isaranto/omega-index>.

³ https://pypi.python.org/pypi/omega_index/.

⁴ <https://github.com/isaranto/community-tracking>.

⁵ <https://github.com/NightmareNyx/CommunityTracking>.

Table 2. Values for \mathbf{q} distribution for the *Expand/Contract* and *Hide/Appear* datasets

Method	Expand/Contract					Hide/Appear				
	t_1	t_2	t_3	t_4	t_5	t_1	t_2	t_3	t_4	t_5
TR-AOC	0.212	0.196	0.198	0.196	0.199	0.214	0.189	0.199	0.191	0.207
TR-NOC	0.217	0.189	0.191	0.196	0.207	0.218	0.183	0.193	0.193	0.213

The output of NNTF does not support dynamic communities that are made up of communities whose composition differs in different timeframes. The composition of communities is obtained from the factors **A** and **B**, and factor **C** declares the timeframes in which each community exists, thus defining dynamic communities that comprise of identical communities in different timeframes. With the above taken into consideration it seems as the Hide/Appear dataset , with its evaluation results presented in 1 is tailored to the NNTF method, as communities in this dataset have exactly the same composition across time, but 10% of them disappear in some timeframe and appear again in another timeframe. TimeRank AOC captures these changes because it connects with all the images of a node, thus being able to place the same community in the corresponding dynamic one. TimeRank NOC on the other hand, although it achieves high performance, it does not provide such strong connections between the node images as in AOC, hence it can misplace nodes in datasets with behaviour similar to this one. In all the changes GED identifies death events in the case where a community is hidden introducing a new dynamic community when the community appears again, accompanied by a birth event.

Reddit Each table showcases the results in the basic (4 timeframes) and extended setting (8 timeframes) which tested the scalability of the methods.

Experiment 1 This experiment has 17 different-sized communities with temporal behaviours such as delayed birth, death, non-uniform size distribution across time, overlaps, death and re-birth. MutuRank-oriented TimeRank algorithms ended up with non-uniform weights and scoring marginally lower than uniform versions. Meanwhile, GED takes the lead managing to detect the sequential behaviour of the communities in the timeframe, especially the birth/death incidents. In the 8 timeframes setting, GED couldn't keep up and lowered its scores making TR-NOC the winner, which was better at differentiating between deaths/births and expands/contracts in the long run. Results in Table 3 below. At this point we can mention that MutuRank cause a significant drop in execution time performance, being more than 100 times slower than TimeRank with uniform q distributions.

Table 3. Experiment 1 results for 4 and 8 timeframes

Method	4 Timeframes					8 Timeframes				
	NMI	Omega	Prec	Rec	F1	NMI	Omega	Prec	Rec	F1
TR-AOC-U	0.385	0.316	0.557	0.56	0.558	0.319	0.243	0.547	0.543	0.545
TR-NOC-U	0.48	0.428	0.639	0.619	0.629	0.377	0.455	0.612	0.580	0.596
TR-AOC	0.390	0.373	0.576	0.605	0.591	0.295	0.217	0.531	0.521	0.526
TR-NOC	0.457	0.473	0.633	0.623	0.628	0.435	0.537	0.610	0.654	0.631
NNTF	0.447	0.496	0.627	0.642	0.634	0.395	0.480	0.590	0.650	0.619
GED-T	0.584	0.776	1.000	0.625	0.769	0.323	0.432	1.000	0.377	0.548

Experiment 2 Here we have three big communities that change size in each timeframe with minor overlap. Results in Table 4 show that it was an easy win for TR-NOC in the 4-timeframe setting, which employed MutuRank correctly catching the underlying temporal structures. Interestingly, the tables are turned when four more timeframes are added, where everything fails except for NNTF, which scores much higher than before, due to the small number communities that express membership homogeneity through this longer period.

Table 4. Experiment 2 results for 4 and 8 timeframes

Method	4 Timeframes					8 Timeframes				
	NMI	Omega	Prec	Rec	F1	NMI	Omega	Prec	Rec	F1
TR-AOC-U	0.050	0.068	0.480	0.544	0.510	0.028	-0.007	0.438	0.438	0.438
TR-NOC-U	0.380	0.487	0.692	0.905	0.784	0.032	0.032	0.435	0.536	0.480
TR-AOC	0.038	0.032	0.471	0.542	0.500	0.028	-0.007	0.438	0.439	0.439
TR-NOC	0.456	0.604	0.741	0.951	0.833	0.031	0.030	0.435	0.536	0.480
NNTF	0.276	0.389	0.723	0.652	0.686	0.637	0.772	0.849	0.933	0.890
GED-T	0.210	0.280	1.000	0.269	0.424	0.099	0.147	1.000	0.132	0.233

Experiment 3 This experiment contains 20 different sized communities that have big overlaps. Unsurprisingly, it proved to be the hardest one so far, especially for TimeRank, as the low scores indicate in Table 5 below. At this point we tried using the Fuzzy C-Means algorithm instead of K-Means in the Spectral Clustering process of TimeRank, to achieve an overlapping version of TimeRank and also used the overlapping version of NNTF. We tested these versions in the first 4 timeframes that showed the biggest overlap. Neither the overlapping versions nor the simple ones managed to score high; former losing points from high overlap generosity and the latter from the inability to predict overlaps. Moreover, the FCM clustering on the spectral embedding returned very uniform assignment matrices which led to predicting very overlapping communities. The highest scores came from GED and simple NNTF, except for the BCubed-Recall in which NNTF-Overlap scored relatively high, because of the

high amount of true positives. TimeRank’s overlapping versions scored poorly and are not included.

Table 5. Experiment 3 results for 4 and 8 timeframes

Method	4 Timeframes					8 Timeframes				
	NMI	Omega	Prec	Rec	F1	NMI	Omega	Prec	Rec	F1
TR-AOC-U	0.043	0.091	0.241	0.512	0.327	0.028	0.029	0.198	0.431	0.271
TR-NOC-U	0.056	0.075	0.256	0.501	0.338	0.080	0.088	0.227	0.566	0.324
TR-AOC	0.044	0.094	0.243	0.520	0.331	0.029	0.031	0.199	0.431	0.272
TR-NOC	0.071	0.114	0.264	0.495	0.345	0.053	0.086	0.228	0.420	0.296
NNTF	0.277	0.410	0.551	0.451	0.496	0.170	0.344	0.582	0.386	0.464
GED-T	0.277	0.398	1.000	0.307	0.470	0.088	0.234	1.000	0.178	0.302
NNTF-Ovlp	0.109	0.134	0.171	0.859	0.286					

6 Conclusions and Future Work

In this work we proposed TimeRank, with its two variations, as a new one-step method that performs both tracking and community detection in dynamic networks in one step. TimeRank represents a dynamic network as a tensor and then performs a random walk on the tensor in order to efficiently expose dynamic community structure. We compared TimeRank with two other methods. There seems to be no universal solution that can guarantee the best outcome. The GED method, could be characterised as a local method, as it can only discover changes between sequential timeframes, while NNTF is more suitable when communities show a membership homogeneity across time. On the other hand, TimeRank is robust to disruptions of homogeneity and is able to discover non-local temporal structure.

Future work involves experiments with more datasets. To speed up execution time we could relax the probabilities’ estimation by adopting the idea from [17]. Finally, the inter-timeframe edges in AOC could have weights, proportional to the distance in terms of timeframes.

References

1. Amigó, E., Gonzalo, J., Artiles, J., Verdejo, F.: A comparison of extrinsic clustering evaluation metrics based on formal constraints. *Inf. Retr.* **12**(4), 461–486 (2009)
2. Appel, A.P., Cunha, R.L., Aggarwal, C.C., Terakado, M.M.: Temporally evolving community detection and prediction in content-centric networks. [arXiv:1807.06560](https://arxiv.org/abs/1807.06560) (2018)
3. Araujo, M., Papadimitriou, S., Günemann, S., Faloutsos, C., Basu, P., Swami, A., Papalexakis, E.E., Koutra, D.: Com2: fast automatic discovery of temporal (‘comet’) communities. In: Pacific-Asia Conference on Knowledge Discovery and Data Mining, pp. 271–283. Springer (2014)

4. Asur, S., Parthasarathy, S., Ucar, D.: An event-based framework for characterizing the evolutionary behavior of interaction graphs. *ACM Trans. Knowl. Discov. Data (TKDD)* **3**(4), 16 (2009)
5. Blondel, V.D., Guillaume, J.L., Lambiotte, R., Lefebvre, E.: Fast unfolding of communities in large networks. *J. Stat. Mech. Theory Exp.* (10), P10,008 (2008)
6. Bródka, P., Saganowski, S., Kazienko, P.: Ged: the method for group evolution discovery in social networks. *Soc. Netw. Anal. Mining* **3**(1), 1–14 (2013)
7. Fortunato, S.: Community detection in graphs. *Phys. Rep.* **486**(3–5), 75–174 (2010)
8. Gauvin, L., Panisson, A., Cattuto, C.: Detecting the community structure and activity patterns of temporal networks: a non-negative tensor factorization approach. *PloS One* **9**(1), e86,028 (2014)
9. Gliwa, B., Saganowski, S., Zygmunt, A., Bródka, P., Kazienko, P., Kozak, J.: Identification of group changes in blogosphere. In: 2012 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM), pp. 1201–1206. IEEE (2012)
10. Goldberg, M.K., Magdon-Ismail, M., Nambirajan, S., Thompson, J.: Tracking and predicting evolution of social communities. In: SocialCom/PASSAT, pp. 780–783. Citeseer (2011)
11. Greene, D., Doyle, D., Cunningham, P.: Tracking the evolution of communities in dynamic social networks. In: Advances in Social Networks Analysis and Mining, pp. 176–183. IEEE (2010)
12. Harshman, R.A.: Parafac: an “explanatory” factor analysis procedure. *J. Acoust. Soc. Amer.* **50**(1A), 117–117 (1971)
13. Kleinberg, J.M.: Authoritative sources in a hyperlinked environment. *J. ACM (JACM)* **46**(5), 604–632 (1999)
14. Lancichinetti, A., Fortunato, S., Kertész, J.: Detecting the overlapping and hierarchical community structure in complex networks. *New J. Phys.* **11**(3), 033,015 (2009)
15. Lancichinetti, A., Fortunato, S., Radicchi, F.: Benchmark graphs for testing community detection algorithms. *Phys. Rev. E* **78**(4), 046,110 (2008)
16. Murray, G., Carenini, G., Ng, R.: Using the omega index for evaluating abstractive community detection. In: Proceedings of Workshop on Evaluation Metrics and System Comparison for Automatic Summarization, pp. 10–18. Association for Computational Linguistics (2012)
17. Ng, M.K.P., Li, X., Ye, Y.: Multirank: co-ranking for objects and relations in multi-relational data. In: Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 1217–1225. ACM (2011)
18. Page, L., Brin, S., Motwani, R., Winograd, T.: The pagerank citation ranking: bringing order to the web. Technical Report, Stanford InfoLab (1999)
19. Palla, G., Barabási, A.L., Vicsek, T.: Quantifying social group evolution. *Nature* **446**(7136), 664 (2007)
20. Tajeuna, E.G., Bouguessa, M., Wang, S.: Tracking the evolution of community structures in time-evolving social networks. In: IEEE International Conference on Data Science and Advanced Analytics (DSAA). 36678 2015, pp. 1–10. IEEE (2015)
21. Takaffoli, M., Fagnan, J., Sangi, F., Zaïane, O.R.: Tracking changes in dynamic information networks. In: 2011 International Conference on Computational Aspects of Social Networks (CASON), pp. 94–101. IEEE (2011)
22. Wu, Z., Cao, J., Zhu, G., Yin, W., Cuzzocrea, A., Shi, J.: Detecting overlapping communities in poly-relational networks. *World Wide Web* **18**(5), 1373–1390 (2015)

23. Yang, J., Leskovec, J.: Overlapping community detection at scale: a non negative matrix factorization approach. In: Proceedings of the Sixth ACM International Conference on Websearch and Data Mining, pp. 587–596. ACM (2013)

Diffusion and Epidemics

Modeling Topical Information Diffusion over Microblog Networks

Kuntal Dey^{1(✉)}, Hemank Lamba², Seema Nagar¹, Shubham Gupta³, and Saroj Kaushik⁴

¹ IBM Research, Delhi, New Delhi, India

{kuntadey, senagar3}@in.ibm.com

² Carnegie Mellon University, Pittsburgh, USA

hlambda@cs.cmu.edu

³ IIIT, Delhi, New Delhi, India

shubhamgupta6531@gmail.com

⁴ Indian Institute of Technology, Delhi, India

saroj@cse.iitd.ac.in

Abstract. Traditional information spread and activation models on social networks, fail to take user interests towards specific content (topics) into account. To this, we propose a predictive topical spreading activation model (*TopSPA*). Following cues from the well-known spreading activation (*SPA*) model, we design the *TopSPA* algorithm to include the affinity of users to given topics. *TopSPA* utilizes the social connection structures of users, along with their topic affinities, to model the information flow. We use topic-based skew in energy seeding and energy propagation resistance in the network to form our overall information diffusion model. We empirically validate our model on multiple social event datasets on Twitter, predicting information diffusion over the social graph with a high accuracy.

1 Introduction

Information spreading has emerged as one of the key focus research areas within the paradigm of social network analysis. Several research studies such as [8, 22, 25] have investigated information diffusion on social media, including microblogs like Twitter. Related studies such as [1, 6, 13, 16] have been conducted to find influence and infection spreading, and applications, on social networks.

Spreading Activation (SPA), first proposed by [7], is one of the network graph based analysis models, that has been successful in capturing information spread path on the network. SPA has been accepted as an effective model of information diffusion on social networks, in a topic-independent setting. Unfortunately, much of the existing literature on spread and activation, attempts to investigate the spread of information on social media in a topic-agnostic manner. These models ignore the fact that not every vertex (representing one individual) in the network graph will have equal affinity to (representing/interest towards) all types of information. As a result, no spread and activation model takes the user interest in topic as an ingredient to the information diffusion process.

However, algorithms, like the traditional SPA, ignore the rich information already present in the tweet content, specifically, the likelihood of a given user attempting to spread information based upon her affinity to a given topic. To address this, we propose *TopSPA*, a topical spreading activation model, where energy is diffused over the online social networks, across the edges connecting the individuals (social connections). Our model can factor both the social connection patterns and strength, as well as the topical content of the information, that would diffuse over the social network. Thus, *TopSPA* is designed to capture this inherent nature of real-life microblog communications.

In our *TopSPA* model, we assign seeding energy, based upon the topic affinity of the vertices belonging to the graph. This energy diffuses, from the energized (*activated*) nodes, over the network edges over multiple steps, thereby spreading out information. The diffusion process continues, until the energy diffusion process subsidizes, and the network becomes stable again. Further, the *TopSPA* algorithm ensures that, at each step of the energy diffusion process, the energy dissipation factor of any given vertex, remains proportionate to its known topic affinity at that point of time.

We validate *TopSPA* using ground truth Twitter data, across multiple datasets. Our model yields significant prediction accuracy of information diffusion, and perform more effectively than traditional SPA as seen in our experiments.

In summary, our contributions in this work are the following.

1. We propose *TopSPA*, a topic diffusion model based upon the traditional spreading activation model (SPA), to model and predict information flow over microblogs.
2. We incorporate topic affinity based initial energies in *TopSPA*.
3. We improve the constant energy dissipation model used for traditional algorithms, including SPA, by a topic-specific dynamic energy dissipation factor for each given vertex for each given topic, proportionate to its known affinity to a given topic, at any point of time.
4. We empirically validate our model on multiple real life Twitter datasets, and observe significant prediction accuracy, comparing with ground truth.

2 Related Work

Information diffusion over topics and influence has been an area of academic interest for some time. We highlight some key literature in this space below.

One body of work attempts to study influence propagation on social networks. Nguyen et al. [21] study the contexts of social influence on online social networks, and showcase on the Digg (<http://digg.com>) social networking and news dataset. Tang et al. [23] investigate the impact of topics on social influence in large-scale networks, validating on an academic collaboration network (<http://arxiv.org>), and a film-director-actor-writer network (http://en.wikipedia.org/wiki/Category:English-language_films). Some other influence propagation works include [14, 23].

Research works also study online information flow. Wu et al. [24] model dynamic multi-topic discussions on online forums, using ParticipationRank to rank users as per their social importance and participation in given topics. Kuo et al. [12] use latent information to predict diffusion of novel topics on social networks, in absence of historical data. Jiang et al. [11] employs a game theoretic framework to model the dynamic information diffusion process in social networks. Bourigault et al. [5] embed social network

users in a latent space to obtain diffusion probabilities, using an independent cascade model. Halberstam et al. [10] study the diffusion of political information in social networks, with the role of homophily in disproportionate access to like-minded information. Other recent related works propose topic-sensitive diffusion models [2] and measure the impact of topic on diffusion [9]. Some other related works include [1, 4, 16, 20].

Romero et al. [22] studies specific kind of information diffusion, namely diffusion of idioms, hashtags and complex contagion on Twitter. Nagar et al. [18] demonstrate how content flow occurs during natural disasters on Twitter. Topical discussions have been shown to flow over social connections over unstructured microblogs like Twitter [19]. Further, topical discussions have also shown to evolve well within geographical boundaries [17].

As we observe, none of the prior works attempt to model information diffusion by modeling the topic-specific affinity for initial energization, or perform topic-specific energy dissipation based topical information diffusion model. This makes our work novel, and an important area of investigation.

3 Modeling the Information Flow

We explore the nature of information flow over social links, with respect to the latent topics that constitute the information. For this, we hypothesize that information pertaining to different topics will flow differently over the same social network, depending upon two primary factors.

- The degree of topical affinity (interest) of the set of users discussing on the topic.
- The social network connection pattern and strength of each of these connections.

Our objective is to model topic-specific information flow, and thereby become capable of predicting such flow.

3.1 Overview of Our Approach

We follow a two-step approach. First, from the unstructured and unlabeled text content found on the social network (microblog) dataset, we find the set of topics under discussion, using document co-occurrence. Second, using the affinity of each user with respect to each topic to energize the social network graph vertices and allow energy (information) to flow through the network, we execute a spreading activation model. We empirically validate our model using ground truth Twitter data. Figure 1 gives an overview of our system.

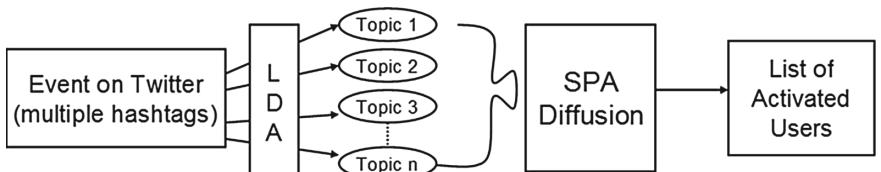


Fig. 1. Overview of our system

3.2 Finding Topics

We discover the latent topics from the unstructured and unlabeled microblog datasets, defining a topic as a cluster of frequently co-occurring concepts. We concatenate all the posts B_v made by a given user v , to form a document d_i . Therefore, we have:

$$d_i = \cup_{(\forall b_v \in B_v)} b_v \quad (1)$$

Treating $D = (\forall i \in 1 \dots n) \{d_i\}$ as the set of documents, we investigate concept co-occurrence across the set of microblog posts. We treat each set of frequently co-occurring concepts as one topic for the rest of our study. Please note that because of our earlier concatenation, each user now has a one-on-one relationship with a unique document.

Let $\{V\}$ and $\{E\}$ be the set of vertices (people) and edges (connections) in a social network (in this case, the microblog). Let $\{T\} = \{t_1, t_2, \dots, t_m\}$ be the set of topics. We assign the affinity of a user (equivalently, of a document) to each topic such that for a given user v , if $\{T_v\} = \{t_{1,v}, t_{2,v}, \dots, t_{m,v}\}$ represents the affinity of a user to each topic, then, for any given $v \in V$:

$$\sum_{l=1}^m t_{l,v} = 1 \quad (2)$$

3.3 The Native SPA Model

The traditional SPA algorithm is an iterative process, in which some vertices have non-zero initial energy values. The set of vertices with non-zero initial energy, as well as their initial energy values, is determined by the application context. Over iterations, this energy is diffused over the network using social connections. At the end of the process, many of the initially inert (zero-energy) nodes will be activated (energized) to a quantifiable degree.

The social network graph $G = (V, E)$ is represented as an $n \times n$ adjacency matrix W . Each element $w_{i,j}$ in W represents the weight of the edge (connection) $e_{i,j}$ between the pair of vertices v_i and v_j . Let the number of vertices in the graph be $|V| = n$. We normalize the adjacency matrix so that the total strength of connection to its outbound neighbors is unity. This is obtained by transforming the matrix W to another $n \times n$ matrix \hat{W} as:

$$((\forall w_{i,j} \in W)(\forall i \in \{1 \dots n\}), \hat{w}_{i,j} = \frac{w_{i,j}}{\sum_{j=1}^n (w_{i,j})}) \quad (3)$$

Clearly, $\sum(\hat{w}_{i,1}, \hat{w}_{i,2}, \dots, \hat{w}_{i,n}) = 1$. \hat{W} , the transformed matrix, is subsequently used to distribute energy proportionate to connection strengths along $\Gamma(v)$, the direct neighbors of a given vertex v .

Let the vector $P_k (= p_{k,1}, p_{k,2}, \dots, p_{k,n})$ represent the total energy accumulated by each of the vertices v_1, v_2, \dots, v_n after k iterations. Let $F_k (= f_{k,1}, f_{k,2}, \dots, f_{k,n})$

represent the distribution of the fresh energy received by all the vertices in the k^{th} iteration. Please note that, initially, $f_{0,i} = 0$ if a node is not given an initial energy, but will contain the given initial energy value otherwise. The P vector for the entire graph is initially assigned to zero (null vector). The F vector is constructed initially as the affinity of each user to the given topic.

At any iteration, a vertex will disperse a fraction of the energy freshly received in the previous iteration. Let d be the SPA energy dissipation factor, which implies that d would represent the fraction of the energy dispersed, and the remaining $(1 - d)$ fraction of energy would be retained, by a vertex, in a given iteration. The energy given out at the $(k + 1)^{th}$ iteration, by the i^{th} vertex, will be $d.f_{k,i}$. The total energy of the i^{th} vertex after the $(k + 1)^{th}$ iteration will be:

$$p_{k+1,i} = p_{k,i} + (1 - d).f_{k,i} \quad (4)$$

The fresh energy received by each vertex, F_{k+1} , is obtained by multiplying the vector F_k with the transformed social network adjacency matrix \hat{W} , and incorporating the spread factor d . The fresh energy at the i^{th} individual vertex, after $k + 1$ iterations, is:

$$f_{k+1,i} = d. \sum_{j=1}^{\Gamma(i)} (f_{k,j}. \hat{w}_{i,j}) \quad (5)$$

Therefore, the total fresh energy across all the vertices in the graph is given by:

$$F_{k+1} = d.F_k.\hat{W} \quad (6)$$

Let $V_i \subseteq V$ be the set of vertices that have been discovered till step i . The native SPA algorithm will terminate when the following conditions are satisfied:

1. No more new vertex is discovered. That is, $V_i = V_{i-1}$.
2. The maximum energy transferred in a given iteration, across all nodes, is less than a given threshold th :

$$(\forall i \in 1 \dots n)((\forall j \in \Gamma(i)) d.(f_{k,j}. \hat{w}_{i,j}) < th) \quad (7)$$

A maximum number of iterations is also specified to ensure that the algorithm stops after a finite run, in case of non-convergence.

3.4 The *TopSPA* Model

We now tailor the SPA model to adapt it for topics, and thereby create our model *TopSPA*. We apply the topics discovered in Subsect. 3.2 for (a) initial energy assignment, and (b) topic-specific energy dissipation factor for each given vertex for each given topic. Like the traditional SPA, *TopSPA* is an iterative process. The spreading activation process is carried out per topic. The universal set of topics is given. The process is carried out by learning the affinity of each user to a given topic using initial observation, and subsequently using this observation to predict the flow of topical information (energy).

Algorithm 1 *TopSPA* USER SEEDING ALGORITHM

```

1: Sort microblog posts in temporal order
2: for each topic  $t_l$  do
3:   Pick the  $r\%$  earliest users,  $\{u_{r,t_l}\}$ , to have posted on  $t_l$ 
4:   for  $i = 1 \rightarrow n$  do
5:     if  $i \in \{u_{r,t_l}\}$  then
6:        $f^t_{0,i} \leftarrow t_{l,i}$ 
7:     else
8:        $f^t_{0,i} \leftarrow 0$ 
9:     end if
10:   end for
11: end for
12: Output:  $F^t$ 
```

In *TopSPA*, we propose that the quantum of the initial energy value of a given vertex for a given topic, be proportionate to the observed initial knowledge (information) of the vertex about the given topic.

The initial energy, therefore, represents the initial affinity of the user to the topic. The intuition behind this assignment of energy assignment is that, an individual belonging to a social network with a given quantum of initial knowledge on a topic, would have a probability proportionate to spread information related to the given topic. By this process, just like in native SPA, some of the vertices will have a non-zero initial energy value, while the rest will have a zero energy value. The total energy accumulated by a given vertex at the end of the process will indicate as the predictor of the vertex (person) of having received the topical information. We execute *TopSPA* for each topic separately, in order to predict the eventual spread of the topic.

Picking seed users and assigning initial energies

We now pick the seed users in order to start the *TopSPA* process. Further, we assign non-zero initial energy to the seed users, at a per-topic level. We sort the microblog posts by timestamp, in order to ensure that the messages from the set of users with knowledge of the topic under consideration appear earlier, compared to those who obtained the knowledge temporally later by means of information diffusion over the network. For each topic, we now pick the $r\%$ earliest users, $\{u_{r,t_l}\}$, to have made a post, in temporal order. For experiments, we pick the 30% earliest users. We use the known overall affinity of the chosen users, to a given topic, as their initial energy for that topic, to execute the *TopSPA* process. This corresponds to the *observed* ground truth data. The rest of the users are assigned an initial energy of zero, and acquire energy by the *TopSPA* energy flow process. These users, the *unseen* energy (topical information) recipients in the algorithm, are subsequently used for validating *TopSPA*, at the end of the process. The seeding process is described in Algorithm 1.

The end-to-end *TopSPA* unfolds as a deep embedding of the *TopSPA* seeding algorithm, into the native SPA algorithm. This is given in Algorithm 2.

Algorithm 2 *TopSPA* END-TO-END ALGORITHM

-
- 1: Invoke the *TopSPA* User Seeding Algorithm
 - 2: Pass the obtained F^t values into the Native SPA algorithm
 - 3: Within the Native SPA algorithm, replace F_0 with F^t
 - 4: Run the Native SPA algorithm with the modified input, to obtain P^t
 - 5: **Output:** P^t
-

Assigning vertices with topic-specific energy dissipation factors

In the *TopSPA* model, we dynamically assign a topic-specific energy dissipation factor to each vertex. We model the behavior of a real-life human being in the following manner. Initially, when a human being comes to know of some information, they tend to pass it on, subject to their conviction and topic affinity. Consequently, stronger incoming messages will get passed around significantly, and topics aligned to the user's interests will tend to get passed more than others. On the other hand, as the information keeps re-arriving to a person, the tendency to keep forwarding the same message dampens, and so does its impact. Further, human beings would, inherently, tend to pass a small amount of information irrespective of topical interest.

For example, if an individual hears a breaking news for the first time, they will tend to echo it, thereby passing information to their social network neighbors. As they keep hearing more about it, they would still echo it, for some more time. But eventually, if they repeat hearing the same message (from the same or another source), they will not keep propagating the message perpetually. The tendency to re-propagate will gradually dampen over time.

We factor for all the above to construct the dynamic dissipation factor, d_{v_i, t_l, p_k} , for a given user, for a given topic, at a given stage. Let α_{v_i} represent the inherent tendency of v_i to pass information irrespective of the topic. Let β represent the scale parameter. Let p_k be the total energy of the vertex, with respect to t_l , at a given time.

$$d_{v_i, t_l, p_k} = \alpha_{v_i} + (1 - \alpha_{v_i})e^{1 - \beta \cdot (p_k | t_l)} \quad (8)$$

To bootstrap the system, we start with the two following assumptions:

- (a) In the first iteration, the initially energized vertices will uniformly dissipate a high 50% of their energy, as these vertices are excited for the given topic *on their own* (without observable external energy push).
- (b) Also, the first time a vertex receives energy, it passes forward only proportion to its inherent tendency to pass energy (information).

Validation

We employ two methods of validation: (a) compare rank order of the ranked lists and (b) compare degree of overlap in the ranked lists. After the *TopSPA* processes for all the topics get over, each user stands to receive some energy as a result of the energy flow for each topic. Clearly, a higher amount of energy for a particular topic corresponds to higher likelihood of interest level in the topic. From the amount of energies each person accumulates in the *TopSPA* process, we construct ranked lists of people for each topic $\{RS_{t_1}, RS_{t_2}, \dots, RS_{t_m}\}$: the higher the energy, the higher the rank of a person.

We also have the topic affinities for each user computed in the process of finding topics. From these topic affinities, we compute ground truth ranked lists of interest level $\{RG_{t_1}, RG_{t_2}, \dots, RG_{t_m}\}$ for each topic.

We now compare the two ranked lists: one obtained by our energy spreading based prediction process, and the other observed from the ground truth topic affinity data at the end of the event. The correlation across these two lists would indicate the effectiveness of the *TopSPA* model.

One well-defined method to quantify the accuracy of a prediction algorithm with respect to a ground truth is the *precision-based* measurement. For prediction set $\{A\}$ and ground truth set $\{B\}$, we measure this as: $prec(A, B) = \frac{|A \cap B|}{|B|}$. In the other validation process, we compare overlaps $\{10\%, 20\%, \dots, 90\%\}$ in the ranked lists of each topic. In this process, we compute the mean Jac.

4 Experiments

Data description

Our experiments are carried out on two real-life Twitter datasets. We collect Twitter data from two well-known political turmoils in March 2011 with significant footprint on Twitter - the Libya turmoil and the Egypt turmoil. The Libya data was collected using (*Libya OR Gaddafi*) as target keyword, and that for the Egypt data was (*Egypt OR Protest*). We further collected the social network connections (followers) data for these users. Table 1 provides the dataset details.

Table 1. Details of the datasets used for our experiments

Data set	Keywords	Tweets	NumUsers
Libya	Libya, Gaddafi	1,011,716	83,177
Egypt	Egypt, Protest	60,948	37,961

Topic identification

We identify the topics as conceptually related word clusters present in the set of tweets for each dataset, using MALLET [15]. We evaluate our method at different levels of fine-granularity of topics, and thereby identify 10, 20, 30 and 40 topics respectively, for each dataset. We apply Eq. 1 to obtain documents, and assign a topic affinity value to each document, with respect to each topic identified. The topic affinity values are within the range of 0 and 1, and is assigned by the MALLET topic inferences subsystem, that uses the LDA [3] algorithm internally.

Seed users and initial energy assignment

For each dataset, we choose the top 30% of the users to have made a post as the seed users. In order to assign initial energy to start the *TopSPA* process, for each given topic, we divide the population into two segments - a *seen* (observed) segment and an *unseen* (unobserved) segment. The *seen* segment of the population comprises of the top 30% users. For this segment, we assign the ground truth topic affinity values, as observed

Table 2. Results for mean precision and Jaccard coefficient using *TopSPA*

		% Results for <i>topSPA</i>							
Data set	Topics	For mean precision				For Jaccard coefficient			
		10%	20%	30%	40%	10%	20%	30%	40%
Libya	10	0.154	0.213	0.288	0.369	0.085	0.121	0.171	0.228
	20	0.147	0.205	0.284	0.367	0.080	0.115	0.166	0.226
Egypt	10	0.263	0.268	0.318	0.402	0.152	0.155	0.189	0.252
	20	0.262	0.266	0.322	0.398	0.151	0.154	0.192	0.248

Table 3. Results for mean precision and Jaccard coefficient using traditional SPA

		% Results for SPA							
Data set	Topics	For mean precision				For Jaccard Coefficient			
		10%	20%	30%	40%	10%	20%	30%	40%
Libya	10	0.1187	0.2071	0.2846	0.3590	0.0593	0.1035	0.1423	0.1795
	20	0.1138	0.2025	0.2804	0.3569	0.0569	0.1012	0.1402	0.1784
Egypt	10	0.1078	0.1973	0.2838	0.3673	0.0539	0.0986	0.1419	0.1836
	20	0.1034	0.1913	0.2857	0.3682	0.0517	0.0956	0.1428	0.1840

by MALLET, as the initial energies, for each topic. For the other (*unseen*) segment, we start with an initial energy level of zero.

Results, validation and findings

We run the *TopSPA* process for each topic independently, and compute the energy (topical information) flow to each user in the *unseen* segment at the end of the process for each of the topics.

We now investigate the experimental results. We validate using (a) precision of our prediction of the top $k\%$ of our predictions with respect to ground truth data and (b) Jaccard's coefficient. Table 2 shows the results for precision-based correlation coefficient and Jaccard's coefficient respectively. The process yields encouraging results for each validations, for both the datasets under consideration.

Improvements Over Traditional SPA

We also run traditional SPA on the two datasets to compare our results. For each dataset, we choose the early 30% of the users who tweeted, as the seed users. We sample initial energies for these users from a normal distribution. We then let the standard SPA algorithm run. In order to validate base SPA results we employed the same techniques as we did for the *TopSPA* model. For constructing ground truth ranked list, we assign ground truth energy to a node as the mean of it's topic affinities observed by MALLET.

Figures 2(a) and 2(b) show the comparison of precision-based correlation coefficient for *TopSPA* and SPA model. We observe that for both the datasets, precision is always higher for *TopSPA* model. Figures 2(c) and 2(d) show the comparison of Jaccard's coefficient for *topSPA* and SPA model. Table 3 shows the results for precision-based correlation coefficient and Jaccard's coefficient respectively.

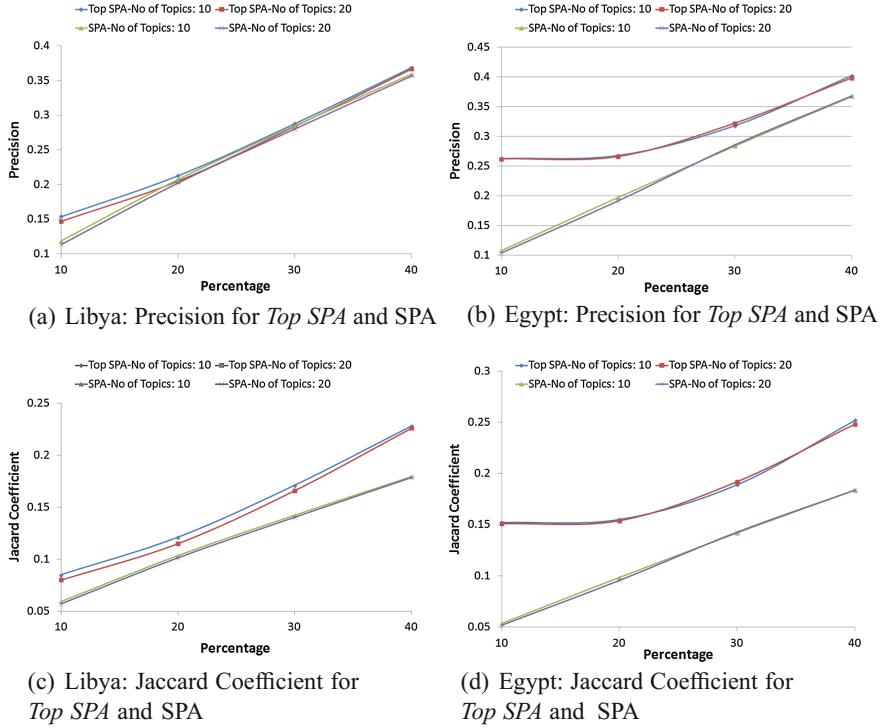


Fig. 2. Comparing the performances of *Top SPA* and SPA

In order to compare the relative performances of traditional SPA algorithm and the proposed *TopSPA* algorithm, we compare Table 2 with Table 3. We observe that for both the datasets, *TopSPA* model is always outperforming traditional SPA model. More often than not, *TopSPA* outperforms traditional SPA significantly, which is clearly reflected by comparing the values of these tables. For example, for Egypt, for top 10% mean value of Jaccard's coefficient as well as precision is more than 2.53 times for *TopSPA* compare to traditional SPA. Further, the trend of *TopSPA* outperforming traditional SPA is consistent. Across the two pairs of tables, there exists no instance where traditional SPA could match the performance of *TopSPA*. This justifies our research hypothesis, proving that information diffusion on microblog networks such as Twitter can be better predicted by using *TopSPA* model, compared to the well-established SPA model.

5 Conclusion

This work modeled the flow of topical information on social networks. We proposed the topical spreading activation (*TopSPA*) model, applicable on content-based social networks. We seeded the initial energies into the system, using the affinities of users to the topic under investigation, for each topic in the given set of topics. The information flow was modeled as energy transfer over the user network graph. The flow was

also subjected to topical affinity of users, implying topic-biased skew in energy propagation resistance. We validated our model using multiple real life microblog datasets from Twitter, and using different topic distribution granularities. Our work is useful for creating applications such as the ones requiring viral flow of information on social networks and media, such as spreading of rumors and viruses across the network. In future, we propose investigating other well-known network cascading models, such as the linear and independent cascade models.

References

1. Bakshy, E., Hofman, J.M., Mason, W.A., Watts, D.J.: Everyone's an influencer: quantifying influence on twitter. In: WSDM, pp. 65–74. ACM (2011)
2. Barbieri, N., Bonchi, F., Manco, G.: Topic-aware social influence propagation models. In: ICDM (2012)
3. Blei, D.M., Ng, A.Y., Jordan, M.I.: Latent dirichlet allocation. *J. Mach. Learn. Res.* **3**, 993–1022 (2003)
4. Bourigault, S., Lagnier, C., Lamprier, S., Denoyer, L., Gallinari, P.: Learning social network embeddings for predicting information diffusion. In: WSDM, pp. 393–402. ACM (2014)
5. Bourigault, S., Lamprier, S., Gallinari, P.: Representation learning for information diffusion through social networks: an embedded cascade model. In: WSDM, pp. 573–582. ACM (2016)
6. Cha, M., Haddadi, H., Benevenuto, F., Gummadi, P.K.: Measuring user influence in twitter: the million follower fallacy. *ICWSM* **10**, 10–17 (2010)
7. Dasgupta, K., Singh, R., Viswanathan, B., Chakraborty, D., Mukherjea, S., Nanavati, A.A., Joshi, A.: Social ties and their relevance to churn in mobiletelecom networks. In: EDBT, pp. 668–677. ACM (2008)
8. Fei, H., Jiang, R., Yang, Y., Luo, B., Huan, J.: Content based social behavior prediction: a multi-task learning approach. In: CIKM, pp. 995–1000 (2011)
9. Grabowicz, P.A., Ganguly, N., Gummadi, K.P.: Distinguishing between topical and non-topical information diffusion mechanisms in social media. In: ICWSM, pp. 151–160 (2016)
10. Halberstam, Y., Knight, B.: Homophily, group size, and the diffusion of political information in social networks: Evidence from twitter. *J. Public Econ.* **143**, 73–88 (2016)
11. Jiang, C., Chen, Y., Liu, K.R.: Evolutionary dynamics of information diffusion over social networks. *IEEE Trans. Signal Process.* **62**(17), 4573–4586 (2014)
12. Kuo, T.T., Hung, S.C., Lin, W.S., Peng, N., Lin, S.D., Lin, W.F.: Exploiting latent information to predict diffusions of novel topics on social networks. In: ACL (2012)
13. Kwak, H., Lee, C., Park, H., Moon, S.: What is twitter, a social media or a news media. In: Proceedings of the WWW (2010)
14. Liu, L., Tang, J., Han, J., Jiang, M., Yang, S.: Mining topic-level influence in heterogeneous networks. In: CIKM (2010)
15. McCallum, A.K.: Mallet: a machine learning for language toolkit. <http://mallet.cs.umass.edu> (2002)
16. Myers, S.A., Zhu, C., Leskovec, J.: Information diffusion and external influence in networks. In: SIGKDD, pp. 33–41 (2012)
17. Nagar, S., Narang, K., Mehta, S., Subramaniam, L., Dey, K.: Topical discussions on unstructured microblogs: analysis from a geographical perspective. In: WISE (2013)
18. Nagar, S., Seth, A., Joshi, A.: Characterization of social media response to natural disasters. In: Proceedings of the WWW (2012)

19. Narang, K., Nagar, S., Mehta, S., Subramaniam, L.V., Dey, K.: Discovery and analysis of evolving topical social discussions on unstructured microblogs. In: ECIR (2013)
20. Nematzadeh, A., Ferrara, E., Flammini, A., Ahn, Y.Y.: Optimal network modularity for information diffusion. *Phys. Rev. Lett.* **113**(8), 088,701 (2014)
21. Nguyen, J.H., Hu, B., Gnnemann, S., Ester, M.: Finding contexts of social influence in online social networks. In: The 7th SNA-KDD Workshop—SNA-KDD’13 (2013)
22. Romero, D.M., Meeder, B., Kleinberg, J.: Differences in the mechanics of information diffusion across topics: idioms, political hashtags, and complex contagion on twitter. In: WWW, pp. 695–704 (2011)
23. Tang, J., Wu, S., Sun, J.: Confluence: Conformity influence in large social networks. In: KDD (2013)
24. Wu, H., Bu, J., Chen, C., Wang, C., Qiu, G., Zhang, L., Shen, J.: Modeling dynamic multi-topic discussions in online forums. In: AAAI Conference on Artificial Intelligence (2010)
25. Yang, J., Leskovec, J.: Modeling information diffusion in implicit networks. In: ICDM, pp. 599–608 (2010)



Fast Variables Determine the Epidemic Threshold in the Pairwise Model with an Improved Closure

István Z. Kiss^{1(✉)}, Joel C. Miller², and Péter L. Simon^{3,4}

¹ Department of Mathematics, School of Mathematical and Physical Sciences,
University of Sussex, Falmer, Brighton BN1 9QH, UK

i.z.kiss@sussex.ac.uk

² Institute for Disease Modeling, Bellevue, WA, United States of America
joel.c.miller.research@gmail.com

³ Institute of Mathematics, Eötvös Loránd University, Budapest, Hungary
simonp@math.elte.hu

⁴ Numerical Analysis and Large Networks Research Group, Hungarian Academy of Sciences, Budapest, Hungary

Abstract. Pairwise models are widely used to model epidemic spread on networks. This includes the modelling of susceptible-infected-removed (SIR) epidemics on regular networks and extensions to SIS dynamics and contact tracing on more exotic networks exhibiting degree heterogeneity, directed and/or weighted links and clustering. However, extra features of the disease dynamics or of the network lead to an increase in system size and analytical tractability becomes problematic. Various “closures” can keep the system tractable. Focusing on SIR epidemics on regular but clustered networks, we show that even for the most complex closure we can determine the epidemic threshold as an asymptotic expansion in terms of the clustering coefficient. We do this by exploiting the presence of a system of fast variables, specified by the correlation structure of the epidemic, whose steady state determines the epidemic threshold. While we do not find the steady state analytically, we create an elegant asymptotic expansion of it. We validate this new threshold by comparing it to the numerical solution of the full system and find excellent agreement over a wide range of values of the clustering coefficient, transmission rate and average degree of the network. The technique carries over to pairwise models with other closures [1], and we note that the epidemic threshold will be model dependent. This emphasises the importance of model choice when dealing with realistic outbreaks.

Keywords: Epidemic · Network · Clustering · Threshold · Pairwise model

1 Introduction

One way to deal with the challenges of modelling stochastic epidemics on networks is to use mean-field models. This approach has led to a number of models including heterogeneous or degree-based mean-field [17, 18], pairwise [8, 19], effective-degree [11], edge-based compartmental [14] and message passing [6], to name a few. The main difference between these models is how the variables over which averaging is done are chosen. Perhaps the most compact model is the edge-based compartmental model [15] and this works for heterogeneous networks with Markovian SIR epidemics, although extensions of it for arbitrary infection and recovery processes are also possible [23].

Pairwise models are popular and the first model for regular networks and SIR epidemics [8, 19] was generalised to heterogeneous networks [3], preferentially mixing networks [3], directed [22] and weighted networks [20], adaptive networks [9], and structured networks [4] among others. Its wide use is perhaps due to its relative transparency where variables are defined in a straightforward way. A downside of the pairwise models is that in constructing them we find that the change in the expected number of individual nodes of a given state depends on to the expected number of edges (or pairs) between nodes of various states. The change in the expected number of edges depends on larger-scale structure. To keep the system tractable, we generally make a “closure assumption” that we can express the frequency of the relevant larger-scale structures in terms of the pairs and individuals, that is, in terms of the lower order moments or structure.

A basic understanding of the network and epidemic dynamics coupled with careful bookkeeping and an appropriate closure assumption produces a pairwise model. Pairwise models have been successfully used to analytically derive the epidemic threshold and final epidemic size. However, these results are mostly limited to networks without clustering. The propensity of contacts to cluster, i.e. that two friends of an individual/node are also friends of each other, is known to lead to many complications, and modelling epidemics on clustered networks using analytically tractable mean-field models is still limited to networks with very specific structural features [4, 7, 12, 13, 16, 21, 25]. However, using approaches borrowed from percolation theory [13] and focusing more on the stochastic process itself [24], some results have been obtained.

For pairwise models, clustering first manifests itself by requiring a different and more complex closure, which makes the analysis of the resulting system, even for regular networks and SIR dynamics, challenging. Furthermore, it turns out that such closures may in fact fail to conserve pair-level relations and may not accurately reflect the early growth of quantities such as closed loops of three nodes with all nodes being infected [5]. Such considerations have led to an improved closure being developed in an effort to keep as many true features of the exact epidemic process as possible [5]. In this paper we will focus on the classic pairwise model for regular networks with clustering but using the improved closure of [5], given below in Eq. (8). We will show that by working with fast variables corresponding to the correlations that develop during the

spread of the epidemic, we can analytically determine the epidemic threshold as an asymptotic expansion in terms of the clustering coefficient.

The use of fast variables is not completely new. They were used in [8] and [2] but not with the improved closure. Even with the simpler closures, the epidemic threshold has only been obtained numerically and it was framed in terms of a growth-rate-based threshold (which is equivalent to the basic reproduction number at the critical point of the epidemic spread). In [2] a hybrid pairwise model incorporating random and clustered contacts is considered, with the analysis focused on the growth-rate-based threshold. The authors of [2] managed to derive a number of results, some analytic (the critical clustering coefficient for which an epidemic can take off) and some semi-analytic, and they have shown, in agreement with most studies, that clustering inhibits the spread of the epidemic when compared to an equivalent network without clustering but with equivalent parameter values governing the epidemic process. However, no analytic expression for the threshold was provided. More recently, in [10], the epidemic threshold in a pairwise model for clustered networks with closures based on the number of links in a motif, rather than nodes, was calculated.

Building on these results and the recent paper by Barnard et al. [1] (where the idea of fast variables was used to derive and analytic epidemic threshold for pairwise models with two different closures corresponding to clustered networks) we set out to take the final step of using fast variables and perturbation theory to determine an asymptotic expansion of the epidemic threshold when the pairwise model is closed with Eq. (8). The paper is structured as follows. In Sect. 2 we outline the model. The main results, both analytical and numerical, are presented in Sect. 3. We conclude with a discussion of the results and possible extensions in Sect. 4.

2 Model Formulation

2.1 The Network and Standard SIR Dynamics

We begin by considering a population of N individuals and describe their contact structure by an undirected network with adjacency matrix $G = (g_{ij})_{i,j=1,2,\dots,N}$ where $g_{ij} = 1$ if nodes i and j are connected and zero otherwise. Because the network is undirected, $g_{ij} = g_{ji}$ for all $i, j = 1, 2, \dots, N$, and because we exclude self-loops, $g_{ii} = 0$ for all i . The network is static and regular, such that each individual has exactly n edges or links. The sum over all elements of G is defined as $\|G\| = \sum_{i,j} g_{ij}$. Hence, the number of doubly counted links in the network is $\|G\| = nN$. More importantly, using simple matrix operations on G , we can calculate the clustering coefficient of the network

$$\phi = \frac{\text{trace}(G^3)}{\|G^2\| - \text{trace}(G^2)}, \quad (1)$$

where $\text{trace}(G^3)$ yields six times the number of closed triples or loops of length three (uniquely counted) and $\|G^2\| - \text{trace}(G^2)$, twice the number of triples (open and closed, also uniquely counted).

Let us consider the standard SIR epidemic dynamics on a network. The dynamics are driven by two processes: (a) infection and (b) recovery from infection. Infection can spread from an infected/infectious node to any of its susceptible neighbours. We model this as a Poisson point process with per-link infection rate τ . Infectious nodes recover at constant rate γ , independently of the network, and gain permanent immunity.

2.2 The Unclosed Pairwise Model

Let A_i be 1 if the individual at node i is of type $A \in \{S, I, R\}$ and zero otherwise. Then single nodes (singles) of type A can be counted as $[A] = \sum_i A_i$, pairs of nodes (pairs) of type $A - B$ can be counted as $[AB] = \sum_{i,j} A_i B_j g_{ij}$ and triples of nodes (triples) of type $A - B - C$ can be counted as $[ABC] = \sum_{i,j,k} A_i B_j C_k g_{ij} g_{jk}$. This method of counting means that pairs are counted once in each direction, so $[AB] = [BA]$, and $[AA]$ is even. Using this notation to track singles, pairs, and triples leads to the following system of pairwise equations describing the SIR epidemic on a regular network:

$$[\dot{S}] = -\tau[SI]; \quad [\dot{I}] = \tau[SI] - \gamma[I]; \quad [\dot{R}] = \gamma[I], \quad (2)$$

$$[\dot{SS}] = -2\tau[SSI]; \quad [\dot{SI}] = \tau([SSI] - [ISI] - [SI]) - \gamma[SI], \quad (3)$$

$$[\dot{SR}] = \gamma[SI] - \tau[RSI], \quad (4)$$

$$[\dot{II}] = 2\tau([ISI] + [SI]) - 2\gamma[II], \quad (5)$$

$$[\dot{IR}] = \gamma([II] - [IR]) + \tau[RSI]. \quad (6)$$

We note that Eqs. (3)–(6) contain triples which are not defined within the entire system of Eqs. (2)–(6). Furthermore, we have chose these variables in order to be able to consistently define our fast variables later. To determine solutions of the system, we must find a way to account for these triples in terms of pairs and singles through a closure assumption. It is worth noting that this system is exact before a closure is implemented [9].

2.3 The Improved Closure and the Closed Pairwise System

The key for deriving the improved closure [5] is to split the non-clustered and clustered part of the network and to determine the propensity of a susceptible node's neighbour to be in state A (where $A \in \{S, I, R\}$), given that the susceptible node is already connected to an infected one. This can be defined as

$$p_{A|S-I} = \begin{cases} p_{A|S-I}^{uc} & \text{with probability } (1 - \phi), \\ \frac{p_{A|S-I}^c}{\sum_a p_{a|S-I}^c} & \text{with probability } \phi, \end{cases} \quad (7)$$

where $p_{A|S-I}^{uc} = \frac{[AS]}{n[S]}$, $p_{A|S-I}^c = p_{A|S-I}^{uc} C_{AI}$ and $C_{AI} = \frac{N[AI]}{n[A][I]}$. In the absence of clustering we assume that the probability the neighbour is of state A is simply given by frequency of $[AS]$ type links relative to all links emanating from

susceptible nodes, $n[S]$. If clustering is present then the probability of finding a susceptible neighbour decreases as the transitive link connects this particular neighbour to the existing infected neighbour. This means that the node is exposed to infection and its probability of remaining susceptible decreases. This effect is captured by C_{AI} which expresses how much more probable it is, compared to the random mixing case, to find a neighbour in state A given that the node is also connected to an infectious node. It is well known that epidemics are negatively correlated in the sense that we are more likely to find $I - I$ type links rather than $I - S$. Unfortunately, $p_{A|S-I}^c$ alone is not a properly defined probability. Despite this the closure resulting from it has been used although it leads to some anomalies such as non-conservation of pair-level relations. However, the normalised form of it, as in Eq. (7), leads to the improved closure [5]. Taking into account the new way of defining $p_{A|S-I}$, this yields

$$\begin{aligned}
[ASI]_c &= (1 - \phi)[ASI] + \phi[ASI] = (1 - \phi)(n - 1)[SI]p_{A|S-I}^{uc} + \phi(n - 1)[SI]\frac{p_{A|S-I}^c}{\sum_a p_{a|S-I}^c} \\
&= (1 - \phi)(n - 1)[SI]\frac{[AS]}{n[S]} + \phi(n - 1)[SI]\frac{\frac{[AS]}{n[S]}C_{AI}}{\sum_a p_{a|S-I}^c} \\
&= (1 - \phi)\frac{(n - 1)}{n}\frac{[AS][SI]}{[S]} + \phi(n - 1)[SI]\frac{\frac{[AS]}{n[S]}\frac{N[AI]}{n[A][I]}}{\sum_a \frac{[aS]}{n[aS]}\frac{N[aI]}{n[a][I]}} \\
&= (1 - \phi)\frac{(n - 1)}{n}\frac{[AS][SI]}{[S]} + \phi(n - 1)\frac{[AS][SI][IA]}{[A]\sum_a \frac{[aS][aI]}{[a]}} \\
&= (n - 1) \left((1 - \phi)\frac{[AS][SI]}{n[S]} + \phi\frac{[AS][SI][IA]}{[A]\sum_a [aS][aI]/[a]} \right), \tag{8}
\end{aligned}$$

where $a \in \{S, I, R\}$ and $[ASI]_c$ is used to distinguish this approximation from its exact equivalent.

3 Results for the Pairwise Model with the Improved Closure

Plugging Eq. (8) into the exact system (2)–(6) leads to the self-consistent system below

$$\dot{[S]} = -\tau[SI]; \quad \dot{[I]} = \tau[SI] - \gamma[I]; \quad \dot{[R]} = \gamma[I] \tag{9}$$

$$\dot{[SS]} = -2\tau[SSI]_c; \quad \dot{[SI]} = -(\tau + \gamma)[SI] + \tau[SSI]_c - \tau[ISI]_c, \tag{10}$$

$$\dot{[SR]} = \gamma[SI] - \tau[Rsi]_c, \tag{11}$$

$$\dot{[II]} = 2\tau[SI] - 2\gamma[II] + 2\tau[ISI]_c, \tag{12}$$

$$\dot{[IR]} = \gamma([II] - [IR]) + \tau[Rsi]_c, \tag{13}$$

where $[ASI]_c$ with $A \in \{S, I, R\}$ is defined in Eq. (8). The standard linear stability analysis of this system around the disease free steady state,

$$([S], [I], [R], [SS], [SI], [SR], [II], [IR]) = (N, 0, 0, nN, 0, 0, 0, 0),$$

leads to some terms such as

$$\alpha = \frac{[SI]}{[I]}, \delta = \frac{[II]}{[I]}, x = \frac{[SR]}{[R]}, y = \frac{[IR]}{[I]}. \quad (14)$$

Interestingly these terms are ill-defined since both denominators and numerators are zero at the equilibrium. However, these variables have a clear biological meaning and are related to the correlation structure of the epidemic. For example, α tracks the average number of susceptible nodes around a typical infected node, and it is well known that epidemics are negatively correlated in the sense that infected nodes are more likely to have other infected neighbours rather than susceptible ones. For example in Fig. 1, $\alpha = [SI]/[I]$ decreases quickly while $\delta = [II]/[I]$ increases reflecting our argument about the correlation above. This is a natural consequence of the spreading process and here we exploit this.

Interestingly however, the epidemic threshold can also be found in a more direct way by looking at Eq. (9). Namely, this leads to

$$[\dot{I}] = \tau[SI] - \gamma[I] = \gamma[I] \left(\frac{\tau}{\gamma} \frac{[SI]}{[I]} - 1 \right), \quad (15)$$

which clearly shows that the epidemic threshold coincides with $\bar{\mathcal{R}}_0 = \frac{\tau}{\gamma} \frac{[SI]}{[I]} = 1$. This is a growth-rate-based threshold of the epidemic and while $\bar{\mathcal{R}}_0$ is different from the basic reproduction number, they are equivalent when both are exactly one. From here, we can see that finding the threshold amounts to finding $\alpha = \frac{[SI]}{[I]}$ at time t close to zero. As we will show next, these new variables of interest are fast variables and settle quickly, even if only temporarily, to a quasi-equilibrium.

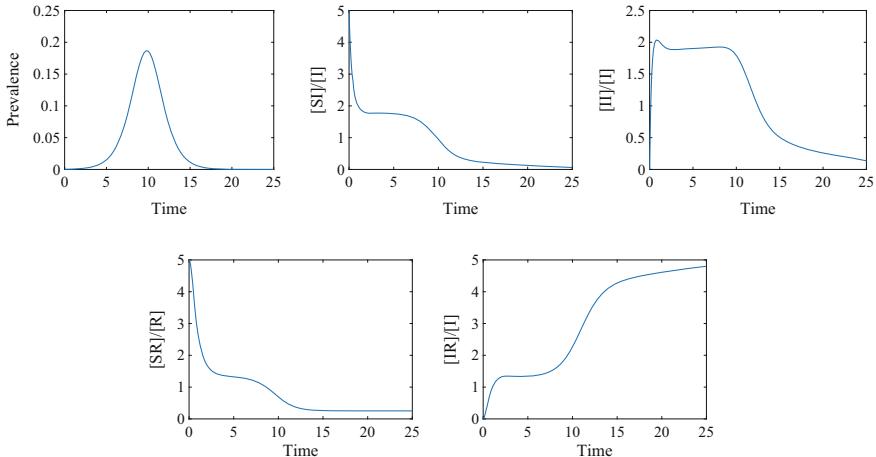


Fig. 1. Time evolution of the prevalence and four fast variables based on numerical integration of the closed pairwise system (9)–(13). Parameter values are $N = 10000$, $n = 5$, $\phi = 0.5$, $\tau = 1$ and $\gamma = 1$.

The time taken to reach this quasi-equilibrium is short compared to the timescale of epidemic growth, and the quasi-equilibrium corresponds to the exponential growth phase of the epidemic.

3.1 Fast Variables with the Improved Closure

In Fig. 1 we plot the evolution of the prevalence together with the newly defined variables. The figure shows clearly that these new variables are fast. Obviously, if the epidemic is fast, for example for high values of the transmission rate, then the time-scale separation is less clear and while the fast variables are indeed fast compared to the timescale of the epidemic, the quasi-equilibria are short lived. Namely, early on, when the prevalence is small, these variables settle to a temporary equilibrium. The natural step would be to investigate the behaviour of α (and all the others) and this can be done by deriving their evolution equations, i.e. compute $d\alpha/dt = d([SI]/[I])/dt$ and use Eqs. (9)–(13). After some simple but lengthy algebra one obtains

$$\frac{d\alpha}{dt} = -\tau\alpha(1 + \alpha) + \tau\xi(1 - \phi)n\alpha + \tau(n - 1)\phi \left(\frac{n\alpha^2 - \alpha^2\delta}{n\alpha + \alpha\delta + xy} \right), \quad (16)$$

where $\xi = (n - 1)/n$. The assumption of being close to $t = 0$ is used to neglect a term of the form $\tau\xi(1 - \phi)\frac{[SI]}{[I]}\frac{[SI]}{[S]}$, together with a few others with a similar structure. While $\alpha = [SI]/[I]$ itself is a well-defined and bounded ratio of two small numbers, $[SI]/[S] \simeq 0$ when t is close to zero. We use similar arguments when deriving the equations for the other variables. Their differential equations are

$$\frac{d\delta}{dt} = -\gamma\delta + 2\tau\alpha - \tau\alpha\delta + 2\tau(n - 1)\phi \frac{\alpha^2\delta}{n\alpha + \alpha\delta + xy}, \quad (17)$$

$$\frac{dx}{dt} = +\tau\alpha^*(\alpha - x) - \gamma(\alpha - x) - \frac{\tau(n - 1)\phi\alpha xy \left(\frac{\tau}{\gamma}\alpha^* - 1 \right)}{n\alpha + \alpha\delta + xy}, \quad (18)$$

$$\frac{dy}{dt} = +\tau\delta - \tau\alpha y + \frac{\tau(n - 1)\phi\alpha xy}{n\alpha + \alpha\delta + xy}. \quad (19)$$

As one notices the four variables are interlinked and are all needed to resolve the evolution equation of each. A key step in the derivation above is the need to introduce α^* which corresponds to the steady state of the system defined by Eqs. (16)–(19). This is needed as in the derivation of the evolution equations for x terms such as $[SI]/[R]$, $[IR]/[R]$ and $[I]/[R]$ can only be dealt with by noticing that at time t close to $t = 0$ we have that

$$[I] = \left(\frac{\tau}{\gamma}\alpha^* - 1 \right) [R]. \quad (20)$$

This follows from the observation that

$$\frac{d[I]}{d[R]} = \frac{\tau[SI] - \gamma[I]}{\gamma[I]} = \left(\frac{\tau}{\gamma} \frac{[SI]}{[I]} - 1 \right) \simeq \left(\frac{\tau}{\gamma}\alpha^* - 1 \right), \quad (21)$$

where we assumed that α stabilises quickly at small time. Integrating this from $s = 0$ to $s = t$ and assuming that $[I](0) = [R](0) = 0$ leads to $[I] = \left(\frac{\tau}{\gamma}\alpha^* - 1\right)[R] + C$, where $C = 0$ if the initial conditions at $t = 0$ are applied. This in turn allows us to write $[SI]/[R] = ([SI]/[I])([I]/[R])$ and $[IR]/[R] = ([IR]/[I])/([I]/[R])$ which ensures that we can cast all terms as functions of the four fast variables.

3.2 Asymptotic Expansion of the Epidemic Threshold

Finding the steady state of the system defined by Eqs. (16)–(19) may seem like a difficult task but it turns out that an asymptotic solution is within reach. To do this each variable v is written as $v = v_0 + \phi v_1 + \dots$, where $v \in \{\alpha, \delta, x, y\}$. Plugging these into Eqs. (16)–(19) leads to the following system at $\mathcal{O}(1)$:

$$((n-2) - \alpha_0)(n\alpha_0 + \alpha_0\delta_0 + x_0y_0) = 0, \quad (22)$$

$$(-\gamma\delta_0 + 2\tau\alpha_0 - \tau\alpha_0\delta_0)(n\alpha_0 + \alpha_0\delta_0 + x_0y_0) = 0, \quad (23)$$

$$(\delta_0 - \alpha_0y_0)(n\alpha_0 + \alpha_0\delta_0 + x_0y_0) = 0, \quad (24)$$

$$(\tau\gamma\alpha_0^2 - \tau\gamma\alpha_0x_0 - \gamma^2\alpha_0 + \gamma^2x_0)(n\alpha_0 + \alpha_0\delta_0 + x_0y_0) = 0. \quad (25)$$

One of the solutions of this system is:

$$(\alpha_0, \delta_0, x_0, y_0) = \left(n-2, \frac{2\tau(n-2)}{\gamma + \tau(n-2)}, n-2, \frac{2\tau}{\gamma + \tau(n-2)}\right). \quad (26)$$

At $\mathcal{O}(\phi)$ from Eq. (16) we have

$$-(\alpha_1 + (n-1))(n\alpha_0 + \alpha_0\delta_0 + x_0y_0) + (n-1)\alpha_0(n-\delta_0) = 0. \quad (27)$$

Plugging in the solutions at $\mathcal{O}(1)$ from Eq. (26) into this equation leads to

$$\alpha_1 = -\frac{2\tau(2n-3)(n-1)}{n(\gamma + \tau(n-2)) + 2\tau(n-1)}. \quad (28)$$

Hence the epidemic threshold, up to the first correction is given by $\overline{\mathcal{R}}_0 = 1$ where

$$\overline{\mathcal{R}}_0 = \frac{\tau}{\gamma}(\alpha_0 + \phi\alpha_1) = \frac{\tau(n-2)}{\gamma} - \frac{\tau}{\gamma} \frac{2\tau(2n-3)(n-1)}{n(\gamma + \tau(n-2)) + 2\tau(n-1)}\phi. \quad (29)$$

The first observation that can be made is that the first order correction is negative and this implies that clustering reduces the growth rate and makes the disease less able to spread. The second is that when $\phi = 0$, $\overline{\mathcal{R}}_0 = 1$ reduces to the well known threshold when a network with no clustering is considered.

3.3 Numerical Examples

In Fig. 2 we show a systematic test comparing the epidemic threshold generated via solving the closed pairwise system (9)–(13) numerically to the epidemic

threshold based on the asymptotic expansion (29), over a wide range of (τ, n) values. Several observations can be made. First, it is clear that higher values of clustering push the location of threshold to higher τ and n values, meaning that the limiting effect of clustering on the epidemic spread can only be overcome if either the value of the transmission rate or average degree increases. Second, the agreement between the numerical and asymptotic threshold is excellent for a large range of clustering values. In fact, a slight discrepancy only really seems to appear at around $\phi = 0.6$. We note that any discrepancy can be separated into error caused by the perturbative approximation and the behaviour of the fast variables. These effects are hard to separate but for the latter we note that during a fast epidemic the transients are reached quickly but they are short lived, whereas for slow epidemics they are reached slower but they live longer. Finally, it is worth noting that finding the final epidemic size numerically can be achieved by using a more compact system, e.g. focusing solely on $[S]$, $[I]$, $[SS]$, $[SI]$ and $[II]$. However, the extended system is preferred here since the derivation of the system of fast variables relies upon it.

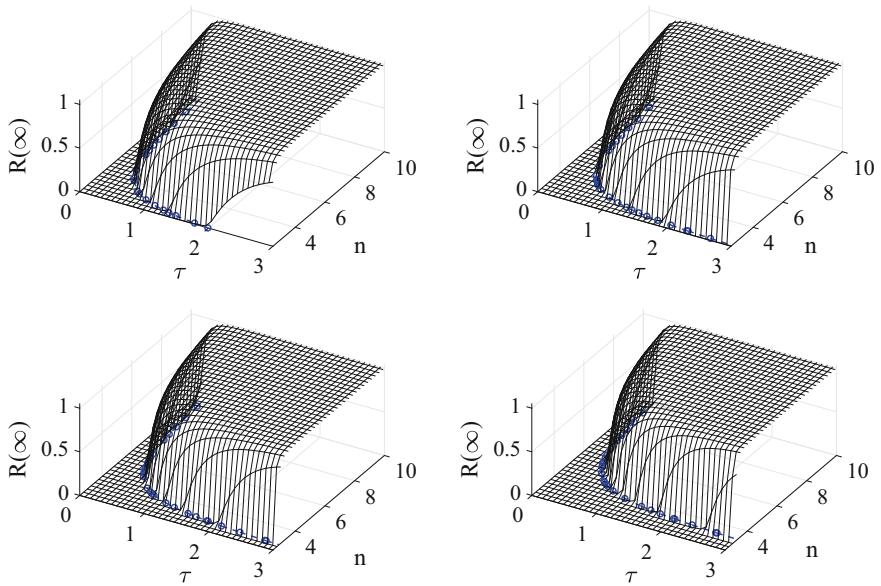


Fig. 2. Assessing the validity of the epidemic threshold based on the asymptotic expansion (29) (dashed line and markers - \circ) by comparing it to the threshold based on the numerical solution of closed pairwise system (9)–(13) (surface). Parameter values are $N = 10000$, $\gamma = 1$ and the clustering coefficients are top (from left to right) $\phi = 0, 0.2$, and bottom (from left to right) $\phi = 0.4, 0.6$.

4 Discussion

In this paper we set out to obtain an analytic epidemic threshold using the pairwise model with an improved closure to account for clustering. This problem has been solved previously in the unclustered case [8]. Here, we went one step further and showed that the quasi-equilibrium can be found as an asymptotic expansion in powers of the clustering coefficient. This paper builds on work in [1] and shows that exploiting the presence of fast variables and combining it with perturbation theory leads to a fruitful methodology which allowed us to compute the epidemic threshold analytically from pairwise models with three different closures. Strictly speaking there is no reason why this approach would not apply to other systems with properties similar to those found in the pairwise model. Reflecting on the results in [1] and in the present paper it is obvious that the epidemic threshold identified is model dependent. In simple terms this means that closing a pairwise model with different closures leads to different systems and hence different thresholds. This may under or overestimate the true threshold and care has to be taken if such models are used to capture real outbreaks, especially close to the threshold.

The ODE systems for the fast variables are worth investigating in more detail. We expect that these systems will exhibit a number of steady states. In fact preliminary numerical simulations suggest that the system corresponding to the fast variables (16)–(19) has at least one steady state which is close to the quasi-steady states shown in Fig. 1. Furthermore, it would be interesting to consider if the idea of fast variables extends to other mean-field models used in epidemiology. In particular it would be worthwhile to investigate if the correlation structure maps onto multi-variable models for heterogenous networks and if this consideration may lead to new insight from existing models. Equally, it remains a challenge to derive compact mean-field models for clustered networks. However, if such models will materialise we expect that our method may be a good candidate when it comes to the analysis of such models.

Finally, the natural next step would be to test our findings against explicit stochastic network simulations. This was beyond the scope of the present work, whose focus was on exploiting the presence of fast variables and the use of perturbation analysis to determine the epidemic threshold analytically.

Acknowledgments. István Z. Kiss acknowledges support from the Leverhulme Trust Research Project Grant (RPG-2017-370). Péter L. Simon acknowledges support from Hungarian Scientific Research Fund, OTKA, (grant no. 115926). Joel C. Miller acknowledges support from Global Good.

References

1. Barnard, R.C., Berthouze, L., Simon, P.L., Kiss, I.Z.: Epidemic threshold in pairwise models for clustered networks: closures and fast correlations. arXiv preprint [arXiv:1806.06135](https://arxiv.org/abs/1806.06135) (2018)
2. Eames, K.T.: Modelling disease spread through random and regular contacts in clustered populations. *Theor. Popul. Biol.* **73**(1), 104–111 (2008)

3. Eames, K.T., Keeling, M.J.: Modeling dynamic and network heterogeneities in the spread of sexually transmitted diseases. *Proc. Natl. Acad. Sci.* **99**(20), 13330–13335 (2002)
4. House, T., Davies, G., Danon, L., Keeling, M.J.: A motif-based approach to network epidemics. *Bull. Math. Biol.* **71**(7), 1693–1706 (2009)
5. House, T., Keeling, M.J.: The impact of contact tracing in clustered populations. *PLoS Comput. Biol.* **6**(3), e1000,721 (2010)
6. Karrer, B., Newman, M.E.: Message passing approach for general epidemic models. *Phys. Rev. E* **82**(1), 016,101 (2010)
7. Karrer, B., Newman, M.E.: Random graphs containing arbitrary distributions of subgraphs. *Phys. Rev. E* **82**(6), 066,118 (2010)
8. Keeling, M.J.: The effects of local spatial structure on epidemiological invasions. *Proc. R. Soc. Lond. B Biol. Sci.* **266**(1421), 859–867 (1999)
9. Kiss, I.Z., Miller, J.C., Simon, P.L.: Mathematics of Epidemics on Networks. Springer, Berlin (2017)
10. Li, J., Li, W., Jin, Z.: The epidemic model based on the approximation for third-order motifs on networks. *Math. Biosci.* **297**, 12–26 (2018)
11. Lindquist, J., Ma, J., Van den Driessche, P., Willeboordse, F.H.: Effective degree network disease models. *J. Math. Biol.* **62**(2), 143–164 (2011)
12. Miller, J.C.: Percolation and epidemics in random clustered networks. *Phys. Rev. E* **80**(2), 020,901 (2009)
13. Miller, J.C.: Spread of infectious disease through clustered populations. *J. R. Soc. Interface* **6**, rsif–2008 (2009)
14. Miller, J.C., Slim, A.C., Volz, E.M.: Edge-based compartmental modelling for infectious disease spread. *J. R. Soc. Interface* **9**(70), 890–906 (2012)
15. Miller, J.C., Volz, E.M.: Model hierarchies in edge-based compartmental modeling for infectious disease spread. *J. Math. Biol.* **67**(4), 869–899 (2013)
16. Newman, M.E.: Random graphs with clustering. *Phys. Rev. Lett.* **103**(5), 058,701 (2009)
17. Pastor-Satorras, R., Castellano, C., Van Mieghem, P., Vespignani, A.: Epidemic processes in complex networks. *Rev. Mod. Phys.* **87**(3), 925 (2015)
18. Pastor-Satorras, R., Vespignani, A.: Epidemic dynamics and endemic states in complex networks. *Phys. Rev. E* **63**(6), 066,117 (2001)
19. Rand, D.: Correlation equations and pair approximations for spatial ecologies. Advanced Ecological Theory: Principles and Applications, vol. 100. Blackwell Science, London (1999)
20. Rattana, P., Blyuss, K.B., Eames, K.T., Kiss, I.Z.: A class of pairwise models for epidemic dynamics on weighted networks. *Bull. Math. Biol.* **75**(3), 466–490 (2013)
21. Ritchie, M., Berthouze, L., Kiss, I.Z.: Beyond clustering: mean-field dynamics on networks with arbitrary subgraph composition. *J. Math. Biol.* **72**(1–2), 255–281 (2016)
22. Sharkey, K.J., et al.: Pair-level approximations to the spatio-temporal dynamics of epidemics on asymmetric contact networks. *J. Math. Biol.* **53**(1), 61–85 (2006)
23. Sherborne, N., Miller, J.C., Blyuss, K.B., Kiss, I.Z.: Mean-field models for non-markovian epidemics on networks. *J. Math. Biol.* **76**(3), 755–778 (2018)
24. Trapman, P.: On analytical approaches to epidemics on networks. *Theor. Popul. Biol.* **71**(2), 160–173 (2007)
25. Volz, E.M., Miller, J.C., Galvani, A., Meyers, L.A.: Effects of heterogeneous and clustered contact patterns on infectious disease dynamics. *PLoS Comput. Biol.* **7**(6), e1002,042 (2011)



Consistent Approximation of Epidemic Dynamics on Degree-Heterogeneous Clustered Networks

A. Bishop¹, I. Z. Kiss², and T. House^{3(✉)}

¹ Centre for Complexity Science, University of Warwick, Coventry, CV4 7AL, UK
A.Bishop@warwick.ac.uk

² Department of Mathematics, School of Mathematical and Physical Sciences,
University of Sussex, Brighton, BN1 9QH, UK
I.Z.Kiss@sussex.ac.uk

³ School of Mathematics, University of Manchester, Manchester, M13 9PL, UK
thomas.house@manchester.ac.uk

Abstract. Realistic human contact networks capable of spreading infectious disease, for example studied in social contact surveys, exhibit both significant degree heterogeneity and clustering, both of which greatly affect epidemic dynamics. To understand the joint effects of these two network properties on epidemic dynamics, the effective degree model of Lindquist et al. [28] is reformulated with a new moment closure to apply to highly clustered networks. A simulation study comparing alternative ODE models and stochastic simulations is performed for SIR (Susceptible–Infected–Removed) epidemic dynamics, including a test for the conjectured error behaviour in [40], providing evidence that this novel model can be a more accurate approximation to epidemic dynamics on complex networks than existing approaches.

Keywords: Networks · Epidemiology · Moment Closure · SIR
Clustering

1 Introduction

Networks offer an unprecedented opportunity to represent and model contacts between interacting units at all scales ranging from proteins and individuals to countries via air transportation. This additional degree of freedom has lead to extensive modelling and analysis in mathematical epidemiology and has allowed the development of a number of network-based models, which are either based or parametrised by real network data, if available, or by using synthetic or theoretical networks which reflect and reproduce some observed local or global properties of real-world networks [6, 22, 27, 39]. Idealised networks often offer a greater

Work supported by the Engineering and Physical Sciences Research Council Grant numbers EP/I01358X/1 and EP/N033701/1.

degree of analytical tractability, which in turn offers clearer insight into the impact of network properties on epidemic outbreak threshold, final epidemic size or prevalence and on effectiveness or choice of control measures.

The many degrees of freedom offered by networks come at the cost of computational and mathematical complexity. In order to handle this, and for a systematic investigation and understanding of network processes, it is desirable not to rely solely on stochastic simulations. As a result a number of differential equation-based models have been developed including pairwise [18, 21, 42, 48], PGF or edge-based compartmental models [30, 33] and effective degree models [13, 28], to name just a few. The goal of all these models is the same: to derive a set of low-dimensional system of ordinary differential equations (ODEs) where variables correspond to some average quantity from the stochastic process—e.g. expected prevalence—over time. All such mean-field models require choice of a ‘state space’. For example, pairwise models initially concentrate on the expected number of nodes in different states, while effective degree models concentrate on the expected number of star like structures, i.e. a central node with all its neighbours, and the possible state that these can be in. Once chosen, evolution equations for these variables are derived. This, often heuristic, step still involves a precise book-keeping which in general yields a dependency on higher order states or moments, e.g. for pairwise models the expected number of infected nodes depends on the expected number of edges where one node is infected and the other is susceptible. These newly introduced higher-order structures or moments require further equations and hence to curtail the dependency on higher-order moments and the fast growth in the number of equations ‘closures’ are needed. This amounts to approximating higher-order moments in terms of lower order ones. The performance of mean-field models is then often tested by direct comparison to results based on explicit stochastic network simulations. If the process is successful and the mean-field model works well, the analysis of the stochastic epidemic on networks is mapped into analysing a system of ODEs. This can be done using dynamical systems tools and such analysis often leads to analytical results which explicitly reveal the interaction between network and disease characteristics. More importantly, it shows how the fundamental properties of the network impact on growth rate, epidemic threshold, final epidemic size and so on.

Degree or contact heterogeneity and degree-based mixing is well accounted for in existing differential equation models, but epidemics on networks which are clustered (i.e. where two nodes with a common neighbour are highly likely to be neighbours of each other [51]) pose more of a challenge. A major factor of this difficulty is the non-unique way in which global or network-level clustering can be achieved, and the same level of clustering can be achieved while keeping the degree distribution the same but using different distributions or different sets of motifs [14, 26, 38, 43].

In general, clustered bond percolation-type [12, 20, 31, 37] or PGF-based models [18, 44, 49] consider specific forms of clustering (e.g. non-overlapping triangles) while pairwise models [15, 16, 18, 21, 23] usually work best in the case where

clustering is ‘randomly’ distributed, as is the case when using rewiring algorithms such as the big-V [3, 14, 18], which precludes certain kinds of interaction between clustering and degree heterogeneity. The question of how to approximate epidemic dynamics on a more general complex network remains open, and is the topic of this paper.

In general, for clustered networks it is difficult to derive accurate differential equation models for epidemics. In this paper, we present an approach based on generalisation of the effective degree model to clustered networks, and we show that this newly-derived model outperforms the state-of-the-art models and display excellent agreement with results based on stochastic simulations for a range of degree distributions and clustering values.

The paper is structured as follows: In Sect. 2 an overview of relevant complex network concepts and terminology are presented; Sect. 3 introduces methods of generating clustered networks with a specific degree distribution; in Sect. 4 existing ODE models are introduced and discussed; Sect. 5 presents the extension of ODE models to dynamics on clustered networks including our novel ODE model; Sect. 6 details the simulation study performed to compare between models and test the closure performance against conjectured errors.

2 Model Definitions

2.1 Networks

A *network* (or *graph*) is a pair $\mathcal{G} = (\mathcal{N}, \mathcal{L})$ where \mathcal{N} is a size- N set of *nodes* and $\mathcal{L} \subseteq \mathcal{N} \times \mathcal{N}$ is a set of *links*. The information about a network can be usefully encoded in terms of an *adjacency matrix*, which has elements $G_{ab} = 1$ if $(a, b) \in \mathcal{L}$, and zero otherwise.

We will consider simple undirected networks without self edges meaning that $G_{aa} = 0$ and $G_{ab} = G_{ba}, \forall a, b \in \mathcal{N}$. The *degree* of node a is the number of links that it participates in, i.e.

$$k_a := \sum_{b \in \mathcal{N}} G_{ab} . \quad (1)$$

We assume that all degrees are integers between 1 and the *maximum degree* M . We will use angled brackets to refer to mean values of functions of degree across the network, i.e. for an arbitrary function f ,

$$\langle f(k) \rangle := \frac{1}{N} \sum_{a \in \mathcal{N}} f(k_a) . \quad (2)$$

One particularly important such expectation is the *probability generating function* (PGF) which is $\psi(x) := \langle x^k \rangle$.

A final network property of interest in this work is the *clustering coefficient*,

$$\phi := \frac{\sum_{a,b,c \in \mathcal{N}} G_{ab} G_{bc} G_{ca}}{\sum_{a,b,c \neq a \in \mathcal{N}} G_{ab} G_{bc}} \in [0, 1] . \quad (3)$$

Our interest is in networks with large size ($N \gg 1$), degree distributions with a non-infinite variance ($\langle k \rangle^2 \leq \langle k^2 \rangle < \infty$), and clustering coefficients that can be non-zero but are not particularly large ($\phi \in [0, 0.3]$).

2.2 Epidemic Dynamics

We will consider SIR (Susceptible–Infective–Removed) dynamics. At the individual level, an individual a has a random state $X_a \in \{S, I, R\}$. Individuals in state I move to state R at rate γ , and individuals in state S move to state I at a rate τ multiplied by the number of I individuals they are linked to on the network.

At the population level, we will consider expected total node, pair and triple counts in given disease state configurations,

$$[A] = \mathbb{E} \sum_{a \in \mathcal{N}} \mathbf{1}_{\{X_a = A\}} , \quad (4)$$

$$[AB] = \mathbb{E} \sum_{a,b \in \mathcal{N}} \mathbf{1}_{\{X_a = A \& X_b = B\}} G_{ab} , \quad (5)$$

$$[ABC] = \mathbb{E} \sum_{a,b,c \neq a \in \mathcal{N}} \mathbf{1}_{\{X_a = A \& X_b = B \& X_c = C\}} G_{ab} G_{bc} , \quad (6)$$

where $A, B, C \in \{S, I, R\}$ and $\mathbf{1}$ is the indicator function. We will distinguish notationally between closed and open triples,

$$[ABC]_\Delta = \mathbb{E} \sum_{a,b,c \in \mathcal{N}} \mathbf{1}_{\{X_a = A \& X_b = B \& X_c = C\}} G_{ab} G_{bc} G_{ca} , \quad (7)$$

$$[ABC]_\wedge = [ABC] - [ABC]_\Delta , \quad (8)$$

which will be particularly important later. We will also consider more detailed states—in particular $[A_{s,i}]$ represents the expected number of nodes in state A with s susceptible and i infective contacts. Another important definition is the *correlation* between states

$$\mathcal{C}_{AB} = \frac{N}{\langle k \rangle} \frac{[AB]}{[A][B]} , \quad (9)$$

which expresses how much more likely an $[AB]$ edge is over the null model. The understanding of this is key to the role of networks in shaping epidemic dynamics [21].

Our aim in this paper is to find methods for approximation of the expected population-level behaviour of the epidemic dynamics that are, as much as possible, logically consistent, well motivated, and accurate.

3 Models of Network Generation

Typically, a full epidemic network is not directly available from data and so a standard method is to work with probabilistic models for the network that

respect certain observable summary statistics—in our case, the degree distribution and clustering coefficient. We will now present several such algorithms in outline form—more detail on these can be found in relevant papers and textbooks, e.g. Newman [36].

In n -regular networks, all nodes have the same degree i.e. $k_a = n, \forall a \in \mathcal{N}$. A typical such network can be generated by starting with a network that is atypical but assuredly n -regular, for example a one-dimensional $(n/2)$ -nearest neighbour. The network is then rewired by removing two edges (a, b) and (c, d) sampled (without replacement) uniformly at random from \mathcal{L} , and then adding new edges (a, c) and (b, d) (Fig. 1 ③ → ④). Performing a large number of such ‘edge swaps’ will result in a n -regular network that is representative of this class of graphs.

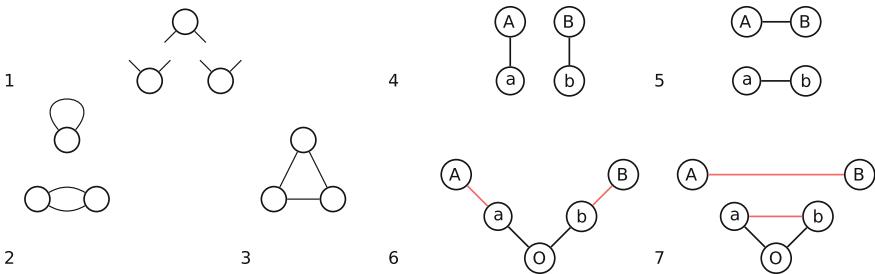


Fig. 1. The configuration model starts with a set of nodes with half-edges (①) and constructs a network by pairing these up to form a full-edge at random, resulting in configurations like ② & ③. ② shows an example where the configuration model can lead to self and duplicate edges, whereas ③ gives a valid configuration. ④ → ⑤ will rewire a network such that local structure is lost, for example performing many of these edge swaps yields a network with both negligible clustering and degree assortativity. In addition this rewiring introduces the small world property to a lattice. The clustering of a network may be increased by looking for V-shaped configurations (⑥) and performing the rewiring ⑥ → ⑦ if it increases the overall clustering coefficient of the network.

Erdős-Rényi networks [10], often referred to simply as ‘random graphs’ due to their importance, are defined such that each possible link is present with independent probability q . This leads to a binomial degree distribution although practically this will always be very well approximated by a Poisson distribution with mean $\langle k \rangle = (N - 1)q$.

In the configuration model (CM), nodes are given a number of ‘stubs’, which are then paired uniformly at random [35]. This construction is useful for the development of asymptotic results, but can cause problems for a given finite value of N due to the presence of ‘defects’—self-edges, multiple links between nodes, and stubs that are not eventually paired with others.

Networks with a given degree distribution can be generated following the procedure described in [9], which generates a sample of a given degree distribution that is graphical (meaning that a network with this degree distribution can

be constructed without defects such as self or multiple edges) and then directly samples a statistically independent network. This algorithm is better behaved than the Configuration Model as it always produces a simple graph without defects, backtracking or rejections.

Algorithms that generate clustered heterogeneous networks using one random process typically generate special network topologies. We therefore generate clustered networks via a two-step process in which we first generate a network of the required degree distribution, and then increase the clustering coefficient using a rewiring method known as the ‘big-V’ [3, 14, 18]. This involves finding a ‘V’ configuration in the network (Fig. 1 ⑥) and proposing a rewiring which produces a triangle and a separate edge (Fig. 1 ⑦).

This rewiring is then performed provided the clustering coefficient is increased. This method preserves the degree distribution whilst increasing the clustering coefficient (empirically) up to between $\phi = 0.3$ and $\phi = 0.4$ before the acceptance ratio of proposed moves critically slows down. The origin node, c , is selected with a weight proportional to $k_c(k_c - 1)$ so that the expected proportion of possible triangles present around each node does not depend on its degree [46].

4 Epidemic Dynamics on Locally Tree-like Networks

In the limit as $\phi \rightarrow 0$ while $N \rightarrow \infty$, it is possible to obtain results for the expected population-level epidemic dynamics that are known to be asymptotically exact.

4.1 Simple SIR Model

It is possible to show [47] that the following equations hold for an arbitrary network:

$$\frac{d}{dt}[S] = -\tau[SI] , \quad \frac{d}{dt}[I] = \tau[SI] - \gamma[I] . \quad (10)$$

There are two limits in which we can accurately approximate the pair variable $[SI]$ in terms of the node variables: (i) for an n -regular graph, as $n \rightarrow N - 1$; and (ii) for an ER graph with mean degree $\langle k \rangle$. In each case we take $[SI] \approx \langle k \rangle [S][I]/N$ with $\beta := \tau\langle k \rangle$ to end up with a special case of the classic model in mathematical epidemiology introduced almost a century ago by Kermack and McKendrick [24], often called the simple SIR model,

$$\frac{d}{dt}[S] = -\frac{\beta}{N}[S][I] , \quad \frac{d}{dt}[I] = \frac{\beta}{N}[S][I] - \gamma[I] . \quad (11)$$

Homogeneous mixing in a population is a poor assumption and more structural information is required to focus on the underlying network of contacts.

4.2 Pairwise Model

The pairwise model was one of the first steps toward this more realistic contact structure, and primarily concerns n -regular graphs. In this approach, we continue to write down equations in the form (10),

$$\frac{d}{dt}[SS] = -2\tau[ISS], \quad (12)$$

$$\frac{d}{dt}[SI] = \tau([ISS] - [ISI] - [IS]) - \gamma[IS], \quad (13)$$

$$\frac{d}{dt}[II] = 2\tau([ISI] + [IS]) - 2\gamma[II]. \quad (14)$$

The closed pairwise model [21, 41] is gained by taking the unclosed ODEs (10 and 12–14) and approximating the number of triples, $[ABC]$, in the system using a moment closure originally attributed to Kirkwood [25],

$$[ABC] \approx \frac{n-1}{n} \frac{[AB][BC]}{[B]}. \quad (15)$$

4.3 Effective Degree (ED) Model

Ball and Neal [2] introduced the notion of an effective degree. In the Ball and Neal construction, individuals (nodes) begin with a number of unpaired half-links/stubs—an *effective degree*—and a contact network is constructed along with SIR epidemic dynamics.

The dynamics occur as follows: stubs of infected nodes randomly connect with stubs of susceptible nodes at rate τ , reducing the effective degree of both nodes by one and infecting the susceptible node; infected nodes become recovered at rate γ , at which point they connect their remaining stubs uniformly at random to any remaining unpaired stubs in the network thus reducing their effective degree to zero. This process results in a configuration model network [35].

Following Ball and Neal [2], Lindquist et al. [29] developed an effective degree model categorising each node by its own disease state (S, I or R) along with the number of neighbours in each disease state resulting in the network being separated into classes representing the state of a node and its neighbours. For example, $S_{s,i}$ denotes the star motif corresponding to a susceptible node where s and i are the number of susceptible and infected neighbours respectively.

Analogously to the Pairwise model (see Sect. 4.2), ODEs describing the time evolution of network motifs may be written down; however for the Effective Degree model there are $M(M+3)$ equations representing the time evolution of each of the possible star motifs (where M is the maximum degree) compared to the equations of the pairwise model (12–14) which represent the time evolution of all node and all edge combinations, of which there are 5 ($[S]$, $[I]$, $[SS]$, $[SI]$, $[II]$).

The susceptible node at the centre of a $S_{s,i}$ star experiences a force of infection, τ , from each of its i infected neighbours with infection resulting in the transition from an $S_{s,i}$ star to an $I_{s,i}$ star. Each of the i infectious neighbours

recover at rate γ and thus transition from the $S_{s,i}$ class to the $S_{s,i-1}$ class at rate $\gamma i[S_{s,i}]$. By the same reasoning the rate at which transitions from $S_{s,i+1}$ to $S_{s,i}$ occur is $\gamma(i+1)[S_{s,i+1}]$. Infection of one the s susceptible neighbours results in a transition from $S_{s,i}$ to $S_{s-1,i+1}$ occurring at a rate of

$$\frac{\sum_{j,l} \tau jl[S_{j,l}]}{\sum_{j,l} j[S_{j,l}]} s[S_{s,i}], \quad (16)$$

where above and henceforth notation of the form $\sum_{j,l} = \sum_{k=1}^M \sum_{j+l=k}$ is adopted to aid brevity. The rate derives from the fact that new infections are generated at rate $\sum_{j,l} \tau l[S_{j,l}]$ which in turn causes the effective degree of the susceptible neighbours of the now infected $[S_{j,l}]$ to change at rate $\sum_{j,l} \tau jl[S_{j,l}]$. Put in the notation of Sect. 4.2 this can be expressed as the rate at which transitions $[ISS] \rightarrow [IIS]$ occur which is $\tau[ISS]$ where,

$$[ISS] = \sum_{j,l} [IS_{j,l} S] = \sum_{j,l} jl[S_{j,l}]. \quad (17)$$

As we wish to express the rate at which a susceptible neighbour of a $S_{s,i}$ star becomes infected, i.e. the rate of $[ISS_{s,i}] \rightarrow [IIS_{s-1,i+1}]$, we must account for the probability that the $S-S$ link in the $[ISS]$ triple will connect to a $S_{s,i}$ star which is given by the following identity,

$$[ISA_{s,i}] = [ISA] \frac{s[A_{s,i}]}{\sum_{j,l} j A_{j,l}}, \quad A \in \{S, I\}. \quad (18)$$

Putting this all together then yields the rate given in (16). Mutatis mutandis the rate of transition from $S_{s+1,i-1}$ to $S_{s,i}$ is obtained.

Using the above rates the set of ODEs describing the time evolution of star motifs with a central susceptible are obtained as (19) below. Following similar reasoning allows (20) below to be derived with the extra term $\gamma[I_{s,i}]$ corresponding to the recovery of the infectious individual at the centre of the star:

$$\frac{d[S_{s,i}]}{dt} = \gamma((i+1)[S_{s,i+1}] - i[S_{s,i}]) + \frac{\sum_{j,l} \tau jl[S_{j,l}]}{\sum_{j,l} j[S_{j,l}]} ((s+1)[S_{s+1,i-1}] - s[S_{s,i}]) - \tau i[S_{s,i}] \quad (19)$$

$$\frac{d[I_{s,i}]}{dt} = \gamma((i+1)[I_{s,i+1}] - i[I_{s,i}]) + \frac{\sum_{j,l} \tau l^2[S_{j,l}]}{\sum_{j,l} j[I_{j,l}]} ((s+1)[I_{s+1,i-1}] - s[I_{s,i}]) - \gamma[I_{s,i}] + \tau i[S_{s,i}] \quad (20)$$

The constraints upon s and i are given by $\{(s,i) : s \geq 0, i \geq 0, s+i \leq M\}$. Figure 2 of Lindquist et al. [28] summarises these two sets of equations governing the system in graphical form.

It is noteworthy to remark that the closure in this model is more implicit than that of the pairwise model, i.e. rather than approximating higher order states with lower order states one makes the assumption that the infectious pressure on the susceptible neighbours of the central node is equal to the population average.

4.4 Probability Generating Function (PGF) Methods

PGF models [34, 50] provide low-dimensional representations of epidemic dynamics on configuration models and have been proved to be asymptotically exact [4, 5, 7, 19].

In the simplest form of PGF model, given in [32], epidemic dynamics can be captured by one differential equation,

$$\frac{d\theta}{dt} = -\tau\theta + \gamma(1 - \theta) + \tau \frac{\psi'(\theta)}{\psi'(1)}. \quad (21)$$

The Volz variable θ represents the probability that a ‘test node’ with one link remains susceptible, and its use is responsible for the massive simplicity of this model.

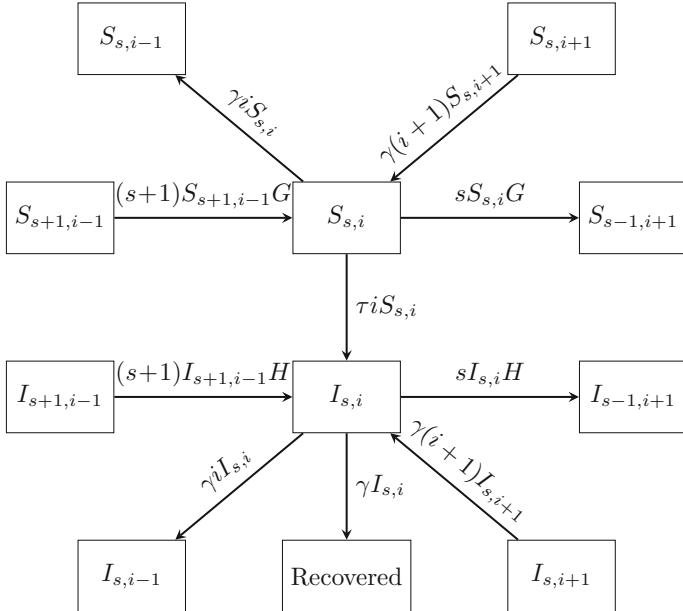


Fig. 2. Flow chart for the effective degree model illustrating the flow rates in to and out of motifs $S_{s,i}$ and $I_{s,i}$, Where $G = \frac{\sum_{j,l} \tau_j l [S_{j,l}]}{\sum_{j,l} j [S_{j,l}]}$, and $H = \frac{\sum_{j,l} \tau_j l^2 [S_{j,l}]}{\sum_{j,l} j [S_{j,l}]}$. Reproduced from [29]

5 Dynamics on Clustered Networks

5.1 Pairwise Model

The Kirkwood closure of Sect. 4.2 can be extended to clustered networks [21] by introducing a correlation term, $\mathcal{C}_{CA} = \frac{N}{\langle k \rangle} \frac{[CA]}{[A][C]}$, that accounts for the number of

transitive links between A and C by measuring the observed $[CA]$ compared to the number of A–C pairs one would expect given independence between C and A, $\frac{\langle k \rangle [A][C]}{N}$. If $\mathcal{C}_{CA} = 1$ then nodes of type C and nodes of type A are connected at random. Given a clustering coefficient, ϕ , the clustered Kirkwood closure is,

$$[ABC] \approx \frac{n-1}{n} \frac{[AB][BC]}{[B]} ((1-\phi) + \phi \mathcal{C}_{AC}) . \quad (22)$$

5.2 Clustered PGF

House and Keeling [18] introduced the clustered PGF model, which consists of the following differential equations with $[S] = N\psi(\theta)$ and $Y = \sum_k k[I_k]$:

$$\frac{d\theta}{dt} = -\tau \frac{[SI]}{N\psi'(\theta)} , \quad \frac{d[I]}{dt} = \tau[SI] - \gamma[I] , \quad \frac{dY}{dt} = \tau \left(\frac{\theta\psi''(\theta)}{\psi'(\theta)} + 1 \right) - \gamma Y , \quad (23)$$

together with (12), (13) and (14), and the closures

$$[ISS] \approx \frac{[SI][SS]\psi''(\theta)}{N\psi'(\theta)^2} \left((1-\phi) + \phi \langle k \rangle \frac{[SI]}{N^2\theta\psi'(\theta)^2} \right) . \quad (24)$$

$$[ISI] \approx \frac{[SI]^2\psi''(\theta)}{N\psi'(\theta)^2} \left((1-\phi) + \phi \langle k \rangle N \frac{[II]}{Y^2} \right) . \quad (25)$$

The clustered PGF is the main model that attempts to deal with epidemics on clustered heterogeneous networks and as such is the main direct comparator to our new approach.

5.3 A New Model

The ED degree model (Sect. 4.3) approximates the epidemic dynamics on a network of arbitrary degree distribution extremely well; however, in the large network limit the clustering coefficient of the configuration model network it explains tends to zero, which does not realistically explain real world contact networks. A new set of equations extending the effective degree model to clustered networks is now presented.

An exact version of the ED model may be written as follows,

$$\frac{d[S_{s,i}]}{dt} = -\tau i[S_{s,i}] + \gamma((i+1)[S_{s,i+1}] - i[S_{s,i}]) - \tau [ISS_{s,i}]_\Delta - \tau [ISS_{s,i}]_\wedge + \quad (26)$$

$$\tau [ISS_{s+1,i-1}]_\Delta + \tau [ISS_{s+1,i-1}]_\wedge ,$$

$$\begin{aligned} \frac{d[I_{s,i}]}{dt} = & \tau i[S_{s,i}] + \gamma((i+1)[I_{s,i+1}] - i[I_{s,i}]) - \tau [ISI_{s,i}]_\Delta - \tau [ISI_{s,i}]_\wedge + \tau [ISI_{s+1,i-1}]_\Delta \\ & + \tau [ISI_{s+1,i-1}]_\wedge - \gamma[I_{s,i}] + \tau((s+1)[I_{s+1,i-1}] - s[I_{s,i}]) , \end{aligned} \quad (27)$$

where we separate out transmission events happening within and outside of the neighbourhood of the star: $[ISS_{s,i}]_\Delta$ denotes a closed triple forming a triangle (e.g. edge ① Fig. 3) and $[ISS_{s,i}]_\wedge$ denotes an unclosed triple (e.g. edge ② Fig. 3).

Table 1. Counts of possible triangles around a node in state $A_{s,i}$.

Triangle states	$[SSA_{s,i}]_\Delta$	$[ISA_{s,i}]_\Delta$	$[IIA_{s,i}]_\Delta$
Combinations	$\frac{1}{2}s(s-1)$	is	$\frac{1}{2}i(i-1)$

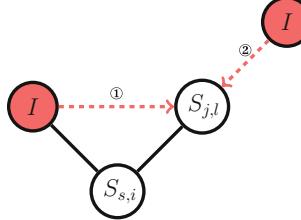


Fig. 3. Illustration of transmission events occurring within the neighbourhood of a central node ($S_{s,i}$) and outside of it. The edge labelled ① denotes a transmission event within the neighbourhood and the edge labelled ② denotes a transmission event outside of the neighbourhood of the central $S_{s,i}$.

In a network with a clustering coefficient of zero then $[ABC]_\Delta = 0$ and a closure on the $[ABC]_\wedge$ terms will yield the ED model [29]; however an effective closure for clustered networks must be made on both Δ and \wedge terms.

Given a clustering coefficient, ϕ , there are an expected $\phi k(k-1)/2$ triangles around a given degree- k node. If we decompose the effective degree (c.f. Sect. 4.3) of a node, k , into its susceptible and infected neighbours such that $s+i=k$ then Table 1 enumerates the expected number of triangles involving different states, given no correlations between states.

$[ISS_{s,i}]_\Delta = \phi is[S_{s,i}]$ as the clustering coefficient, ϕ , gives the expected ratio of edges connecting a node's neighbours together to the maximum possible number of such edges (Table 1). Correlations between the states cannot be ignored, therefore the correlation between nodes of state A and B, $C_{AB} = \frac{N}{\langle k \rangle} \frac{[AB]}{[A][B]}$, is introduced which is used to account for how many $[AB]$ pairs there are compared to how many one would expect from random mixing, $\langle k \rangle [A] \frac{[B]}{N}$. This yields the closure Eq. (31), and mutatis mutandis (33) is obtained.

The original ED closure for the \wedge terms must now be modified to account for the infection events that happen within the Δ closure. The new closure is achieved by taking the original expressions and, for $A \in \{S, I\}$, using the identity $[ISA]_\Delta + [ISA]_\wedge = [ISA]$ along with the original and the Δ closure. For example,

$$[ISS] = \sum_{j,l} [IS_{j,l}S] = \sum_{j,l} jl[S_{j,l}], \quad (28)$$

$$[ISS]_\Delta = \sum_{j,l} [IS_{j,l}S]_\Delta = \sum_{j,l} \phi C_{SI} jl[S_{j,l}], \quad (29)$$

$$\implies [ISS]_\wedge = \sum_{j,l} [IS_{j,l}S]_\wedge = \sum_{j,l} jl(1 - \phi C_{SI})[S_{j,l}]. \quad (30)$$

Using identity (18), one gains the final clustered effective degree closures (31–34) which when substituted into (26–27) yield the clustered ED model.

$$[ISS_{s,i}]_\Delta = \phi \mathcal{C}_{\mathcal{S}\mathcal{I}} si[S_{s,i}] , \quad (31)$$

$$[ISS_{s,i}]_\wedge = \frac{\sum_{j,l} jl(1 - \phi \mathcal{C}_{\mathcal{S}\mathcal{I}})[S_{j,l}]}{\sum_{j,l} j[S_{j,l}]} s[S_{s,i}] , \quad (32)$$

$$[ISI_{s,i}]_\Delta = \phi \mathcal{C}_{\mathcal{I}\mathcal{I}} si[I_{s,i}] , \quad (33)$$

$$[ISI_{s,i}]_\wedge = \frac{\sum_{j,l} l(l-1)(1 - \phi \mathcal{C}_{\mathcal{I}\mathcal{I}})[S_{j,l}]}{\sum_{j,l} j[I_{j,l}]} s[I_{s,i}] . \quad (34)$$

6 Simulation Study

To test the accuracy of the clustered ED model we perform a simulation study.

6.1 Methodology

First, unclustered networks are generated for three different degree distributions according to the methods presented in Sect. 3. The big-V algorithm (Fig. 1 (7)) was then applied to each unclustered network to generate a series of networks with approximate clustering coefficients $\phi \in \{0.05, 0.10, 0.15, 0.20, 0.25, 0.30\}$.

Numerical simulation was performed using an individual-based analogue of Gillespie's algorithm [11], which generates a statistically correct trajectory with regard to the master equation of the underlying stochastic process. Simulations were performed for 20 uniquely generated networks of 10^5 nodes with the dynamics being simulated on each network 5 times, for a total of 100 epidemics per network type. To account for the large stochastic variability at the beginning of an epidemic, we shifted the time-origins of each of the 100 epidemics to coincide at the point where 500 individuals are infected before averaging the dynamics, which we then compare to each differential equation model.

We choose to generate and simulate networks of mean degree four as a rigorous model comparator—closures perform better as $\langle k \rangle$ increases as the behaviour tends towards the simple SIR model since these networks are closer to complete graphs.

6.2 Results

The results in Figs. 4, 5 and 6 show that our novel clustered ED model generally outperforms the clustered PGF approach in terms of errors in expected prevalence, with exceptions to this only occurring after the epidemic peak.

Work by Pellis et al. [40] argued (based on a rigorous analysis of finite-size networks) that we should expect moment closure to be exact when (i) triangles are not overlapping and (ii) recovery happens after a constant time (or not at all as when $\gamma = 0$). When the clustering coefficient is small, the proportion of overlapping triangles will be $O(\phi^2)$, and so for small values of γ we expect

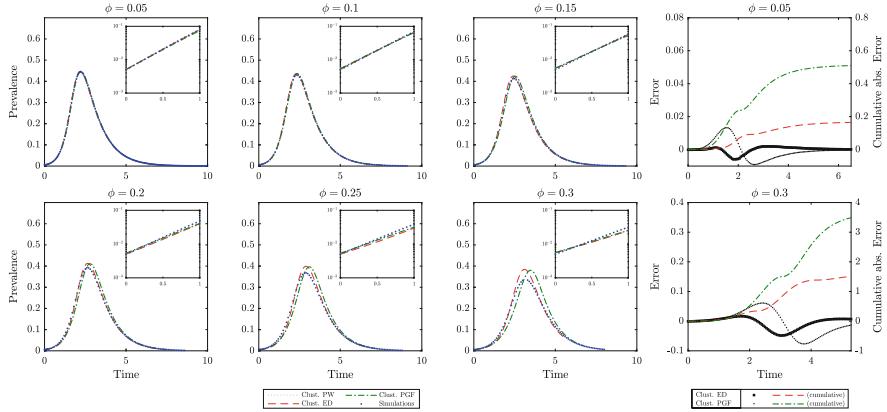


Fig. 4. Simulated epidemic dynamics for 4-regular networks with $\gamma = 1, \tau = 2, N = 10^5$.

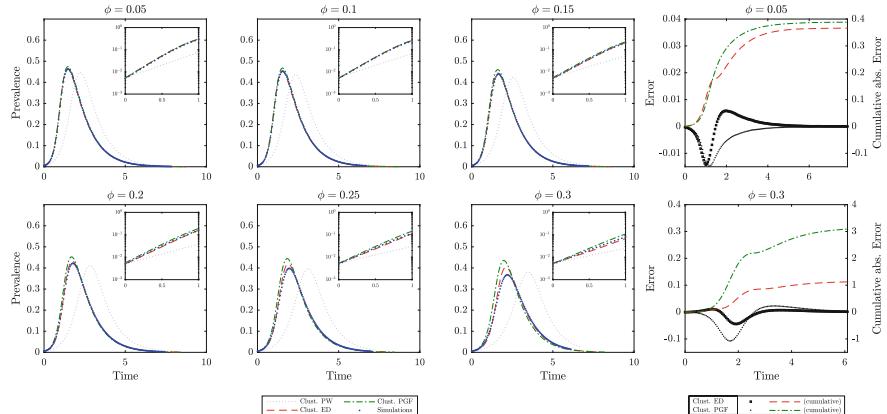


Fig. 5. Simulated epidemic dynamics for Erdős-Rényi networks with $\langle k \rangle = 4, \gamma = 1, \tau = 2, N = 10^5$.

errors in the prediction of prevalence of infection to be $O(\phi^2)$, while for larger values we expect them to be $O(\phi)$. We found that in general, obtaining accurate assessments of absolute error in predictions of prevalence was numerically challenging, and it is likely that deeper theoretical understanding of the source of errors would be required to perform a definitive computational analysis of this question. Nevertheless, we were able to obtain the results shown in Fig. 7, which demonstrate that as expected errors lie between the $O(\phi)$ and $O(\phi^2)$ lines.

7 Discussion

In this paper, we have shown how to combine effective degree approaches to epidemics on heterogeneous networks with moment closure approaches to clustering. Numerical results suggest that the errors introduced in so doing may be

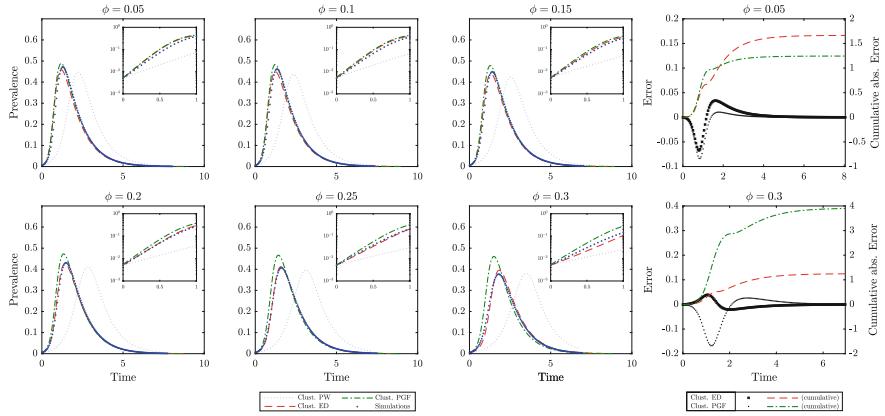


Fig. 6. Simulated epidemic dynamics for Negative binomial networks with $\langle k \rangle = 4, r = 5, \gamma = 1, \tau = 2, N = 10^5$. We use the formulation where we are counting k successes given r failures.

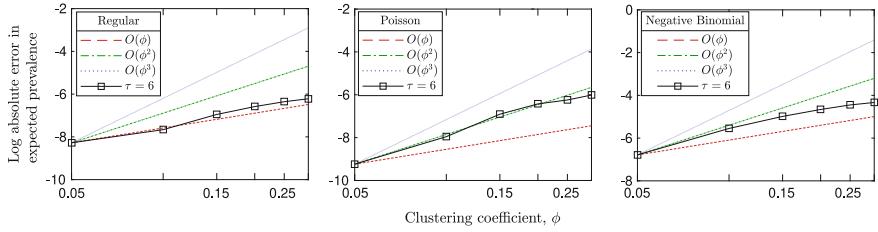


Fig. 7. Error between the clustered ED model and expected prevalence of the stochastic simulations detailed in Sect. 6.1 is plotted against errors of order ϕ , ϕ^2 , and ϕ^3 plotted in red, green, and blue respectively.

better than $O(\phi)$ but worse than $O(\phi^2)$, meaning that the potential for improvements remains. In particular, the closure due to Kirkwood [25] could be replaced by the improved closure in [17] or the maximum entropy method [45]. In particular, in the large ϕ limit we expect graphs to be dominated by cliques [8], whose differential equation limit is described by equations such as those written in [1].

References

1. Ball, F.: Stochastic and deterministic models for SIS epidemics among a population partitioned into households. *Math. Biosci.* **156**(1), 41–67 (1999)
2. Ball, F., Neal, P.: Network epidemic models with two levels of mixing. *Math. Biosci.* **212**(1), 69–87 (2008)
3. Bansal, S., Khandelwal, S., Meyers, L.A.: Exploring biological network structure with clustered random networks. *BMC Bioinform.* **10**(1), 405 (2009)
4. Barbour, A., Reinert, G.: Approximating the epidemic curve. *Electron. J. Probab.* **18**(54), 1–30 (2013)

5. Bohman, T., Picollelli, M.: SIR epidemics on random graphs with a fixed degree sequence. *Random Struct. Algorithms* **41**(2), 179–214 (2012)
6. Danon, L., et al.: Networks and the epidemiology of infectious disease. *Interdiscip. Perspect. Infect. Dis.* **2011** (2011)
7. Decreusefond, L., Dhersin, J.S., Moyal, P., Tran, V.C.: Large graph limit for an SIR process in random network with heterogeneous connectivity. *Ann. Appl. Probab.* **22**(2), 541–575 (2012)
8. Del Genio, C.I., House, T.: Endemic infections are always possible on regular networks. *Phys. Rev. E* **88**, 040,801 (2013)
9. Del Genio, C.I., Kim, H., Toroczkai, Z., Bassler, K.E.: Efficient and exact sampling of simple graphs with given arbitrary degree sequence. *PLoS one* **5**(4), e10,012 (2010)
10. Gilbert, E.N.: Random graphs. *Ann. Math. Stat.* **30**(4), 1141–1144 (1959)
11. Gillespie, D.T.: A general method for numerically simulating the stochastic time evolution of coupled chemical reactions. *J. Comput. Phys.* **22**(4), 403–434 (1976)
12. Gleeson, J.P.: Bond percolation on a class of clustered random networks. *Phys. Rev. E* **80**(3), 036107 (2009)
13. Gleeson, J.P.: Binary-state dynamics on complex networks: pair approximation and beyond. *Phys. Rev. X* **3**(2), 021004 (2013)
14. Green, D., Kiss, I.: Large-scale properties of clustered networks: Implications for disease dynamics. *J. Biol. Dyn.* **4**(5), 431–445 (2010)
15. House, T.: Generalised network clustering and its dynamical implications. *Adv. Complex Syst.* **13**(3), 281–291 (2010)
16. House, T., Davies, G., Danon, L., Keeling, M.J.: A motif-based approach to network epidemics. *Bull. Math. Biol.* **71**(7), 1693–1706 (2009)
17. House, T., Keeling, M.J.: The impact of contact tracing in clustered populations. *PLoS Comput. Biol.* **6**(3), e1000721 (2010)
18. House, T., Keeling, M.J.: Insights from unifying modern approximations to infections on networks. *J. R. Soc. Interface* **8**(54), 67–73 (2011)
19. Janson, S., Luczak, M., Windridge, P.: Law of large numbers for the SIR epidemic on a random graph with given degrees. *Random Struct. Algorithms* **45**(4), 726–763 (2014)
20. Karrer, B., Newman, M.: Random graphs containing arbitrary distributions of subgraphs. *Phys. Rev. E* **82**, 066,118 (2010)
21. Keeling, M.J.: The effects of local spatial structure on epidemiological invasions. *Proc. R. Soc. London. Ser. B: Biol. Sci.* **266**(1421), 859–867 (1999)
22. Keeling, M.J., Eames, K.T.: Networks and epidemic models. *J. R. Soc. Interface* **2**(4), 295–307 (2005)
23. Keeling, M.J., House, T., Cooper, A.J., Pellis, L.: Systematic approximations to susceptible-infectious-susceptible dynamics on networks. *PLOS Comput. Biol.* **12**(12), e1005,296 (2016)
24. Kermack, W., McKendrick, A.: Wo kermack and ag mckendrick, proc. r. soc. london, ser. a **115**, 700 (1927). *Proc. R. Soc. London, Ser. A* **115**, 700 (1927)
25. Kirkwood, J.G., Boggs, E.M.: The radial distribution function in liquids. *J. Chem. Phys.* **10**(6), 394–402 (1942)
26. Kiss, I.Z., Green, D.M.: Comment on ‘properties of highly clustered networks’. *Phys. Rev. E* **78**(4), 048101 (2008)
27. Kiss, I.Z., Miller, J.C., Simon, P.L.: Mathematics of Epidemics on Networks. Springer, Berlin (2017)
28. Lindquist, J., Ma, J., van den Driessche, P., Willeboordse, F.: Effective degree network models. *J. Math. Biol.* **62**, 143 (2010)

29. Lindquist, J., Ma, J., van den Driessche, P., Willeboordse, F.H.: Effective degree network disease models. *J. Math. Biol.* **62**(2), 143–164 (2011)
30. Miller, J., Slim, A., Volz, E.: Edge-based compartmental modelling for infectious disease spread. *J. R. Soc. Interface* **9**(70), 890–906 (2012)
31. Miller, J.C.: Percolation and epidemics in random clustered networks. *Phys. Rev. E* **80**(2), 020,901 (2009)
32. Miller, J.C.: A note on a paper by Erik Volz: SIR dynamics in random networks. *J. Math. Biol.* **62**(3), 349–358 (2011)
33. Miller, J.C., Slim, A.C., Volz, E.M.: Edge-based compartmental modelling for infectious disease spread. *J. R. Soc. Interface*, rsif20110403 (2011)
34. Miller, J.C., Slim, A.C., Volz, E.M.: Edge-based compartmental modelling for infectious disease spread. *J. R. Soc. Interface* **9**(70), 890–906 (2012)
35. Molloy, M., Reed, B.: A critical point for random graphs with a given degree sequence. *Random Struct. Algorithms* **6**(2–3), 161–180 (1995)
36. Newman, M.: Networks : An Introduction. Oxford University Press, Oxford (2009)
37. Newman, M.: Random graphs with clustering. *Phys. Rev. Lett.* **103**(5), 058701 (2009)
38. Newman, M.E.: Properties of highly clustered networks. *Phys. Rev. E* **68**(2), 026121 (2003)
39. Pastor-Satorras, R., Castellano, C., Van Mieghem, P., Vespignani, A.: Epidemic processes in complex networks (2014). arXiv preprint [arXiv:1408.2701](https://arxiv.org/abs/1408.2701)
40. Pellis, L., House, T., Keeling, M.J.: Exact and approximate moment closures for non-Markovian network epidemics. *J. Theor. Biol.* **382**, 160–177 (2015)
41. Rand, D.: Correlation equations and pair approximations for spatial ecologies. *Adv. Ecol. Theory: Princ. Appl.* **100** (1999)
42. Rand, D.: Advanced ecological theory: principles and applications, chap. Correlation equations and pair approximations for spatial ecologies, pp. 100–142. Wiley, New York (2009)
43. Ritchie, M., Berthouze, L., House, T., Kiss, I.Z.: Higher-order structure and epidemic dynamics in clustered networks. *J. Theor. Biol.* **348**, 21–32 (2014)
44. Ritchie, M., Berthouze, L., Kiss, I.Z.: Beyond clustering: Mean-field dynamics on networks with arbitrary subgraph composition (2014). arXiv preprint [arXiv:1405.6234](https://arxiv.org/abs/1405.6234)
45. Rogers, T.: Maximum-entropy moment-closure for stochastic systems on networks. *J. Stat. Mech.: Theory Exp.* **2011**(05), P05,007 (2011)
46. Serrano, M.A., Boguñá, M.: Percolation and epidemic thresholds in clustered networks. *Phys. Rev. Lett.* **97**, 088,701 (2006)
47. Simon, P., Taylor, M., Kiss, I.: Exact epidemic models on graphs using graph automorphism driven lumping. *J. Math. Biol.* **62**, 479–508 (2010)
48. Taylor, M., Simon, P.L., Green, D.M., House, T., Kiss, I.Z.: From markovian to pairwise epidemic models and the performance of moment closure approximations. *J. Math. Biol.* **64**(6), 1021–1042 (2012)
49. Volz, E., Miller, J., Galvani, A., Ancel-Meyers, L.: Effects of heterogeneous and clustered contact patterns on infectious disease dynamics. *PLoS Comput. Biol.* **7**(6), e1002042 (2011)
50. Volz, E.M.: SIR dynamics in random networks with heterogeneous connectivity. *J. Math. Biol.* **56**(3), 293–310 (2008)
51. Watts, D., Strogatz, S.: Collective dynamics of ‘small-world’ networks. *Nature* **393**(6684), 440–442 (1998)



DiffuGreedy: An Influence Maximization Algorithm Based on Diffusion Cascades

George Panagopoulos^{1(✉)}, Fragkiskos D. Malliaros², and Michalis Vazirgiannis^{1,3}

¹ Ecole Polytechnique, Palaiseau, France

{george.panagopoulos,mvazirg}@polytechnique.edu

² CentraleSupélec and Inria Saclay, Gif-sur-Yvette, France

fragkiskos.malliaros@centralesupelec.fr

³ Athens University of Economics and Business, Athens, Greece

Abstract. Finding a set of nodes that maximizes the spread in a network, known as the influence maximization problem, has been addressed from multiple angles throughout the literature. Traditional solutions focus on the algorithmic aspect of the problem and are based solely on static networks. However, with the emergence of several complementary data, such as the network’s temporal changes and the diffusion cascades taking place over it, novel methods have been proposed with promising results. Here, we introduce a simple yet effective algorithm that combines the algorithmic methodology with the diffusion cascades. We compare it with four different prevalent influence maximization approaches, on a large scale Chinese microblogging dataset. More specifically, for comparison, we employ methods that derive the seed set using the static network, the temporal network, the diffusion cascades, and their combination. A set of diffusion cascades from the latter part of the dataset is set aside for evaluation. Our method outperforms the rest in both quality of the seed set and computational efficiency.

Keywords: Influence maximization · Information spreading
Large-scale network analysis

1 Introduction

Albeit the massive amount of work over this problem over the past 15 years, influence maximization (IM) remains the holy grail of social network analysis. The problem, at its core, is to find a set of nodes that would infect the largest possible part of the network, if a spreading process started from them. It is proven to be NP-hard, but a greedy algorithm [11] can get at least as close as $(1-1/e)$ to the optimum, under the two most prevalent spreading models, the independent cascade (IC) and the linear threshold (LT). The edge weights represent the probability of influence in IC and the amount of influence in LT. Typically, IM algorithms work with uniform or degree-based edge weights. This

renders their spreading estimation wildly divergent from the actual spreading [8], due to the complexity and diversity that governs spreading processes [25]. To this end, some methods focus on learning the transmission probabilities between nodes using real diffusion cascades [22]. This research branch is divided into models that use the cascades to define an inferred network [18] or to adjust the edge weights of an existing underlying network [7]. An example of the former is to infer a transmission probability between two news blogs based on how often and fast one copies the other, while an example of the latter is defining the strength of a follow relationship in twitter based on how many times one node retweeted the other. IM algorithms can then run on such inferred or weighted [8] networks to derive the seed set. Although these methods tend to approximate real spreading better, they suffer from issues of scalability and overfitting.

In this work, we propose DIFFUGREEDY, an algorithm based on SIMUGREEDY [11] that utilizes the real diffusion cascades instead of simulations over the network. Particularly, it follows the same hill climbing manner to construct the seed set iteratively, but the computation of the marginal gain is based on a candidate seed's most suitable diffusion cascade. This function is submodular, which allows us to retain the theoretical guarantees. To showcase the effectiveness of our method, we employ the temporal Sina Weibo follower network (1.7 m nodes, 400 m edges), that spans 32 days and is accompanied with the retweet cascades of that time span. We keep the diffusion cascades and the network of the first 25 days as the train set and the last week as the test set. We use four different IM approaches in the train set for comparison with DIFFUGREEDY. The first is ranking the nodes by k-core decomposition on the follower network, which has indicated strong correlations with influence [15]. Secondly, we employ IMM [23], one of the fastest IM algorithms, to extract a seed set efficiently from the follower network as is formed at the end of the training set. The third approach utilizes NETRATE [18] to infer a network from the diffusion cascades in the train set, and applies PMIA [3] to perform IM on it. In the final method, we use the diffusion cascades to weigh the follow edges proportionally to the nodes' activity and apply SIMPATH [9] on the resulting network.

It should be noted that a comparison between all these approaches has not been attempted in the literature before, to the best of our knowledge. This can be due to the lack of a common realistic evaluation methodology. Methods that work on static networks are evaluated based on their computational time and the estimated influence spread. On the other hand, diffusion learning methods are evaluated based on the behavior of the chosen seed set in unseen cascades. We deem the latter more realistic, and we choose to validate our seed set using the number of distinct nodes influenced by it in the test set [5]. The results indicate that Diffusion Greedy and its CELF counterpart clearly outperform the rest in terms of the seed set's influence spread in the test set. In addition, the CELF approach is an order of magnitude faster to the second fastest method. Finally, we notice that the methods based on k-core and NETRATE perform adequately, unlike the IMM. The paper is organized as follows. Section 2 is a brief literature review on influence maximization. Section 3 delineates some state

of the art influence maximization methods we employed for our comparative analysis. Sect. 4 describes the new algorithm we propose. Section 5 presents the dataset and the results of our experiments, along with insights and justifications. The paper concludes in Sect. 6 with a contribution synopsis and suggestions for future work.

2 Related Work

The basis of most IM algorithms is SIMUGREEDY [11]. Starting with an empty seed set, the algorithm adds in the set the node that provides the best marginal gain i.e. the increase of the set’s influence spread, in each iteration. Due to the monotonicity and submodularity of the influence spread function under the two diffusion models, the algorithm is guaranteed to reach a near optimal solution. The most time-consuming part of SIMUGREEDY is the influence spread estimation, which has proven to be P-hard [3]. Since the diffusion models are stochastic, all possible paths of influence need to be taken into account proportionally to their probability, which is not feasible, thus Monte Carlo simulations are employed. Most attempts to improve the algorithm focus on that part. CELF [13] capitalizes on the submodularity of a node’s marginal gain, which can only diminish as the seed set grows. This means that if a candidate seed’s marginal gain is higher than what the rest candidates had in the previous iteration, it is higher at the current iteration as well, hence removing the need to recompute the marginal gain of all candidates. PMIA [3] is a heuristic approach developed for the IC model, and it is based on the maximum influence in and out arborescence (MIIA and MIOA) of every node. This is the union of all maximum influence paths that end up or start from that node. The key part of the algorithm is that paths with probability under a certain threshold are removed, assuming that influence is mostly local. SIMPATH [9] is also based on the path-pruning idea, but for the LT, computing the influence spread of a node by summing the probabilities of paths that start from it. Although providing a substantial speedup, those heuristic methods do not retain theoretical guarantees, in contrast to sketch-based algorithms. The idea of sketch-based approaches is to create several instances of the network that represent varying outcomes of the edge probabilities beforehand and use them to estimate influence spread of a seed set. SKIM is an example of sketch-based IM [4], which alleviates the need for Monte Carlo simulations, achieving extreme acceleration with $(1 - \frac{1}{e} - \epsilon)$ approximate guarantees, where ϵ is a trade-off between accuracy and efficiency. An alternative and faster sketch based methodology with the same theoretical guarantees is based on Reverse Reachable (RR) sets [24]. An RR set of a node consists of other nodes that can influence it. After generating a sufficient number of RR sets for random nodes, the optimum seed set can be derived by selecting the nodes that cover their majority. The intuition is that the frequency of a node’s appearance in the RR sets is analogous to its influence.

While the previous algorithms work with a static network, there’s a growing literature dedicated to IM based on diffusion cascades. A diffusion cascade is a

series of events that takes place over the network and indicates how information spreads in it, e.g. a tweet and its list of retweets. The first model that utilized real diffusion cascades for IM, attempted to learn the transmission probabilities between nodes in IC [22]. The model uses survival analysis to express the probability of a node getting influenced in the course of a cascade, and it is solved using an EM algorithm. The same group proposed learning an extension of IC in continuous time (CTIC) [21]. CTIC extends traditional IC, by defining the diffusion probability between two nodes analogous to the number of times one node was influenced by the other, as well as the time it took for the latter to get influenced by the former. The intuition behind this is that the longer it takes for a node to copy an action the less likely it is to copy it. NETRATE [18] is a seminal algorithm that steps on both aforementioned works. It is based on similar modeling as [22], but it learns the transmission delays between nodes. NETRATE can be used to infer how the nodes of the cascades are connected when the underlying network is not available. Subsequently, time-constrained IM can run on that inferred network, to give an estimate of the most influential users [5, 19], solely based on the cascades. A similar line of work, but with a different purpose, is diffusion cascade learning [1]. These machine learning models use cascades to predict whether a node will get infected or not [17], or the size of the cascade [14] when a cascade has already started. In the intersection of the two aforementioned approaches, lie methods that use both, the follower network and the diffusion cascades. A characteristic example is the credit distribution model [8]. Whenever a node u copies a node v in the diffusion cascades, credits are given to v and to the nodes that v copied. The influence spread of a seed set is the total influence credit of its seeds and is estimated efficiently, by alternating between credit estimations from the action logs and CELF. A simpler approach is to weigh the edges of the graph analogously to the activity of the nodes, e.g. how many times v has copied u in the cascades [7]. Subsequently, an IM algorithm can run on this weighted network.

All aforementioned approaches, though differing methodologically, address the same problem. However, their evaluation methods are quite deviant. IM algorithms on static networks are evaluated based on their estimated spreading and time efficiency. These methods totally overlook the real spreading dynamics of the network, as they focus on the problem from a more algorithmic than data-driven perspective. In some cases, epidemic simulations like SIR and SIS are utilized to give an estimate of a seed set's spreading in the network [15]. However, these models suffer from oversimplifying assumptions [16] and overlook several important characteristics of real diffusion cascades [6]. Moreover, their spreading estimate has proved inaccurate compared to actual diffusions that take place over the network [20]. Hence, although epidemic models might be a valid choice in the absence of diffusion data, in our case, we can form a more realistic ground truth based on the diffusions. Even in this case, however, evaluation is not straightforward. An erroneous example is representing the spread of a seed set in the test set by the sum of the average size of each seed's test cascades [26]. This is inherently problematic because large cascades from individual seeds do not

guarantee a large combined spread. A similar fault occurs when evaluating a seed set based on each individual seed’s follows, mentions, retweets, and tweets [10]. Instead, our evaluation tactic is based on the number of distinct nodes influenced in the test set, by the seed set [5]. Although not devoid of assumptions, it is the closest and most objective measure of a seed set’s influence over a network at a given time span.

3 Influence Maximization Analysis

In this section, we describe the analytical framework we followed to apply different approaches of IM on the same dataset, which is comprised of a temporal network and diffusion cascades. As mentioned above, we split the dataset in the train and test set. The train set corresponds to the diffusion cascades and the follow relationships that took place during the first 25 days, as well as the initial follower graph which is formed before the first day of crawling. The test set consists of the last week’s diffusion cascades. In the train set, we utilize four different IM techniques to derive seed sets for comparison with the seed set derived by the proposed DIFFUGREEDY (described in Sect. 4). A general overview of the methodology can be seen in Fig. 1. Below we analyze each technique and how we applied it.

3.1 Ranking by K-Core Decomposition

K-core decomposition has proven a strong reliable predictor of influence in previous studies [12, 15]. The K-core of a network is the maximal subgraph such that each vertex has at least K degree. As a proxy for IM, we can rank the nodes based on the maximum K-core they belong to, i.e. their coreness, and take the top as a seed set.

3.2 Influence Maximization via Martingales

As a representative to classic IM approaches, we use IMM, an algorithm based on the aforementioned RR sets, to derive the seed set from the follower network. This is a network of more than 95 million edges, so efficiency is of utmost importance. The most important advantage of IMM is that, in contrast with the rest of RR-based algorithms, it derives RR sets that depend on one another. As a result, the number of RR sets is diminished dramatically. In addition, it achieves a theoretical guarantee of $((1 - 1/e)/(1 + \epsilon)^2)$ using martingale analysis. The network is weighted using weighted cascade [11] and the parameter ϵ , which governs the trade-off between speed and accuracy, is set to 0.1.

3.3 PMIA on the NETRATE Network

To perform IM based exclusively on the diffusion cascades, we employ NETRATE [18] to infer the transmission rates between users in the train set. This can,

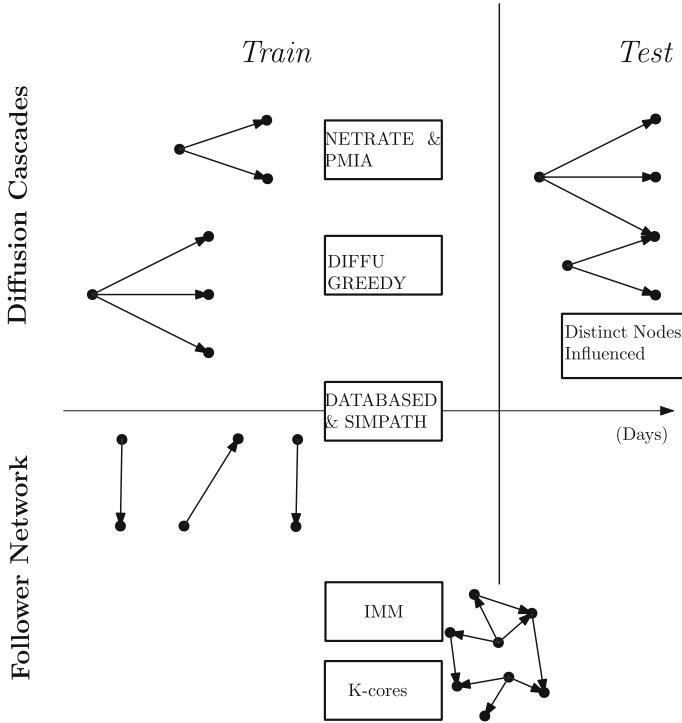


Fig. 1. An overview of the models we employed and which data they utilize. The upper quadrants correspond to diffusion cascades and the lower to the follow relationships established in time. The left quadrants belong to the train set and the right to the test set. Each network represents a different type of data. The static follower network is formed at the last day of the training set, so it is under the vertical axis. Each rectangle represents a different method, and its position indicates which type of data it is based on.

in turn, define a new network with edge weights proportional to the inferred rates. The size of the inferred network needs to be very limited to satisfy the computational demands of NETRATE. Hence we follow the literature [26] and filter the cascades to keep only the most important nodes. In our experiments, degree proved to produce the most effective diffusion network. We filter all the cascades to remove nodes, either starting or participating in the cascade, that do not belong to the top 3000 nodes. Having computed the network and its transmission rates, we tried to use InfluMax, which is the archetype algorithm for continuous-time IM. However, NETRATE's inferred transmission rates that exceeded 10^{-6} were a mere 50 out of the 37937 inferred edges. Instead, we used PMIA [3] on the diffusion network with weights defined by weighted cascade and pruning parameter $\theta = 1/320$. Although inferior to InfluMax, it has served as a method of comparison [19] and provides a more than descent approximation to SIMUGREEDY. Moreover, PMIA is based on IC, which is closer to NETRATE's

continuous-time IC then LT. Finally, since the diffusion network is small the computational requirements were minuscule.

3.4 SIMPATH on the DATABASED Weighted Network

To combine the information of the temporal network with the diffusion cascades, we employed an edge weighting technique [7], which belongs to the DATABASED approaches [8]. We assume that a node u copies a node v , whenever u appears after v in a diffusion cascade and the time that u started following v is before the cascade's initiation. The edge weight is defined as:

$$E_{v,u} = \frac{A_{v2u}}{A_v} \times e^{-\frac{\bar{D}t_{v,u}}{\delta}}, \quad (1)$$

where A_{v2u} is the number of times u copied v , A_v is the total number of tweets and retweets of v , and $\bar{D}t_{v,u}$ is the average time that takes for u to copy v .

The first term captures the relationship's strength while the second is analogous to its speed and depicts the exponential decay of influence in time [7, 21]. The parameter δ facilitates containing the second term over 0 and is set empirically to 1000. The resulting network is in the scale of 1 million edges, because their overwhelming majority had zero weight. We perform IM using SIMPATH with pruning parameter $\eta = 0.01$.

4 The DIFFUGREEDY Algorithm

In this section, we propose a new influence maximization algorithm that utilizes the diffusion cascades in the train set to extract a seed set. The basic idea is to use the standard SIMUGREEDY algorithm [11], but substitute the candidate seed's influence spread estimation, with a summary of the seed's diffusion cascades. The algorithm can iteratively build an influence spread network, using the most suitable seed in each iteration and its final size represents the cumulative influence spread of the seed set. The main difference with SIMUGREEDY lies in the calculation of a seed's influence spread. In the original algorithm, it is computed using Monte Carlo simulations of a diffusion process that starts from that seed. In our case, we substitute this with an estimate from the list of diffusion cascades that the node has initiated in the train set, for brevity's sake the node's train cascades. We define a node's influence spread as the node's train cascade that provides the highest marginal gain. Initially we experimented by using the train cascade with the median marginal gain, as a more objective estimate, or the number of distinct nodes in the node's train cascades. These approaches performed worse in our experiments, hence we kept the definition based on the cascade with the highest marginal gain. If our definition of influence spread is submodular, we can retain the $(1 - 1/e)$ theoretical guarantee [11]. Below we provide the proof of submodularity and the algorithm.

Theorem 1. *Computing the influence spread of a candidate seed using the diffusion cascade that maximizes the set's marginal gain, is a submodular function.*

Proof. The influence spread of a node u at step t is represented by its train diffusion cascade with the highest marginal gain at that step, c_u^t . If another node v is added to the seed set at step t , the influence spread of c_u^t at $t + 1$ can only be diminished, due to overlaps with v 's influence spread. If c_u^t has a high overlap with v , then another one of u 's cascades will be used, let c_u^{t+1} , in order to maximize marginal gain at step $t + 1$. Since we always choose the cascade with the maximal marginal gain, c_u^t had larger marginal gain than c_u^{t+1} at step t . In addition, by definition, a cascade's marginal gain can only diminish or stay the same as the seed set grows. Thus, c_u^{t+1} at $t + 1$ will always have smaller marginal gain than what c_u^t had at t , whether it is the same cascade or not, which is to be shown.

Algorithm 1 FIND THE SEED'S CASCADE WITH MAXIMUM MARGINAL GAIN

```

procedure MARGINALGAIN(final_spread,seed_cascades)
2:   set max_gain  $\leftarrow -1, casc_idx  $\leftarrow -1
    for casc  $\leftarrow 0$ ; casc  $< \text{size}(\text{seed\_cascades}); casc  $\text{++}$  do
4:     marginal_gain  $\leftarrow \text{size}(\text{final\_spread} \cup \text{seed\_cascades}[\text{i}])
        if marginal_gain  $> \text{max\_gain}$  then
6:            max_gain  $\leftarrow \text{marginal\_gain}
            casc_idx  $\leftarrow \text{i}$ 
8:   return max_gain, casc_idx$$$$$ 
```

Algorithm 2 INFLUENCE MAXIMIZATION USING NODES' DIFFUSION CASCADES

```

procedure DIFFUGREEDY(node_cascades,seed_set_size)
2: seed_sed  $\leftarrow [], final_spread  $\leftarrow \emptyset
    while size(seed_set)  $< \text{seed\_set\_size}$  do
4:     set max_seed  $= -1, max_gain  $= 0, max_cascade  $= -1
        for seed  $= 0$ ; seed  $< \text{size}(\text{node\_cascades}); seed  $\text{++}$  do
6:         marginal_gain, cascade_idx = MARGINALGAIN(final_spread, node_cascades[seed])
            if marginal_gain  $> \text{max\_gain}$  then
8:                max_gain  $\leftarrow \text{marginal\_gain}
                max_seed  $\leftarrow \text{seed}$ 
10:               max_cascade  $\leftarrow \text{cascade\_idx}$ 
                final_spread  $\leftarrow \text{final\_spread} \cup \text{node\_cascades}[\text{max\_seed}][\text{max\_cascade}]
12:               seed_sed.insert(max_seed)
                    delete node_cascades[max_seed]
14:   return size(influence_spread)$$$$$$$$ 
```

The complexity of DIFFUGREEDY is $O(KVC)$, where K is the size of the seed set, V is the number of nodes that initiated a cascade and C is the average size of cascades. Since the marginal gain estimation is submodular, we can utilize CELF, which does not change the worst case complexity, but has proven to accelerate greedy [13]. We do not add the Diffusion CELF here due to space limitations,

but its derivation is similar to the way CELF is derived from SIMUGREEDY [13]. It should be noted here, that although SIMUGREEDY and CELF estimate the same marginal gains, the final seed set may differ, because of multiple nodes having the same gain and each algorithm choosing based on different seed orders. This results in Diffusion CELF having a slightly inferior performance in our experiments.

5 Experiments

5.1 Data

We apply our methodology in the Sina Weibo dataset [27], a network consisting of more than 1.7 million nodes and 0.4 billion edges, accompanied by a set of 300,000 retweet cascades. The actual expanding follower network is given for a time span of 32 days (2012.9.28 to 2012.10.29), throughout which, almost 10 million new follow relationships occurred. The diffusion cascades are gathered by the most popular of the past 1,000 tweets of each node in the network, and they date back since the year 2009. Since our methodology relies on the intersection of the follower network and the retweet cascades, we cannot utilize the cascades before 2012.9.28, as we do not know the structure of the follower network at that time. Concurrently, we can not use the nodes in the network that are not present in the retweet cascades, because we have no information about their interactions. We thus extract the diffusion cascades of those 32 days and remove nodes from the network that are not present in these cascades. That results in a network of 641,575 nodes, with 95,272,167 edges and 18,652 cascades. We split the cascades into training (14,555) and testing (4,097).

5.2 Results

As mentioned above, our evaluation method is based on the number of distinct nodes influenced (DNI) by the seed set in the test set. We consider influenced, every node that participates in a test set diffusion initiated from one of the predicted seeds. Since we measure the size of the distinct set, potential overlaps between diffusions of different seeds are taken into account. The DNI of each method are shown in Fig. 2 and Table 1 shows the average DNI of each method throughout all seed set sizes. DIFFUGREEDY and DIFFUCELF clearly outperform the other approaches by a considerable gap. In addition, DIFFUCELF takes only 16 seconds, which is almost 40 times faster than k-core decomposition, and 1000 times faster than DIFFUGREEDY.

One important observation is the failure of the algorithmic IM approach. Only 15 out of the 100 seeds selected by IMM had started at least one cascade in the test set, and their spread was scant. This can be attributed to a lot of follow relationships not translating into retweets, a well-known phenomenon [2] plays a vital role in social influence analysis. Regarding SIMPATH in the databased weighted network, its failure might stem from the data. More specifically, we

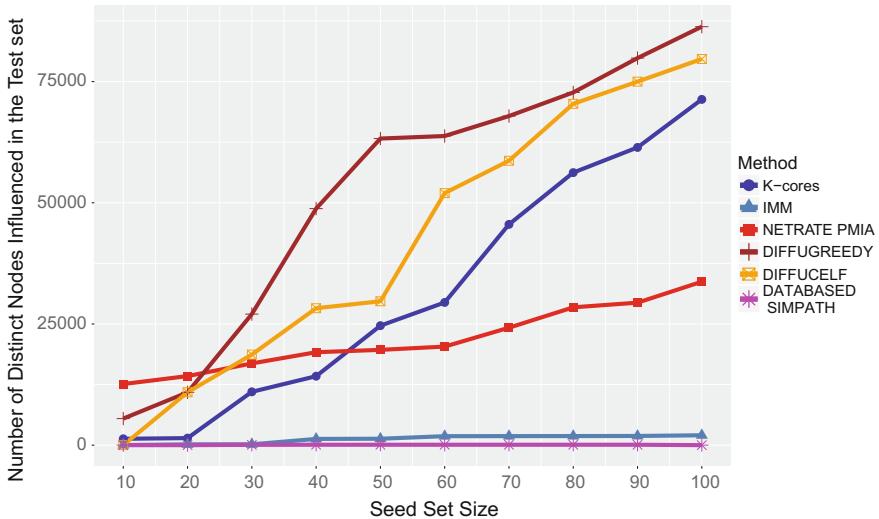


Fig. 2. Number of distinct nodes influenced (DNI) in the test set by the seed set of each method. Method labels are ordered based on their average DNI.

Table 1. Average evaluation metrics and computational time for each method

Method	DNI	Computational Time (sec)
DIFFUGREEDY	52,600	16,504
DIFFUCELF	42,325	16
K-cores	31,657	632
NETRATE PMIA	21,863	27,966 ^a
IMM	1,248	104,078 ^b
DATABASED SIMPATH	56	96,908 ^c

^a Preprocessing took 999, NETRATE 26398 and PMIA 569

^b Extraction and weighing took 103898 and IMM 180

^c Extraction took 2981, weighing 93898 and SIMPATH 29

observed that the follower networks of retweet cascades are extremely sparse i.e. the ratio between the number of edges and nodes is 0.84. This happens because during crawling, 100 users were chosen at random and their follower ego-network were crawled up to three hops. Subsequently, last 1000 tweets of each user are retrieved, each one containing a list of retweets. This list of retweets is filtered to contain only nodes that are in the crawled set. However, the follow relationships indicating how the tweet reached a node can be lost and as a result, the cascades are comprised of mostly unconnected nodes. Therefore, the DATABASED weighing results in very few retained edges. Finally, the lack of success of the diffusion network approach could be attributed to the substantial mismatch with the actual follower network. Less than 1% of the inferred edges

were actual follow edges. Even though as mentioned above, there were a lot of follow relationships in the diffusion cascades missing, it is safe to assume that a large part of the diffusion network was not based on direct follow edges, but rather on higher order relationships. These relationships although useful, are not as stable as direct ones i.e. an active follower is more likely to retweet than a follower's active follower. The retweets in the test set might consist of mostly followers, which caused this approach's deficiency. The code, along with instructions to reproduce the experiments can be found on github¹.

6 Conclusion

As network science drifts towards data-driven approaches and increasingly more networks are accompanied by diffusion cascades, we have to reconsider our view of many important problems. In this study, we address influence maximization on a large scale social network. We employ multiple state-of-the-art methods, each exploiting a different aspect of the dataset, and propose an algorithm that outperforms them. In addition, we utilize an evaluation methodology based on actual diffusion cascades, as a more realistic alternative to epidemic simulation models. For future work, we plan to examine methods based on machine learning to derive the seed set. More specifically, while numerous neural network algorithms have been developed recently for influence or outbreak prediction [14, 17], the problem of influence maximization remains unaddressed. This is a promising path that we would like to explore further in subsequent steps.

References

1. Bourigault, S., Lamprier, S., Gallinari, P.: Representation learning for information diffusion through social networks: an embedded cascade model. In: Proceedings of the Ninth ACM International Conference on Web Search and Data Mining, pp. 573–582. ACM (2016)
2. Cha, M., Haddadi, H., Benevenuto, F., Gummadi, P.K., et al.: Measuring user influence in twitter: the million follower fallacy. ICWSM **10**(10–17), 30 (2010)
3. Chen, W., Wang, C., Wang, Y.: Scalable influence maximization for prevalent viral marketing in large-scale social networks. In: Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 1029–1038. ACM (2010)
4. Cohen, E., Delling, D., Pajor, T., Werneck, R.F.: Sketch-based influence maximization and computation: scaling up with guarantees. In: Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management, pp. 629–638. ACM (2014)
5. Du, N., Song, L., Rodriguez, M.G., Zha, H.: Scalable influence estimation in continuous-time diffusion networks. In: Advances in Neural Information Processing Systems, pp. 3147–3155 (2013)
6. Gallos, L.K., Song, C., Makse, H.A.: Scaling of degree correlations and its influence on diffusion in scale-free networks. Phys. Rev. Lett. **100**(24), 248,701 (2008)

¹ <https://github.com/GiorgosPanagopoulos/DiffuGreedy-Influence-Maximization>.

7. Goyal, A., Bonchi, F., Lakshmanan, L.V.: Learning influence probabilities in social networks. In: Proceedings of the Third ACM International Conference on Web Search and Data Mining, pp. 241–250. ACM (2010)
8. Goyal, A., Bonchi, F., Lakshmanan, L.V.: A data-based approach to social influence maximization. Proc. VLDB Endow. **5**(1), 73–84 (2011)
9. Goyal, A., Lu, W., Lakshmanan, L.V.: Simpath: an efficient algorithm for influence maximization under the linear threshold model. In: 2011 IEEE 11th International Conference on Data Mining (ICDM), pp. 211–220. IEEE (2011)
10. Jendoubi, S., Martin, A., Liétard, L., Hadji, H.B., Yaghlane, B.B.: Two evidential data based models for influence maximization in twitter. Knowl. Based Syst. **121**, 58–70 (2017)
11. Kempe, D., Kleinberg, J., Tardos, É.: Maximizing the spread of influence through a social network. In: Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 137–146. ACM (2003)
12. Kitsak, M., et al.: Identification of influential spreaders in complex networks. Nat. Phys. **6**(11), 888 (2010)
13. Leskovec, J., Krause, A., Guestrin, C., Faloutsos, C., VanBriesen, J., Glance, N.: Cost-effective outbreak detection in networks. In: Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 420–429. ACM (2007)
14. Li, C., Ma, J., Guo, X., Mei, Q.: Deepcas: an end-to-end predictor of information cascades. In: Proceedings of the 26th International Conference on World Wide Web, pp. 577–586. International World Wide Web Conferences Steering Committee (2017)
15. Malliaros, F.D., Rossi, M.E.G., Vazirgiannis, M.: Locating influential nodes in complex networks. Sci. Rep. **6**, 19,307 (2016)
16. Pei, S., Morone, F., Makse, H.A.: Theories for influencer identification in complex networks. Complex Spreading Phenomena in Social Systems, pp. 125–148. Springer, Berlin (2018)
17. Qiu, J., Tang, J., Ma, H., Dong, Y., Wang, K., Tang, J.: Deepinf: modeling influence locality in large social networks. In: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2018) (2018)
18. Rodriguez, M.G., Balduzzi, D., Schölkopf, B.: Uncovering the temporal dynamics of diffusion networks. arXiv preprint [arXiv:1105.0697](https://arxiv.org/abs/1105.0697) (2011)
19. Rodriguez, M.G., Schölkopf, B.: Influence maximization in continuous time diffusion networks. arXiv preprint [arXiv:1205.1682](https://arxiv.org/abs/1205.1682) (2012)
20. Rossi, M.E.G., Vazirgiannis, M.: Exploring network centralities in spreading processes. In: International Symposium on Web Algorithms (ISWAG) (2016)
21. Saito, K., Kimura, M., Ohara, K., Motoda, H.: Learning continuous-time information diffusion model for social behavioral data analysis. In: Asian Conference on Machine Learning, pp. 322–337. Springer, Berlin (2009)
22. Saito, K., Nakano, R., Kimura, M.: Prediction of information diffusion probabilities for independent cascade model. In: International Conference on Knowledge-Based and Intelligent Information and Engineering Systems, pp. 67–75. Springer, Berlin (2008)
23. Tang, Y., Shi, Y., Xiao, X.: Influence maximization in near-linear time: a martingale approach. In: Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data, pp. 1539–1554. ACM (2015)

24. Tang, Y., Xiao, X., Shi, Y.: Influence maximization: near-optimal time complexity meets practical efficiency. In: Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data, pp. 75–86. ACM (2014)
25. Vespignani, A.: Modelling dynamical processes in complex socio-technical systems. *Nat. Phys.* **8**(1), 32 (2012)
26. Xie, M., Yang, Q., Wang, Q., Cong, G., De Melo, G.: Dynadiffuse: a dynamic diffusion model for continuous time constrained influence maximization. In: AAAI, pp. 346–352 (2015)
27. Zhang, J., Liu, B., Tang, J., Chen, T., Li, J.: Social influence locality for modeling retweeting behaviors. In: IJCAI, vol. 13, pp. 2761–2767 (2013)



Predicting Information Diffusion in Online Social Platforms: A Twitter Case Study

Kateryna Lytvyniuk¹, Rajesh Sharma^{1(✉)}, and Anna Jurek-Loughrey²

¹ University of Tartu, Tartu, Estonia

kateryna.lytvyniuk@ut.ee, rajesha.sharma@ut.ee

² Queen's University Belfast, Belfast, UK

a.jurek@qub.ac.uk

Abstract. Online social media has become a part of everyday life of modern society. A lot of information is created on these platforms and shared with the community continuously. Predicting information diffusion on online social platforms has been studied in the past by many researchers as it has its applications in various domains such as viral marketing, news propagation etc. Some information spreads faster compared to others depending on topic of interest of the online users. In this work, we investigate the information diffusion problem using Twitter data as a use case study. We define tweet popularity as number of retweets any original message receives. In total we extracted 27 features which can be categorised into content, user, sentiment and initial retweeting behaviour for creating our prediction model. We study the problem of predicting as a multiclass prediction task. Three datasets from Twitter about three different topics are collected and analysed for building and testing various models based on different machine learning algorithms. The models were able to predict up to 60% of overall accuracy and an F1 score of 67% is obtained. The models are created using one of the dataset and tested on all the datasets, which shows that the model is robust enough to handle different topics.

Keywords: Online social networks · Information diffusion · Machine learning · Data analytics

1 Introduction

Social media platforms allow Internet users to create and consume content in a very convenient and quick way. The influence of such online networks is very high as the Internet has become the primary source of new information in the present society. Understanding the information diffusion processes on these networks may help addressing many real world challenges such as investigation and prevention of terrorism activity [15]. The data on social media can also be analysed with an objective of observing the trends of elections results [14], correlating

events between social media platforms such as Twitter and stock market [13]. In this work, we investigate the information diffusion problem using Twitter, which is one of the most popular social platform used by internet users. According to recent updates there are 335 million active users on Twitter¹. Messages (referred to as Tweets) appear continuously and spread according to the interest of the Twitter's users. Each message can be forwarded by other users, also called as retweeting, or it can be liked or commented by others. These all activities helps in spreading the message through the network.

Various studies have been performed by researchers, that aimed to analyse information diffusion in online social platforms and to understand why certain messages are more popular than others [1–3]. People often express their opinions about specific topic or events which are most relevant to them or related to present real world events. Some studies have been very specific in investigating the diffusion of information with respect to news [9], advertising campaigns [10]. As opposed to previous research, our study is about predicting information diffusion and is topic independent. In particular, we investigate following research questions (RQ):

1. RQ 1: How can information diffusion be modelled?
2. RQ 2: What features are the most discriminative for the diffusion prediction?
3. RQ 3: How well a message diffusion can be predict using the identified features?
4. RQ 4: How initial retweet activity can help in predicting tweet popularity?
5. RQ 5: Is it possible to predict tweet popularity in a coming time window (for example an hour) based on tweet's behaviour in previous time window?

The first three research questions have been extensively studied in the past, however, RQ4 and RQ5 have not gained much attention from researchers. We study these five research questions by first collecting dataset from Twitter about three topics namely, (1) *Cryptocurrency*, (2) *Smartphone brands*, (3) *Football*. We created our model using the tweet data associated with *Cryptocurrency*. We extracted in total 27 features which are either related to (1) the users who are tweeting , (2) the content of the tweets, (3) the sentiment associated with the tweets and (4) the initial behavior of the tweets, which basically calculated the number of times a tweets attracts the retweet in some initial time period. We study the importance of various features and then incorporate most relevant features into our models for predicting the retweeting behavior.

With respect to RQ 5, we are interested in the tweet diffusion for a particular tweet T_w . In other words, given the tweet diffusion pattern for a certain time period t , what would be the retweeting behavior for the tweet T_w in the next time window $t + \delta$. The problem in the past has been studied in different forms such as by calling it popularity [1] or interestingness [2]. In our study, for modelling the prediction problem, we used four different machine learning algorithms namely (1) Random Forest, (2) Tree Bag, (3) Gradient Boosting Machine and (4)

¹ <https://www.statista.com/statistics/282087/number-of-monthly-active-twitter-users/>

Xgboost. We studied this problem as a multiclass classification problem inline with [1]. We created different sets of classes where each class represents a range of retweet numbers. We then predicted the class (or the range) for each tweet. The model created using the *Cryptocurrency* topic, was tested on the other two datasets that is *Smartphone brands* and *Football*. Using our methodology, we were able to predict up to 60% of overall accuracy and a F1 score of 67 was obtained.

The rest of the paper is organized as follows. In Sect. 2, we present related work. In Sect. 3, we describe our dataset and in Sect. 4 methodology is described. In Sect. 5 we discuss results of our research investigations. Section 6 concludes the work by also discussing the future works.

2 Related Work

In this section, we discuss relevant literature which are either closely related or have some overlaps with respect to our research. In [3], the authors investigated how topics spread through network structures in particularly analysing the impact of the presence of @username in tweets, which basically means involving users more proactively in the diffusion process. They concluded that a topic might have a different propagation efficiency at different time stages of its life-cycle.

A lot of previous work has focused on popularity prediction problem using the number of future retweet as a measure of the popularity. In [1] the authors tried to find out what different factors influence information propagation on Twitter. They focused on content features of the tweets as well as the user features who initiates the tweets. Since it is difficult to predict the exact number they put forward the concept of multiclass classification for the prediction task. In contrast to [1] in [2] the researchers predicted the retweeting behavior using tweet characteristics only. They discovered that there is not such a strong correlation between number of followers of a user initiating the tweet and retweet count. It was concluded that messages were more likely to be retweeted if they were about a general popular topic compared to a specific personal topic.

Most of the studies which have modelled information cascades have either focused on user properties, such as importance of the users [6], or on network properties such as edge growth rate, diameter and degree distribution [7]. Some other works have analysed the diffusion by following explicit paths of the propagation [4, 5, 16]. In [4], after collecting data of tweets and users activities, Tweet-Trees are created, which represent tweet propagation in the network. Nodes of a tree represent followers of the user that has retweeted the initial message. Authors used each tweet's linguistic features and a profile of initial creator of the tweet to make the prediction. In a similar approach, in [5], authors analysed information cascades from one user to another from a stream of tweets and the social graph. They not only focused on tweets but also considered other types of information over a social network e.g. links or hashtags. In addition, they suggested methods to deal with the missing data, with regards to constructing the information cascades.

In [16] the authors used the propagation of Twitter posts to model the influence of users. As oppose to the other papers, they considered reposting, rather than re-tweeting, to indicate the diffusion level. For every initial post an influence tree (cascade) was generated, where the number of users included in the tree defined the influence score of the seed post. Two types of features were used as predictors: user-related attributes (e.g. number of followers) and characteristics related to user's past performance. The authors also investigated the role of a post's content in the propagation process. According to the authors' findings, the average number of past reposts by user's immediate followers and the number of user's followers can be used to predict future influence of a user (propagation of the user's posts). The content of a post was not found to improve the predictive performance.

This work is different from the application oriented work such as prediction of tweet popularity related to breaking news analysis [9], or study of advertising campaign strategy based on information cascades in Twitter [10], or study of brand popularity prediction in social networks [11]. Our work is more close to that of [1, 2, 8] however, compared to all the previous studies which have mainly considered user and content features, we also explored the sentiments present in the tweets and the initial propagation of the tweets in predicting the information diffusion.

3 Dataset

In this section, we describe the dataset we used for our analysis. We wrote a Python script for collecting data from Twitter, using Streaming API in JSON format. The collected data is related to three different topics: *Cryptocurrency*, *Smartphone brands* and *Football*. Description of the collected data is provided in the Table 1.

Table 1. Dataset description

Dataset	# tweets	# original tweets	Keywords	Duration	Train or Test	Description
Cryptocurrency	3,110,500	1,606,696	cryptocurrency, bitcoin, blockchain, etherium	Jan-Feb, 2018	Both	Tweets about cryptocurrency trends.
Smartphone brands	601,380	340,504	'Samsung', 'Huawei', 'Xiaomi', 'iPhone', 'Lenovo', Nokia, 'LG', 'smartphone'	April, 2018	Test	Tweets about some of most popular smartphones brands.
Football	192,593	103,755	football, World Cup	June, 2018	Test	Tweets about football and in particular 2018 FIFA World Cup.

Each Twitter message is represented in JSON format and consist of many attributes. Besides the message content, metadata of tweet is extracted as well.

This data includes user profile, location, statuses, counts of entities (such as special symbols, links) and language. User data consists of user profile characteristics. The most relevant for our research are: number of followers, number of friends and date of account creation. Retweeted status has fields with original tweet metadata including its author profile data. Entities of a tweet contain some additional information about a message, such as lists of hashtags, urls, user mentions and symbols. These characteristics can also have an impact in the research. Retweet is a repost of another message of user of Twitter on someone's profile. With retweet information, it is possible to see how users interact and what information they share.

Data preprocessing: We perform various data cleaning steps before analysing the data. Firstly, we remove the special characters as they can impact features like length of the tweet as well the sentiment analysis. We also convert the time into Unix timestamp in seconds. For the missing values, observations that have most of their attributes empty are removed from the dataset. In other cases, if there is no information about one or few numeric features, zeros are inserted instead. For instance, user simply may not have any friends or followers. We also removed non-English tweets from out datasets. As expected, the collected datasets was very imbalanced, as there are a lot of messages that have zero retweets. In our dataset only 20% *Cryptocurrency*, 22% *Smartphone brands*, and 25% *Football* tweets were retweeted. In order to help prevent overfitting, we decided to downsample the majority class to make the same number of observations in each class, to avoid the problem of imbalanced dataset.

4 Methodology

In this section, we describe our proposed approach to information diffusion analysis and prediction. The number of retweets is the most indicative measure of information diffusion [1], thus, we use it as a target variable in our prediction models. We started with the regression approach of predicting the closest value for the retweets however, it is not easy to predict the exact number [1]. Thus, we performed two types of classification tasks. Firstly, the binary classification where we only predicted if a tweet will be retweet or not. Secondly, in multiclass classification, we created classes and predicted for a certain tweet to which types of a class it belongs. In addition, we normalized this data so that we predict the retweet number after the same period of time for each original tweet. In this case, period of time is the time range from the moment each message had appeared till some point of time in the future. The time period of 7 days was chosen for further analysis. Each tweet which has been retweeted several times has a different popularity window in time and eventually the retweeting process stops. Tweet lifespan of various tweets in our dataset shows scale-free pattern and the bigger set of tweets is concentrated below 50 h duration which is around two days.

4.1 Class Labelling

In order to apply any of the supervised machine learning models, depending on the type of classification, we need to label the data appropriately or in other words, make classes from our numerical target variable. We created two classes: “retweeted” and “no retweet” for binary classification task, and 4 classes for multiclass task. In case of Binary classification, for each original tweet, we calculate the number of retweets for it. If the value is more than 1, we assign it “retweeted” otherwise “no retweet”. For multiclass classification, we created 4 classes based on the number of retweets: Very Low (0–10 retweets), Low (11–90 retweets), Medium (91–170 retweets), High (more than 170 retweets). There are more samples with less retweets and few samples with high value of retweet count which are more difficult to predict.

4.2 Feature Extraction

In this section, we describe various features that we considered for our prediction model.

1. User features: User profile information is very likely to be influential on how many times a tweet of a user will be retweeted. We selected the following most intuitive features:

- Followers count: Number of people who follow a user.
- Account age: Period of time calculated as difference between the time account was created and the time of tweeting a message.
- Listed count: Number of public lists that a user is a member of.
- Verified: Indicator if a user is verified or not. Binary variable.
- Friends count: Number of friends of a user.
- Statuses count: Number of tweets posted by a user.

2. Content features: There are many features that can be extracted from tweet’s text. Some of them such as number of user mentions, hashtags and URLs lists are given in a tweet’ metadata. Following is the list of all the content features:

- Tweet length: Number of symbols in a tweet, including spaces.
- User mentions: Number of user mentions with @ notation.
- Hashtags: Number of hashtags.
- URLs: Number of URLs.
- Exclamation and question marks: Number of exclamation and question marks.

3. Sentiment features Sentiment analysis identifies emotions, for example, a text could have positive or negative sentiment in it. To find the role of sentiments in tweet diffusion nine different emotions and sentiments were defined and were analysed for their presence in each tweet. Sentiment extraction was done using Syuzhet package. The list of the sentiment (and emotions) includes *negative*, *positive*, *trust*, *joy*, *anger*, *disgust*, *sadness*, *fear*, *anticipation*, *surprise*.

4. Initial behaviour features The initial behavior of a tweet could also be an influential parameter in predicting tweets's diffusion in the future. In other words, the hypothesis is based on the fact that if a tweet attracts a lot of retweet in some “initial time period” then it is highly likely to attract more tweets in the near future before eventually the retweeting about the original tweet dies out. Following features are used for characterising the initial retweet behaviour:

- Current retweet count: Number of retweets of the message happened in the given initial time period
- Time alive of message: Period of time since the original message appeared till the last retweet in the given initial time period
- Tweet rate: Number of retweets in the time frame of one hour divided by time alive of message in the given initial time period
- Mean difference between retweets: Mean difference between retweets in the given initial time period
- Max difference between retweets: Max difference between retweets in the given initial time period

In addition, network of user followers can also contribute in tweet diffusion. Subfollowers are the number of people who follow users who retweeted an original tweet in the given initial time period. We added this feature using information of each user who have retweeted a message and added these features to an original tweet data.

4.3 Splitting and Cross-Validation

As for any standard prediction task in machine learning we split the data in training (80% of the data) and testing sets (20% of the data). Considering that the information propagation diffusion prediction task is oriented on future popularity of a message it seems more reasonable to split by time when tweets appeared. However, it was studied in [12] that there is no significant difference between random and chronological splitting methods for this particular prediction task. For our analysis we used k-fold cross-validation technique, which randomly splits the data into k-samples, and trains model multiple times so that each k-th fold serves as a test dataset. For our experiments we used k = 5 and we repeat each fold for 5 times for evaluating our model.

5 Evaluation

One of the important steps in working with predictive algorithms is to measure and compare obtained results. We evaluate our model using standard machine learning metrics such as confusion matrix, accuracy, F1-measure. We are not explaining these metrics due to space limitation and also, these metrics are well known in data science community.

5.1 Results

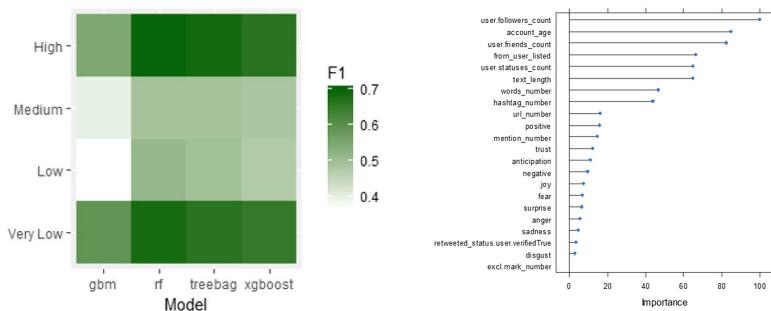
As described in the Sect. 4.1 instead of predicting the exact value of retweets, we divided the total retweets into four classes. We used two bagging based methods namely (1) Random Forest and, (2) Treebag and two boosting based methods namely (1) Gbm (Gradient Boosting Machine), and, (2) Xgboost algorithms for the prediction tasks. We train the models using training data of cryptocurrency dataset and then evaluated it on the test set of cryptocurrency as well as on the other two datasets. The Table 2 shows the summary of results for multiclass prediction task.

Table 2. Performance results of classification

Feature set	Random forest		Gbm		Xgboost		Treebag	
	AUC	F1	AUC	F1	AUC	F1	AUC	F1
	Cryptocurrency dataset							
User + Content	0.6024	0.6741	0.4820	0.5830	0.5013	0.5991	0.5879	0.6816
User + Content + Sentiment	0.6037	0.6898	0.4822	0.5924	0.5110	0.6638	0.5907	0.6682
	Smartphone brands dataset							
User + Content + Sentiment	0.6011	0.6823	0.4696	0.581	0.5014	0.6532	0.5827	0.6777
	Football dataset							
User + Content + Sentiment	0.5799	0.6645	0.4683	0.5745	0.5010	0.6612	0.5789	0.6725

From the summary Table 2, we can conclude that Random Forest and Treebag algorithms performed better than Xgboost and Gbm. Random Forest showed the best on the performance metrics. Bagging algorithms confirmed their good performance by each class as well. Other algorithms showed worse performance overall (as seen from the Table 2) and by each class. However, the edge classes (High and Very low) are predicted quite good in all algorithms up to 72% of accuracy of one class. To look at the performance of each class we used confusion matrix (CM) which is a good visualisation for spotting the prediction errors. Figure 1 shows the CM for the two best algorithms.

Feature importance: Since there are many performance metrics that can be obtained from confusion matrix (CM) we focus more on ones selected for this prediction task. Figure 2a shows F1-measure for each class and model and it is evident that Random Forest performs the best. However, looking at each class separately, we can conclude that they have not brought significant improvement. They could make a model more stable so we need to see the feature importance values. Figure 2b shows ordered importance of the best model of used features for the Random Forest. Other algorithms have their own order of important features that is not presented here due to space limitation but it is worth to note that first 5-7 features are the same across all the models (user related features) and group of first 8 features have relatively high level of significance comparing to others. These are 5 User features and 3 three Content features. In addition, positive sentiment is the most important from Sentiment feature set for the prediction.

**Fig. 1.** Confusion matrix**Fig. 2.** Confusion matrixes of predictions using different range initial behaviour features in Cryptocurrency dataset

5.2 Predicting Tweet Popularity Using Initial Retweet Behaviour Features

This part of analysis requires usage of initial behaviour features introduced in the Subsect. 4.2. These features most likely boost the performance for multiclass prediction as they could give more accurate result for each class of target variable. Moreover, it was studied how performance changes with increase in the initial time range. Analysing this, we obtained the point of time range after which there is no significant change in performance of prediction model. We defined the following thresholds for the analysis: 1, 2, 3, 4, 5, 10, 30, 60 min (see Table 3). Since the Random Forest algorithm showed the best performance and this type of analysis is more complex than previous one, we decided to compare output for our datasets using only Random Forest.

Table 3. Performance of multiclass prediction with initial behaviour features

Dataset	Cryptocurrency		Smartphone brands		Football	
	Acc	F1	Acc	F1	Acc	F1
Initial behaviour time range						
1 min	0.626	0.726	0.611	0.685	0.579	0.680
2 min	0.643	0.728	0.631	0.692	0.587	0.685
3 min	0.651	0.729	0.646	0.696	0.594	0.689
4 min	0.655	0.746	0.653	0.710	0.602	0.697
5 min	0.656	0.747	0.655	0.717	0.606	0.702
10 min	0.656	0.748	0.658	0.724	0.611	0.709
30 min	0.659	0.752	0.661	0.729	0.614	0.713
60 min	0.674	0.764	0.667	0.738	0.622	0.722

As expected, the performance increases with increasing the initial time range. Compared to 1 min, in 60 min period, accuracy increased by 4.8% and F1-measure by 3.8% in *Cryptocurrency* testing set. In addition, we can see that even 1 min initial behaviour features improves the accuracy of model by 2%. *Smartphone brands* and *football datasets* did not perform well but still we can observe similar improvement trend. Certainly, different datasets have different retweet activity. From the confusion matrices (Fig. 3), we can see the improvement of prediction by each class. It is clearly seen that the initial time features strongly affect the result, especially detecting well Low and Medium classes that are more difficult to distinguish.

Feature importance: From the previous experiments we found out several most significant features for prediction. Considering the fact that initial time features improved the results we can see how the order of importance changes. Obviously, current number of retweets gets the first place with increasing of the initial time range. Most of the content and user features like number of followers, number of friends, account age, if user listed or not remain important (in top 27

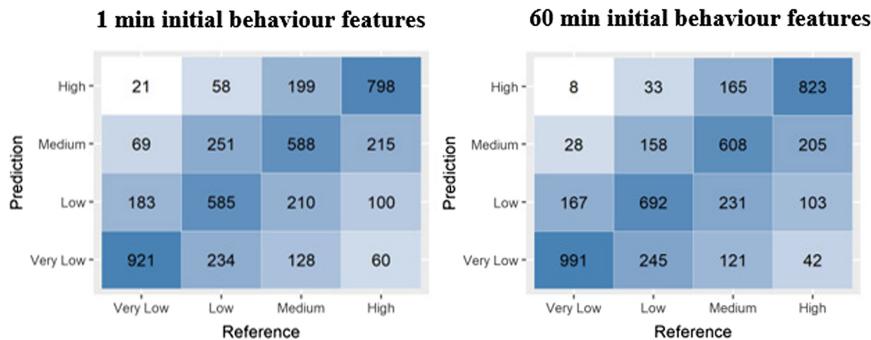


Fig. 3. Confusion matrixes of predictions using different range initial behaviour features in Cryptocurrency dataset

features). Some initial behaviour time features in addition to current count of retweets ranked higher positions (such as subfollowers) in 60 min time interval.

Another approach is predicting behaviour of certain message in the next period of time in the future. For this task we decided to use one-hour time frame. To evaluate it, we first organize the dataset so that the data is divided into one-hour time frames for each unique tweet. Target variable (retweet number) was taken using information of the next hour. In this task, the goal was to predict what would happen in the next hour based on the information about retweet activity from the previous hour. For this purpose, the same features are used as in previous approach of min by min analysis, but they were created from reorganized one-hour-frame data. Concerning Initial Behaviour features, they were created using whole one-hour time frame data for each original tweet. Therefore, they are used as retweet behaviour characteristics of previous hour. Training and testing sets were adjusted according to this task observations that already belong to class High in previous hour were removed. Therefore, we could see if observations of Very Low, Low or Medium class can move to class with bigger number of retweets. The Table 4 gives an overview of the results.

Table 4. Performance of multiclass prediction of next hour based on previous hour

Dataset	Random Forest		Gbm		Xgboost		Treebag	
	Acc	F1	Acc	F1	Acc	F1	Acc	F1
Cryptocurrency	0.571	0.678	0.513	0.567	0.522	0.581	0.543	0.662
Smartphone brands	0.563	0.669	0.504	0.558	0.519	0.580	0.538	0.650
Football	0.544	0.654	0.502	0.551	0.513	0.575	0.533	0.645

The performance of the models is worse comparing to multiclass models in previous sections. We can observe that it is difficult to predict tweet popularity

using this approach even with four number of classes. This might be caused by the retweeting behaviour of our training data from *Cryptocurrency* dataset or due to not enough amount of training data.

6 Conclusion

In this work, we analysed Twitter messages and extracted various features for predicting the retweet trends. We extracted in total 27 features, broadly categorized into namely (i) Content, (ii) User, (iii) Sentiment and, (iv) Initial Behaviour and analysed their impact on model prediction results. We performed multiclass prediction task to understand approximately how much retweet a tweet can attract. Our approach showed decent performance using Content, User and Sentiment features, that is an overall accuracy of 60%. We also analysed retweet behaviour in the first minutes of tweet existence and find out how it affects the prediction power of the model. We used all sets of features for this task and discovered that having information even of 5 minutes is enough to increase the overall accuracy value of 5%.

We have multiple future directions towards this work. We believe that larger dataset certainly would improve the stability and effectiveness of models. We would like to study if people retweet more when they see that a message is already popular. We would also like to include more features such as by including information about all friends and followers of a tweet's originator and by improving sentiment analysis by including emojis etc. We would also like to predict different measure of popularity such as predicting how many comments a tweet can get.

Acknowledgements. This work is supported by H2020 framework project, SoBig-Data, grant number 654024.

References

1. Hong, L., Dan, O., Davison, B.D.: Predicting popular messages in twitter. In: Proceedings of the 20th International Conference Companion on World Wide Web, vol. 46, no. 3, pp. 57–58 (2011)
2. Naveed, N., Gottron, T., Kunegis, J., Alhadi, A.C.: Bad news travel fast: a content-based analysis of interestingness on twitter. In: Proceedings of the 3rd International Web Science Conference, pp. 8:1–8:7 (2011)
3. Yang, J., Counts, S.: Predicting the speed, scale, and range of information diffusion in twitter. In: Proceedings of the Fourth International Conference on Weblogs and Social Media, vol. 2010, no. 10(2010)
4. Kafeza, E., Kanavos, A., Makris, C., Vikatos, P.: Predicting information diffusion patterns in twitter. In: 10th IFIP International Conference on Artificial Intelligence Applications and Innovations (AIAI), vol. 2014, no. 10 (2014)
5. Taxidou, I., Fischer, P.M.: Online analysis of information diffusion in twitter. In: Proceedings of the 23rd International Conference on World Wide Web, vol. 2014 (2014)

6. Kupavskii, A., Ostroumova, L., Umnov, A., Usachev, S., Serdyukov, P., Gusev, G., Kustarev, A.: Prediction of retweet cascade size over time. In: Proceedings of the 21st ACM International Conference on Information and Knowledge Management, vol. 2012 (2012)
7. Shafiq, Z., Liu, A.: Cascade size prediction in online social networks. In: IFIP Networking Conference (IFIP Networking) and Workshops, vol. 2017 (2017)
8. Chen, J., Li, H., Wu, Z., Hossain, M.S.: Sentiment analysis of the correlation between regular tweets and retweets. In: IEEE 16th International Symposium on Network Computing and Applications (NCA), pp. 1–5 (2017)
9. Wu, B., Shen, H.: Analyzing and predicting news popularity on twitter. *Int. J. Inf. Manag.* **35**(6), 702–711 (2015)
10. Okubo, K., Oida, K.: A successful advertising strategy over twitter. *Comput. Inf. Sci.* **10**, 10–22 (2017)
11. Mazloom, M., Rietveld, R., Rudinac, S., Worring, M., van Dolen, W.: Multimodal popularity prediction of brand-related social media posts. In: Proceedings of the 2016 ACM on Multimedia Conference, pp. 197–201 (2016)
12. Sarabchi, F.: Quantitative Prediction of Twitter Message Dissemination: A Machine Learning Approach. Technical University of Delft (2015)
13. Cazzoli, L., Sharma, R., Treccani, M., Lillo, M.: A large scale study to understand the relation between twitter and financial market. In: Third European Network Intelligence Conference (ENIC), pp. 98–105 (2016)
14. Adamic, L.A., Glance, N.: The political blogosphere and the 2004 U.S. election: divided they Blog. In: Proceedings of the 3rd International Workshop on Link Discovery, pp. 36–43 (2005)
15. Cohen, K., Johansson, F., Kaati, L., Mork, J.C.: Detecting linguistic markers for radical violence in social media. *Terror. Polit. Violence* **26**(1), 246–256 (2014)
16. Bakshy, E., Hofman, J.M., Mason, W.A., Watts, D.J.: Everyone's an influencer: quantifying influence on twitter. In: Proceedings of the Fourth ACM International Conference on Web Search and Data Mining, pp. 65–74 (2011)



Modelling and Analysis of Delayed SIR Model on Complex Network

Md Arquam^{1(✉)}, Anurag Singh^{1(✉)}, and Rajesh Sharma^{2(✉)}

¹ Department of Computer Science and Engineering, National Institute of Technology Delhi, New Delhi 110040, India
{arquam,anuragsg}@nitdelhi.ac.in

² Institute of Computer Science, University of Tartu, Tartu, Estonia
rajesh.sharma@ut.ee

Abstract. Complex networks are often used to model the network of individuals for analyzing various problems in human networks e.g. information diffusion and epidemic spreading. Various epidemic spreading models are proposed for analyzing and understanding the spreading of infectious diseases in human contact networks. In the classical epidemiological model, a susceptible person becomes infected instantly after getting in contact with the infected person. However, this scenario is not realistic. In real, a healthy person become infected with some delay in time not spontaneously after contacting with the infected person. Therefore, research is needed for creating more realistic models to study the dynamics of epidemics in the human population with delay. In order to handle delays in the infection process, we propose an epidemic spreading SIR (Susceptible-Infected-Recovered) model in human contact networks as complex network. We introduce time delay parameter in infection to handle the process to become a node infected after some delay. The critical threshold is derived for epidemic spreading on large human contact network considering the delay in infection. We perform simulations on the proposed SIR model on different underlying complex network topologies, which represents the real world scenario, e.g., random geometric network with and without mobile agents. The simulation results are validated in accordance with our theoretical description which shows that increment in delay decreases the critical threshold of epidemic spreading rate and the disease persists for the longer time.

Keywords: Complex network · SIR · Delay · Epidemic spreading · RGG

1 Introduction

Dynamics processes such as information diffusion [1] and epidemic spreading [2] in human networks is often studied through complex systems [3,4]. The underlying network in these system consists of nodes representing individuals and edges

as the connections. Specifically, the problem of the spreading of infectious diseases is an important topic as it can help in improving public health policies. Various models have been proposed for understanding the epidemic spread in human networks [2, 5]. In this course of research and development, to find the patterns of spread, Vespignani et al. [3] investigated to understand the effect of network structure for epidemic spreading.

The two most widely studied models to explain the epidemic spreading in the human population are (i) susceptible-infected-removed (SIR) [6] and (ii) susceptible-infected-susceptible (SIS) model [7]. The theoretical approach of the epidemiological model is based on the compartmental concept, in which the whole population is categorized into compartments. In SIR model, there are three compartments (i) Susceptible (denoted by S), (ii) Infected (I), and (iii) Recovered (R). Individuals in the susceptible compartment represent the group of healthy persons that can be infected when getting in contact with infected persons. If a person gets infected, it is transferred into the infected compartment. Finally, after recovery, an infected person is moved into the recovered compartment. Subsequently, various variations of these models have been proposed. For example, Takeuchi et al. [8] proposed a delayed SIR epidemic model to analyze the propagation of vector diseases with fixed incubation time. Later, Wang et al. [5] investigated the delayed SIR model with time delay in incubation with some carrying capacity. This carrying capacity decides the growth of the susceptible individual in absence of disease.

Most of the studies have not focused on the underlying network structure, which also plays an important part in epidemic spreading [2, 7]. In summary, except works like [9], these studies consider the underlying networks as static which is unrealistic from the real-world scenario where the network is dynamic. By the term dynamic we mean, the connections between individuals are not permanent. That is new connections are formed and some of the connections disappeared with time. Thus, rigorous studies are need of time for understanding the epidemic dynamics to show how dynamic human interaction impact epidemic spreading.

In this work, we study epidemic spreading in the human population using *SIR with delay*, by modelling human networks using static as well as dynamic networks. To incorporate the delay, we introduce a parameter, which we call as *delay in infection*. This parameter handles the real world scenario where people generally do not get infected instantly when they come in contact with the infected person. We also derive the critical threshold for epidemic spreading by considering the delay in the infection. In particular, we evaluated our model to investigate the effect of delay using the following three types of networks:

- 1 Random Graph:** Random Graph is proposed by Erdos and Renye [10]. A Random Graph is created when a node is connected with another node with a given connecting probability, p , it is represented by $G(N, p)$, where N is the number of nodes.
- 2 Random Geometric Graph:** A Random Geometric Graph (RGG) is created by randomly distributed nodes in 2-dimensional space and two nodes are

connected if the distance between two nodes exists in a given range, called connectivity radius, r . A Random Geometric Graph of N nodes is represented by $G(N, r)$ [11]. The random location of N nodes in \mathbb{R}^2 is represented by $\mathcal{X} = \{X_1, \dots, X_N\}$, where X_i are i.i.d. uniformly distributed random variable and represent the random geometry of i^{th} node. The nodes i and j are connected if $|X_i - X_j| \leq r$.

- 3 **Random Geometric Graph with Mobility:** A Random Geometric Graph with mobile agents is similar to RGG but in RGG, the position of all nodes changes uniformly, while in RGG with mobile agents, the position of only mobile nodes changes and rest of the nodes remain static for some given time interval.

The remainder of this paper is organized as follows: Sect. 2 discusses the current state of arts regarding traditional disease models and underlying network topology. Section 3 explains the effect of delay in the SIR epidemic model. Section 4 presents simulation and result analysis. In this section, we have simulated the model for the number of times to get the result. Finally, Sect. 5 describes conclusions and outlines some of our future directions.

2 Related Work

In this section, we discuss two streams of relevant literature at the intersection of which our work lies. The first work is related to SIR and SIS model and their modifications by introducing the delay in time to become infected. The second work related to epidemic spreading which have analyzed using different types of networks for proposed SIR and SIS model.

2.1 Mathematical Modelling of Epidemic Spreading

In 1760, Daniel Bernoulli [12] proposed the first mathematical approach for epidemics study for the spread of infectious diseases. In this course of action, the classical SIR model is proposed by Kermack and McKendrick [6] where rate equation for all three compartment given by,

$$\frac{dS(t)}{dt} = -\beta S(t)I(t) \quad (1)$$

$$\frac{dI(t)}{dt} = \beta S(t)I(t) - \mu I(t) \quad (2)$$

$$\frac{dR(t)}{dt} = \mu I(t). \quad (3)$$

where, $S(t), I(t), R(t)$ is the fraction of susceptible, infected and recovered population at time t , while, β and μ are spreading rate and recovery rate respectively. However, the classical SIR epidemic model proposed by MK does not consider the heterogeneity and topology of the complex network. Later, Anderson and

May proposed various models to describe the disease-related deaths and disease-reduced reproduction that had the great impact on the population size [2]. There are numerous works which explored the SIS and SIR models for understanding the epidemic spreading. Singh et al. proposed the modified SIR model by considering the standard SIR rumor spreading model with degree dependent tie strength of nodes and nonlinear spread of rumor by introducing two parameters named as nonlinear exponent and degree dependent tie strength exponent [13]. Shi et al. [14] studied the behaviour of the SIS epidemic model by including the propagation vector and observe that the propagation vector reduced the epidemic threshold and lead the disease spreading.

In addition, various modified SIR models have been proposed by considering delay as well. In delay models of SIR [15], a delay parameter (τ) is introduced during infection, which subsequently modifies the above Eqs. (1)–(3) to follow, $\frac{dS(t)}{dt} = -\beta S(t)I(t-\tau)$, $\frac{dI(t)}{dt} = \beta S(t)I(t-\tau) - \mu I(t)$, and $\frac{dR(t)}{dt} = \mu I(t)$. Some of the researches introduce delays in the epidemic spreading models, such as [16, 17]. Both Takeuchi [8] and Wang [5] proposed the mathematical model of delayed SIR without considering the underlying network topology. Xia et al. [18] proposed a delayed SIR model based on the SIR model and combined the delayed SIR with propagation vector to investigate the impact of infection delay in recovery and propagation vector on the spreading behaviours in complex networks, however, they did not explain the impact of infection delay on susceptible.

In delay models of SIR [17], demography is considered with introduction of a carrying capacity of infection (K) with delay in infection, which subsequently modifies the above Eqs. (1)–(3) to follow,

$$\frac{dS(t)}{dt} = rS(t)\left(1 - \frac{S(t)}{K}\right) - \beta S(t)I(t-\tau) \quad (4)$$

$$\frac{dI(t)}{dt} = \beta S(t)I(t-\tau) - (\gamma_1 + \mu)I(t) \quad (5)$$

$$\frac{dR(t)}{dt} = \mu I(t) - \gamma_2 R(t). \quad (6)$$

where the parameters r is the birth rate, γ_2 and γ_2 are the death rate of infective and recovered population. $S(t)$, $I(t)$, $R(t)$, β , μ have the same meaning as in Eqs. (1)–(3).

2.2 Epidemic on Network

Due to stochastic nature of epidemic spreading as it changes with time, different underlying network structures are used by various researchers to show the different spreading patterns. The advancement in the area of complex networks sets the base for the epidemic dynamics and initiated several related studies [19]. For example, a lot of emergent events in social networks and biological networks are pretended using the concept of complex networks [20]. Therefore, disease spreading patterns in the human population can be seen and analyzed by using the different topological structure [4].

Vespignani et al. [3] proposed the epidemic spreading model on the scale-free network to analyze the absence of epidemic threshold and its associated critical behaviour. Their proposal was based on computer virus spreading on communication and social networks. Moreno et al. [9] presented a new epidemiological framework characterized by a highly heterogeneous response of the system to the introduction of infected individuals with different connectivity considering underlying scale-free network. Li et al. [21] proposed the general spreading dynamical behaviours in small-world evolving networks when control strategies are applied to suppress the propagation of diseases, viruses, and disasters.

But realistic model should include some time delay as delay to become infected, plays an important role in the dynamics of the epidemic. For instance, it can be the incubation period of the infectious disease [22], the infectious period of patients [5], and the immunity period of recovery of the disease with time delay [18]. However, very less attention has been given to the epidemic models with time delays on heterogeneous networks as most of the dynamical process on networks is done without considering the delay in the process.

3 Proposed Methodology

In this section, we explain the *SIR epidemic model with delay*. Let, $G(N, E)$ defines the network of N nodes that represent the total population and E denotes the connections between nodes representing the interaction between individuals through which epidemic spreads. The propagation of disease is explained as: Each healthy node takes a time delay of τ to get infected. Thus, a node which gets in contact with the infected person at time $(t - \tau)$ becomes infected after the time delay of τ . The infection rate is represented by β and μ represents the recovery rate where, β and $\mu \in [0, 1]$.

Let $S_k(t)$, $I_k(t)$ and $R_k(t)$ be the fraction of the susceptible, infected and recovered nodes at time t with degree of k and $S_k(t) + I_k(t) + R_k(t) = 1$. Let $p(k)$ be the degree distribution of the network, during the epidemic process, which describes the degree of the node where, $k \in [1, N - 1]$.

The transition rules of nodes' from one state to other state is defined as:

1. A healthy node get in contact with infected node at time t , but it become infected after time delay τ .
2. A node may be recovered spontaneously at any time with rate μ . Recovery of a node doesn't require any contact. Hence, $\mu = 1$ is considered at each time stamp a node will be recovered.
3. Once a node will get recovered it will never be infected and susceptible.
4. In addition, we are not considering demography that is birth and death of nodes, therefore, the total number of nodes will be constant throughout the transition.

Now, using the mean-field rate equations are defined for dynamics of epidemic on network, based on the above transitions with time delay τ for heterogeneous networks,

$$\frac{dS_k(t)}{dt} = -\beta k S_k(t) \Omega_k(t - \tau) \quad (7)$$

$$\frac{dI_k(t)}{dt} = \beta k S_k(t) \Omega_k(t - \tau) - I_k(t) \quad (8)$$

$$\frac{dR_k(t)}{dt} = I_k(t). \quad (9)$$

Where,

$$\Omega_k(t) = \sum_{k=1}^{k_{max}} P(k' | k) I_k(t) \quad (10)$$

$$\Omega_k(t - \tau) = \sum_{k=1}^{k_{max}} P(k' | k) I_k(t - \tau) \quad (11)$$

$$\Omega_k(t - \tau) = \frac{\sum_{k=1}^{k_{max}} k' P(k') I_k(t - \tau)}{\langle k \rangle}. \quad (12)$$

The dynamics of SIR are coupled through the function $\Omega(t)$ that describes the probability that an any given link of the susceptible node connected to an infected node of degree k at time t . Here, the heterogeneous uncorrelated network [19] is considered, hence, $P(k' | k) = \frac{k' p(k')}{\langle k \rangle}$, where, $P(k' | k)$ is degree-degree correlation and $\langle k \rangle$ is mean degree of node. Generally, a healthy node is infected after certain delay, and this infected node is converted into recovered node. So we can say $S_k(t)$ is converted into $R_k(t)$. Therefore, from Eqs. 7, 9 and 11,

$$\frac{dS_k(t)}{dR_k(t)} = \frac{-\beta k S_k(t) \sum_{k=1}^{k_{max}} P(k' | k) I_k(t - \tau)}{I_k(t)}. \quad (13)$$

Where, Eq. 13 shows the rate of change of susceptible nodes to recovered nodes. After setting non-negativity and boundness of solution by using [23] as $S_k(0) > 0$, $I_k(s) = 0$ for $s \in [-\tau, 0]$ and $R_k(0) = 0$ where $0 \leq \tau \leq t$. Delay(τ) should always be less than current time of spreading, because if τ is greater than t that creates negativity. Integrating both side of Eq. 13,

$$S_k(t) = e^{-\beta k \sum_{k=1}^{k_{max}} P(k' | k) \tau R_k(t)} \quad (14)$$

$$\text{Let, } \theta_k(t) = \frac{\sum_{k=1}^{k_{max}} k' P(k') R_k(t)}{\langle k \rangle}$$

$$S_k(t) = e^{-\beta k \tau \theta_k(t)}.$$

The negative exponent in Eq. 14 shows that the number of susceptible nodes are decreasing and converted into recovered nodes. Epidemic arrives at steady state at $t \rightarrow \infty$ hence, $I_k(\infty) = 0$. Therefore, normalized condition for steady state is,

$$S_k(\infty) = e^{-\beta k \tau \theta_k(\infty)} \quad (15)$$

$$R_k(\infty) = 1 - e^{-\beta k \tau \theta_k(\infty)}. \quad (16)$$

$$\begin{aligned} \theta_k(\infty) &= \frac{\sum_{k=1}^{k_{max}} k' P(k') (1 - e^{-\beta k \tau \theta_k(\infty)})}{\langle k \rangle} \\ &= G(\theta_k(\infty)). \end{aligned}$$

where, $G(\theta_k(\infty))$ is a continuous, and increasing function for $\theta_k(\infty)$,

$$\theta_k(\infty) = \frac{\sum_{k=1}^{k_{max}} k' P(k') (1 - e^{-\beta k \sum_{k=1}^{k_{max}} P(k') |k| \tau R_k(\infty)})}{\langle k \rangle}.$$

Therefore, the solution of above equation must lies between 0 and 1 because $[0, 1]$ is the boundary solution. To get the non trivial solution, the derivative of $G(\theta_k(\infty))$ with respect to $\theta_k(\infty) = 0$ must hold following condition,

$$\left. \frac{dG(\theta_k(\infty))}{d\theta_k(\infty)} \right|_{\theta_k(\infty)=0} > 1. \quad (17)$$

After solving Eq. 17 we get, $\beta = \frac{\langle k \rangle}{\langle k^2 \rangle \tau} > 1$. Therefore, the critical spreading rate under heterogeneous network can defined as $\beta > \beta_c = \frac{\langle k \rangle}{\langle k^2 \rangle \tau}$.

If $\tau \rightarrow 0$, then, $\beta_c \rightarrow \infty$, it means that epidemic spread quickly to whole population, which is not realistic. To make value of β_c countable $\tau + 1$ is used in place of τ , hence, if $\tau > 0$, $\beta > \beta_c = \frac{\langle k \rangle}{\langle k^2 \rangle (\tau+1)}$. If there is no infection delay ($\tau = 0$) then the delayed SIR will become the standard SIR model and the critical threshold is $\beta_c = \frac{\langle k \rangle}{\langle k^2 \rangle}$ which is similar to that in [19]. Therefore, the critical threshold is inversely proportional to infection delay in susceptible, as delay increases critical threshold decreases. If the delay is too large then epidemic will die out automatically and disease will not spread out.

4 Simulation and Results

In this section, we first explain our simulation setup and further, we discuss the results of our simulations performed using *SIR with delay* model using three different types of underlying networking topologies: Random Graph, RGG with and without mobile agents. Each of the network consists of 2000 nodes that represents the human population in an area of $2500\text{m} \times 2500\text{m}$ 2-D region, where the connecting probability of a node with other nodes is considered 0.2

for random network model $G(N, p)$. For random geometric networks with or without mobile agents, the connecting radius, r of 2 m is kept for a possibility of creating a connection with other nodes in the region. The reason behind keeping this distance small is that as generally infectious diseases like chicken pox and tuberculosis spread when two persons get in contact at a short distance. For the random geometric network with mobile agents, velocity (V) of nodes has been assigned randomly from the range of [3, 100] km/h. The logic behind keeping the varying velocity is due to the fact that some individuals prefer to walk and others tend to move by vehicles. The expected length between two random points is $(0.521 * \text{length of simulation area})$ [24]. Various parameters for simulations are listed in Table 1.

We perform the different simulations to explain the proposed delayed SIR on the complex network by using different parameters (see Table 1). We focus on the effect of delay on the dynamics of epidemics on heterogeneous networks. In simulation if $\tau = 0$, the critical threshold will become $\beta_c = \frac{\langle k \rangle}{\langle k^2 \rangle}$.

The epidemic spreading with delay using random networks as underlying topology is shown in Fig. 1. We have taken the four value of τ i.e. 0, 10, 20 and 30 to analyze the effect of delay in infection. At $\tau = 0$ the epidemic spreading behaves like the standard SIR spreading model as shown in Fig. 1(a). When the value of τ increases, the spreading of infection decreases and time span of existence of epidemic becomes longer as shown in Fig. 1(b), (c) and (d). We observed that with the increase in delay, there is the decrease in the critical threshold but the timescale for the existence of disease increases.

Table 1. Simulation parameter

Name of parameter	Value
Nodes	2000
Simulation area (Square Region)	2500 m × 2500 m
Length of simulation area (a)	2500 m
Connectivity radius (r) for (RGG)	2 m
Spreading rate (β)	0.6
Recovery rate (μ)	0.1
Delay (τ)	[0, 10, 20,100]
Connectivity probability for (E-R model)	0.2
Expected length between two random point	$0.521 * a$
Velocity (V) for (Mobile RGG)	Random (3, 100)

The epidemic dynamics with different values of τ starting with $\tau = 0$ to 30, with an interval of 10 in case of the random geometric network is shown in Fig. 2. Apart from the decrease in the critical threshold, it is observed that epidemic spreading in case of random geometric networks does not differ too much from

random networks as the average degree of the random network as well as in case of the random geometric network is almost constant.

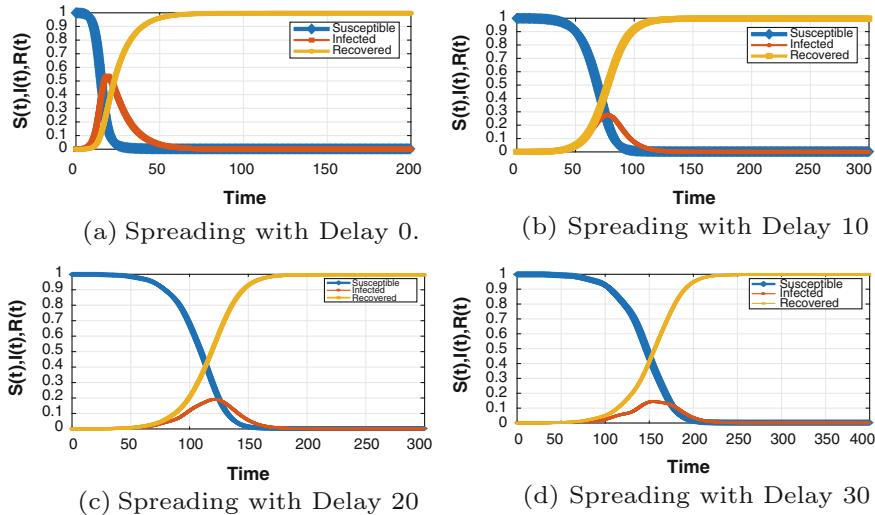


Fig. 1. Effect of delay in SIR considering underlying random network (E-R)

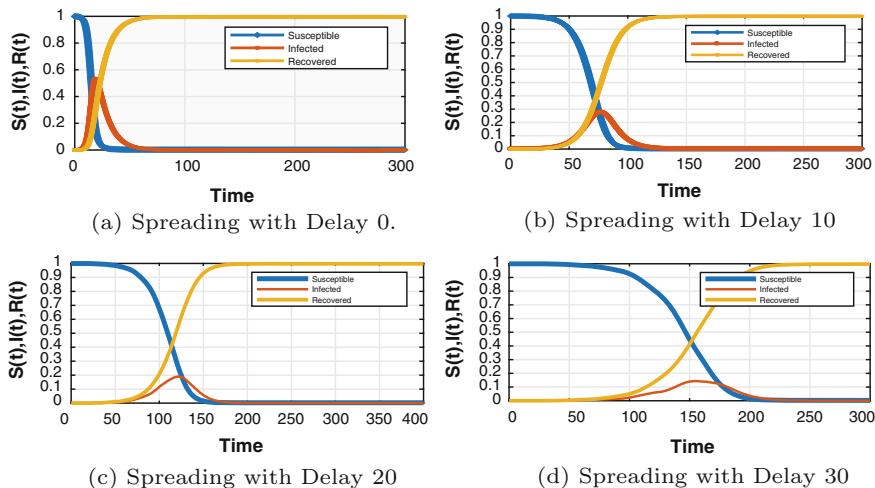


Fig. 2. Effect of delay in SIR considering underlying random geometric network

The results of the epidemic spreading with delay by considering random geometric networks with mobile agents is shown in Fig. 3. The epidemic spreading

without delay is shown in Fig. 3(a) that is standard SIR. When we increase the value of delay τ as 10, 20 and 30, it is found that the rate of infection spreading decreases. But the rate of epidemic spreading remains much greater than the rate of epidemic spreading on the random network and random geometric network (Fig. 3(b), (c) and (d)). We observed that in case of random geometric networks with mobile agents, spreading pattern is different from other two networks i.e. random network and random geometric network. In particular, the epidemic spreading varies on the node's location because the average degree of a node is stochastic in random geometric networks with mobile agents. That is the number of neighbours it comes in contact at any particular location.

It may also be seen that delay in infection decreases the critical threshold of spreading rate as shown in Fig. 4. This shows that critical threshold in SIR model is higher on a random network as compared to the random geometric network with mobility and random geometric network. The critical spreading rate for having an outbreak in the random graph is high, which increases with the value of delay. Therefore, if the delay will be increased in the random graph then for an outbreak to happen in the network, a higher spreading rate is required. It may be concluded that higher delay may be used to stop the epidemic outbreak against the given spreading rate.

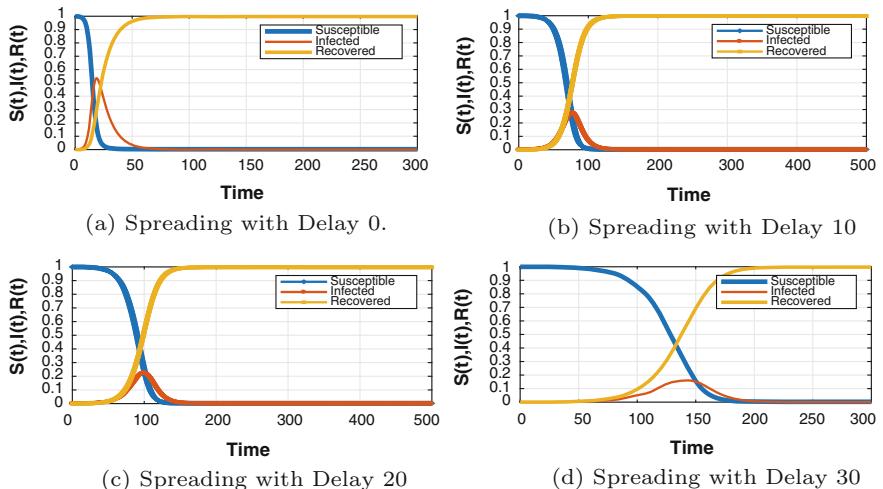


Fig. 3. Effect of delay in SIR considering underlying random geometric network with mobile agents

Simulation of delayed SIR model proposed by Xia et al. [18] without propagation vector by considering the random network as underlying topology. The simulation shows that the delay time (τ) in recovery increases the number of the infected person in the population that accelerates the spread of epidemic quickly. To make the comparison between our proposed work and Xia's model of

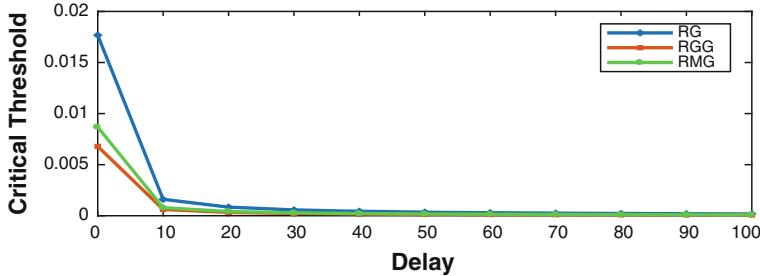


Fig. 4. Effect of delay on critical threshold on RG, RGG, RMG

delayed SIR we take same value for the delay as 10 and 20. For the delay $\tau = 10$, infection increases but timescale of existence of epidemic is smaller (Fig. 5(a)) as compared to random network (Fig. 1(b)). When the delay in recovery increased to 20 then infected population increases rapidly (Fig. 5(b)). Therefore, larger delay in recovery makes infection spreading continue to the whole population, while the larger delay in infection makes epidemic die out automatically.

5 Conclusion and Future Work

Classical epidemiological models unable to describe the spreading pattern of infectious diseases and the effect of delay in spreading. Underlying network shows the contact pattern between the human population. Delay in infection plays an important role in epidemic spreading. In our model, we have considered delay when an agent changes its state from susceptible to infectious while Xia's model considered the delay in the recovered state from infected state. We have obtained the spreading threshold that is inversely proportional to delay (τ). We have simulated the delayed SIR model considering 3 different underlying networks as Random Network, Random Geometric Network with and without mobile agents. Simulations also show that delay decreases the critical threshold value of spreading rate. By considering a different delay in the conversion from susceptible to infected nodes for different instances we have found that diseases persist for

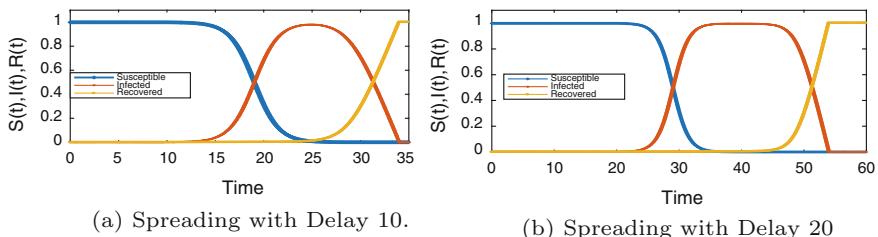


Fig. 5. Epidemic spreading in random graph by considering delay in recovery

the longer time in the human population as delay increases the duration of existence of diseases. The simulation also shows that if the delay is much larger then epidemic die out automatically.

We plan to include various future directions for this work. We plan to use additional dynamic networks for our future study. Another direction could be to use larger and real networks for understanding the epidemic behaviour. Important, we plan to include infection delay and recovery delay simultaneously in our model, in our future studies.

Acknowledgements. This work is supported by H2020 framework project, SoBig-Data, grant number 654024.

References

1. Sharma, R., Datta, A.: GoDisco++: a gossip algorithm for information dissemination in multi-dimensional community networks. *Pervasive Mob. Comput.* **9**(2), 324–335 (2012)
2. Anderson, R.M., May, R.M., Anderson, B.: *Infectious Diseases of Humans: Dynamics and Control*, vol. 28. Wiley Online Library, New York (1992)
3. Vespignani, A.: Modelling dynamical processes in complex socio-technical systems. *Nat. Phys.* **8**(1), 32 (2012)
4. Pastor-Satorras, R., Castellano, C., Van Mieghem, P., Vespignani, A.: Epidemic processes in complex networks. *Rev. Mod. Phys.* **87**(3), 925 (2015)
5. Wang, J.-J., Zhang, J.-Z., Jin, Z.: Analysis of an sir model with bilinear incidence rate. *Nonlinear Anal. R. World Appl.* **11**(4), 2390–2402 (2010)
6. Kermack, W.O., McKendrick, A.G.: A contribution to the mathematical theory of epidemics. In: Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences, vol. 115, pp. 700–721. The Royal Society, London (1927)
7. Hethcote, H.W.: Qualitative analyses of communicable disease models. *Math. Biosci.* **28**(3–4), 335–356 (1976)
8. Takeuchi, Y., Ma, W., Beretta, E.: Global asymptotic properties of a delay sir epidemic model with finite incubation times. *Nonlinear Anal. Theory Methods Appl.* **42**(6), 931–947 (2000)
9. Moreno, Y., Pastor-Satorras, R., Vespignani, A.: Epidemic outbreaks in complex heterogeneous networks. *Eur. Phys. J. B-Condens. Matter Complex Syst.* **26**(4), 521–529 (2002)
10. Erdos, P., Rényi, A.: On the evolution of random graphs. *Publ. Math. Inst. Hung. Acad. Sci.* **5**(1), 17–60 (1960)
11. Penrose, M.: *Random Geometric Graphs*, vol. 5. Oxford University Press, Oxford (2003)
12. Bernoulli, D.: Essai dune nouvelle analyse de la mortalité causée par la petite vérole et des avantages de la inoculation pour la prévenir. *Histoire de l'Acad. Roy. Sci. (Paris) avec Mém. des Math. et Phys. and Mém.* **1**, 1–45 (1760)
13. Singh, A., Singh, Y.N.: Nonlinear spread of rumor and inoculation strategies in the nodes with degree dependent tie strength in complex networks. arXiv preprint [arXiv:1208.6063](https://arxiv.org/abs/1208.6063) (2012)
14. Shi, H., Duan, Z., Chen, G.: An sis model with infective medium on complex networks. *Phys. A Stat. Mech. Appl.* **387**(8–9), 2133–2144 (2008)

15. Beretta, E., Takeuchi, Y.: Global stability of an sir epidemic model with time delays. *J. Math. Biol.* **33**(3), 250–260 (1995)
16. Zhang, J.-Z., Wang, J.-J., Su, T.-X., Jin, Z.: Analysis of a delayed sir epidemic model. In: 2010 International Conference on Computational Aspects of Social Networks (CASoN), pp. 192–195. IEEE (2010)
17. Liu, L.: A delayed sir model with general nonlinear incidence rate. *Adv. Differ. Equ.* **2015**(1), 329 (2015)
18. Xia, C., Wang, L., Sun, S., Wang, J.: An sir model with infection delay and propagation vector in complex networks. *Nonlinear Dyn.* **69**(3), 927–934 (2012)
19. Nekovee, M., Moreno, Y., Bianconi, G., Marsili, M.: Theory of rumour spreading in complex social networks. *Phys. A Stat. Mech. Appl.* **374**(1), 457–470 (2007)
20. Albert, R., Barabási, A.-L.: Statistical mechanics of complex networks. *Rev. Mod. Phys.* **74**(1), 47 (2002)
21. Li, X., Wang, X.: Controlling the spreading in small-world evolving networks: stability, oscillation, and topology. *IEEE Trans. Autom. Control* **51**(3), 534–540 (2006)
22. Wang, W., Zhao, X.-Q.: An epidemic model in a patchy environment. *Math. Biosci.* **190**(1), 97–112 (2004)
23. Nakata, Y.: A periodic solution of period two of a delay differential equation. arXiv preprint [arXiv:1801.09244](https://arxiv.org/abs/1801.09244) (2018)
24. Bettstetter, C., Resta, G., Santi, P.: The node distribution of the random waypoint mobility model for wireless ad hoc networks. *IEEE Trans. Mob. Comput.* **2**(3), 257–269 (2003)

Dynamics On/Of Networks



A Markov Model for Inferring Flows in Directed Contact Networks

Steve Huntsman^(✉)

BAE Systems FAST Labs, 4301 North Fairfax Drive, Arlington, VA 22203, USA
steve.huntsman@baesystems.com

Abstract. *Directed contact networks* (DCNs) are a particularly flexible and convenient class of temporal networks, useful for modeling and analyzing the transfer of discrete quantities in communications, transportation, etc. Transfers modeled by contacts typically underlie *flows* that associate multiple contacts based on their spatiotemporal relationships. To infer these flows, we introduce a simple inhomogeneous Markov model associated to a DCN and show how it can be effectively used for data reduction and anomaly detection through an example of kernel-level information transfers within a computer.

Keywords: Temporal networks · Markov model

1 Introduction

To identify and prevent data exfiltration by advanced persistent threats (APTs), it is useful to analyze graphs that encode causality data about system activities. The form of this causality data varies widely among providers, but it can be caricatured as tagged and tracked information (micro-)transfers. In practice, these are realized as annotations of system calls or similar kernel-level events [2, 6]. We have found the following intermediate representation or *summary* useful:

$$(timestamp, subject, verb, object).$$

In our tools, this takes the explicit form

$$(timestamp, (processname, processID), eventtype, filename) \quad (1)$$

where the “everything is a file” philosophy applies to the last entry (e.g., for a fork event, the filename is the forked process ID).

Meanwhile, any causal representation of the transfer of some token must involve a *source* s , a *target* t , and some notion of the *time* τ at or over which the transfer occurs. The simplest notion of such a time is a single instant, and the resulting notion of a (*directed*) *contact* as an ordered triple (s, t, τ) is not only the simplest, but also a very general causal representation of a transfer. For example, a transfer from a source s to a target t over the time interval $[\tau_0, \tau_1]$

could be represented with the two contacts $(s, *, \tau_0)$ and $(*, t, \tau_1)$, where here $*$ is a shorthand for the triple $(s, t, [\tau_0, \tau_1])$.¹

Many kernel-level events have an unambiguous directionality with respect to potential information transfer. For example, if process A closes file X , this might entail deleting X , which can be regarded as a degenerate information transfer from A to X (i.e., overwriting X and its metadata with \emptyset), but there is no possibility of information transfer from X to A as a result of the closure *per se*. Similarly, if A forks process B , information might be transferred from A to B as part of the fork, but not the other way around. In both of these examples the source A can reasonably be interpreted as “writing” to a target in some sense. Events which can be viewed as generalized reads or writes in this way naturally correspond to directed contacts. Other events do not have an unambiguous directionality, and as such conservatively correspond to pairs of directed contacts with the source and target swapped (e.g., the act of opening a file can entail a [generalized] read or write).

This paper focuses on how replacing each event of the form (1) with either one or two contacts provides a further useful abstraction of causality/transfer data that is mathematically convenient. In particular, we show how a natural model arises for probabilistically modeling potential flows. This model involves just one parameter, and there is a simple heuristic for setting it that we follow in practice. We detail the model behavior through analytical and practical examples. It is important to note at the outset that the model is not statistical in the sense that it involves no learning, fitting, optimization, etc. Its construction instead follows the tradition of physics by starting from various required symmetries (e.g., time translation invariance) that *any* model built from contacts ought to obey and deriving the most general mathematical structure that is consistent with those symmetries. As a byproduct of this generality, the model also applies to related problems such as traffic analysis.

The difference between (1) and a directed contact is manifested in two graphs that we have used to analyze system behavior. The *enterprise provenance graph* (EPG) is morally an undirected multigraph, with vertices labeled by files (including processes) and edges labeled by event types and timestamps.² The *temporal digraph* (TD) is directed, with vertices labeled by ordered pairs of files and timestamps, *spatial arcs* corresponding to contacts as outlined above, and *temporal arcs* linking files through time. While the EPG has demonstrated its utility in several forensic challenges under the DARPA Transparent Computing program and supports time-aware backtracking [7], it represents the passage of both time and (potentially) information implicitly through annotations, sacrificing explicit

¹ A useful analogy is of a flight departing from s at τ_0 and arriving at t at τ_1 : the contacts $(s, *, \tau_0)$ and $(*, t, \tau_1)$ respectively correspond to embarking and debarking. This analogy also highlights that alternative representations could also include additional contacts $(s, *, \tau_*)$ with $\tau_0 \leq \tau_* < \tau_1$ depending on the desired behavior.

² In fact the EPG is a directed graph with vertices bipartitioned into files and events; arcs indicating a subject or object go from events to files. However, there is an obvious bijective correspondence between this and our moral characterization.

structure for precision and expressiveness. The TD places this same structure above other considerations (though its arcs could also easily be annotated with summarized events). This has several distinct benefits: for example, it turns the backtracking problem into a trivial breadth-first search, and also naturally leads to the model at the heart of the present paper.

Notwithstanding the context of information transfer, the closest work to ours is [15], which demonstrates that the most probable paths in a Markovian model of a very complicated temporal network (viz., ocean water transport in the Mediterranean) suffice to describe the network's key features. The particulars of our model and context are very different apart from the gross feature of Markovity, but our conclusion is essentially identical.

The paper is structured as follows: Sect. 2 introduces directed contact networks and temporal digraphs; Sect. 3 discusses the Markov model at the heart of this paper; Sect. 4 discusses its performance in data reduction and anomaly detection, and Sect. 5 concludes the paper.

2 Directed Contact Networks and Temporal Digraphs

Digraphs admit a natural temporal generalization called *directed contact networks (DCNs)* that are a particularly simple incarnation of *temporal networks* [4,9]. While we can think informally of DCNs as collections of contacts as described in Sect. 1, this allows certain degenerate situations to occur that a slightly more formal and restrictive notion will avoid. Towards this end, a DCN with vertex set $V \equiv [n] \equiv \{1, \dots, n\}$ is a finite nonempty set \mathcal{C} for which each *contact* $c \in \mathcal{C}$ corresponds to a unique triple $(s(c), t(c), \tau(c)) \in [n] \times [n] \times \mathbb{R}$ with $s(c) \neq t(c)$; when convenient we identify contacts and their corresponding triples. There is an obvious notion of a temporally coherent path which we do not bother to write out formally but which is indicated in Fig. 1.

Define the *temporal digraph* of \mathcal{C} (see Fig. 1 for an example) to be the digraph $T(\mathcal{C})$ with vertex and arc sets

$$\begin{aligned} V(T(\mathcal{C})) &:= \{(v, \pm\infty) : v \in V\} \\ &\cup \{(v, \tau(c)) : [(v, c) \in V \times \mathcal{C}] \wedge [s(c) = v \vee t(c) = v]\} \end{aligned} \quad (2)$$

$$\begin{aligned} A(T(\mathcal{C})) &:= \{((s(c), \tau(c)), (t(c), \tau(c))) : c \in \mathcal{C}\} \\ &\cup \{((v, \tau_{j-1}^{\oplus v}), (v, \tau_j^{\oplus v})) : v \in V, j \in [|C @ v| - 1]\} \end{aligned} \quad (3)$$

where the *temporal fiber at v* is

$$\mathcal{C} @ v := \{\pm\infty\} \cup \{\tau(c) : c \in \mathcal{C} \wedge (s(c) = v \vee t(c) = v)\} \equiv \{\tau_j^{\oplus v}\}_{j=0}^{|\mathcal{C} @ v| - 1}. \quad (4)$$

The first set in the union on the RHS of (3) is the set of *temporal arcs*; the second set is the set of *spatial arcs*. Note that $|V(T(\mathcal{C}))| = \sum_v |\mathcal{C} @ v| \leq 2|V| + 2|\mathcal{C}|$ and $|A(T(\mathcal{C}))| = |V(T(\mathcal{C}))| - |V| + |\mathcal{C}| \leq |V| + 3|\mathcal{C}|$, so that $T(\mathcal{C})$ can be formed with only linear overhead (though this requires some care in practice).

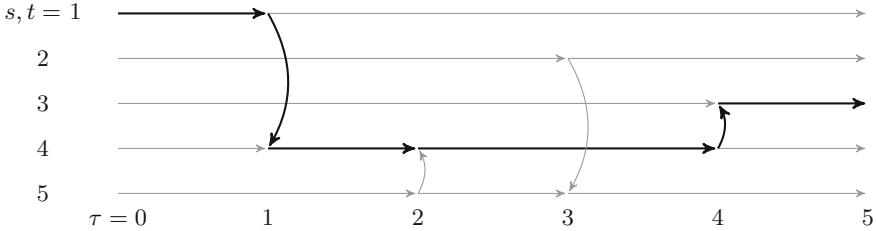


Fig. 1. Temporal digraph of the DCN $\mathcal{C} := \{(1, 4, \tau_1), (5, 4, \tau_2), (2, 5, \tau_3), (4, 3, \tau_4)\}$ with $\tau_1 < \tau_2 < \tau_3 < \tau_4$. Note that there is a temporally coherent path from \mathcal{C} -vertices 1 to 3 (indicated with bold black versus gray arrows), but not from 2 to 3.

3 Markov Model Associated to a DCN

The reader may initially be surprised that a useful probabilistic model of potential information flow can be built upon $T(\mathcal{C})$ alone. However, we note that the claim of utility merely asserts that ubiquitous traffic-analytic products such as pen registers or trap and trace devices (defined in 18 USC §3127) actually produce data that allow a user to make substantive inferences about the sources and diffusion of information in a communications network. In this light, the claim is obviously supported by general knowledge that in turn informs the basic assumption of the model.

Namely, suppose we have contacts of the form $(A, B, 0)$ and (B, C, τ) : the model probability that information (potentially) flows from A to C ought to decrease from unity to zero as $\tau \uparrow \infty$. It is important to note that this behavior does not entail that the model will have problems with capturing APTs: even a “low and slow” data exfiltration is almost certain to involve at least some system call-scale directed contacts that are either unusual in their own right or are temporally localized.

Besides the qualitative requirement above, a reasonable model that assigns probabilities to the arcs of $T(\mathcal{C})$ is tightly constrained by a number of basic symmetries that it must obey, namely w.r.t.

- (i) vertices (probabilities assigned to spatial arcs must not explicitly depend on their source or target);
- (ii) time windowing (the model must yield probabilities for information flows from a source at an initial time to a target at a terminal time that coherently compose over different sets of adjacent time windows spanning the same interval);
- (iii) non-local behavior (probabilities assigned to temporal arcs must only depend on their duration and the number of spatial arcs sharing the same source);
- (iv) simultaneous vs. infinitesimally separated events (the probabilities for these cases must only differ infinitesimally).

For the sake of brevity, we will simply construct a model that uniquely satisfies these properties, without formally interpreting them or giving proofs. However, the nature and proof of these properties should be reasonably evident to the mathematically inclined reader from the construction itself.

Define the *restriction* of a DCN \mathcal{C} to $X \subset \mathbb{R}$ by $\mathcal{C}|_X := \tau^{-1}(\tau(\mathcal{C}) \cap X)$, i.e., the subset of contacts with times in X . Next, for $a_1 \notin \tau(\mathcal{C})^3$ and $a_0 < a_1$, consider the *restricted temporal digraph* $T(\mathcal{C})|_{[a_0, a_1]}$ obtained by modifying $T(\mathcal{C}|_{[a_0, a_1]})$ by replacing instances of $-\infty$ and ∞ in (2) with a_0 and a_1 , respectively, keeping the form of (3) apart from analogous replacements in (4). That is, $T(\mathcal{C})|_{[a_0, a_1]}$ is obtained from $T(\mathcal{C}|_{[a_0, a_1]})$ by replacing the second component of the vertices $(v, -\infty)$ with a_0 and the second component of the vertices (v, ∞) with a_1 , while retaining all the arcs.

We now introduce a physically inspired model of temporally coherent random paths in which an “inverse temperature” $\beta \in \mathbb{R}$ governs a balance between temporal and spatial arcs in a temporal digraph.⁴ Specifically, for $\varepsilon \geq 0$, $a_0 < \dots < a_M$ with $a := \{a_m\}_{m=0}^M$, $a \cap \tau(\mathcal{C}) = \emptyset$, and $m \in [M]$, we form the Markov chain on $V(T(\mathcal{C})|_{[a_{m-1}, a_m]})$ with transition matrix $P_{(\mathcal{C}, a, m)}^{(\beta, \varepsilon)}$ defined by

$$Z_{(v, \tau_j^{\oplus v})} \cdot P_{(\mathcal{C}, a, m)}^{(\beta, \varepsilon)}((v, \tau_j^{\oplus v}), (w, \tau_k^{\oplus w})) := \begin{cases} 1 & \text{if } [v \neq w] \wedge [\tau_j^{\oplus v} = \tau_k^{\oplus w}] \\ \max(\varepsilon, \exp(-\beta[\tau_{j+1}^{\oplus v} - \tau_j^{\oplus v}])) & \text{if } [v = w] \wedge [j+1 = k] \wedge [d_{(v, \tau_{j+1}^{\oplus v})}^+ > 0] \\ \max(\varepsilon, \exp(-\beta[\tau_{(a, m)}^{\oplus v} - \tau_j^{\oplus v}])) & \text{if } [v = w] \wedge [j+1 = k] \wedge [d_{(v, \tau_{j+1}^{\oplus v})}^+ = 0] \\ 1 & \text{if } [v = w] \wedge [j = k] \wedge [d_{(v, \tau_j^{\oplus v})}^+ = 0] \\ 0 & \text{otherwise.} \end{cases} \quad (5)$$

Here $\tau_{(a, m)}^{\oplus v+} := \min(\inf[(a_m, \infty) \cap \mathcal{C} @ v], a_M)$, the $Z_{(v, \tau_j^{\oplus v})}$ are defined so that the rows of $P_{(\mathcal{C}, a, m)}^{(\beta, \varepsilon)}$ sum to unity, and d^+ denotes outdegree in $T(\mathcal{C})|_{[a_{m-1}, a_m]}$.

The details of (5) might appear very arbitrary, but upon examination it can be seen that quite the opposite is true. The provision for $\varepsilon > 0$ makes it possible to avoid degeneracies *in silico*⁵, whereas the absolute requirement $a \cap \tau(\mathcal{C}) = \emptyset$ is imposed to ensure that the Markov chain defined by (5) has exactly n absorbing

³ If $a_1 \in \tau(\mathcal{C})$, we can simply consider instead $a'_1 = a_1 + \varepsilon_C$, where $\varepsilon_C := \frac{1}{2} \min_{t, t' \in \tau(\mathcal{C}), t \neq t'} |t - t'|$. Note that here we assume without loss of generality that $|\tau(\mathcal{C})| > 1$, i.e., that \mathcal{C} is nontrivial as a DCN (versus, e.g., a digraph).

⁴ Bearing the concept of negative absolute temperature [12] in mind, we note that $\beta = -\infty$ corresponds to “absolute hot”, and $\beta = \infty$ corresponds to absolute zero. We follow a natural convention (and it is nothing more) for our model, in which lower temperatures correspond to slower dynamics: thus $\beta \uparrow \infty$ and $\beta \downarrow -\infty$ are respectively the limits in which no temporal and spatial arcs are traversed. **In practice, we follow a physical analogy and set β^{-1} to the average time between contacts.**

⁵ If there are (say) contacts of the form (v, w, τ_*) and (w, v, τ_*) with $\tau_* = \tau_{|\mathcal{C} @ v|-2}^{\oplus v} = \tau_{|\mathcal{C} @ w|-2}^{\oplus w}$, then (5) entails that the probability of a transition from (v, τ_*) to (v, a_m)

states, all of the form (v, a_m) . Each of these has the corresponding “emitting” state (v, a_{m-1}) , and so it is natural to consider the probability $\mathcal{P}_{(\mathcal{C}, a, m)}^{(\beta, \varepsilon)}(v, w)$ of absorption in (w, a_m) when starting from (v, a_{m-1}) . This quantity can be easily and efficiently computed using the so-called *fundamental matrix* [1]. Meanwhile, the term $\tau_{(a, m)}^{\oplus v+}$ provides a mechanism to mitigate artificial “boundary effect” behavior for $m = M$. Furthermore, (5) leads to a very clean temporal coherence property and a very straightforward physical interpretation.

Regarding the symmetry properties enumerated above, the unit and zero terms in (5) merely express property (i), viz. an equivalence among different spatial transitions and the underlying topology of the temporal digraph. Property (ii) is embodied in the following

Proposition 1. *If $a_0, a_{|a|-1} \in a' \subseteq a$, then*

$$\mathcal{P}_{(\mathcal{C}, a', 1)}^{(\beta, \varepsilon)} \cdots \mathcal{P}_{(\mathcal{C}, a', |a'|-1)}^{(\beta, \varepsilon)} = \mathcal{P}_{(\mathcal{C}, a, 1)}^{(\beta, \varepsilon)} \cdots \mathcal{P}_{(\mathcal{C}, a, |a|-1)}^{(\beta, \varepsilon)}. \quad (6)$$

□

The import of the proposition is that for any (β, ε) there is a temporally coherent family of time-inhomogeneous Markov chains associated to \mathcal{C} . The temporal coherence manifests in two ways: first, the transition probabilities for any given time interval correspond to temporally coherent random paths over that interval; second, transition matrices in successive intervals can be coherently multiplied.

Note that the proposition does not depend on the specific form of (5): indeed, the “ $\exp(-\beta \cdot \Delta\tau)$ ” terms could be replaced with fairly generic alternatives while still satisfying property (ii).⁶ However, these particular terms are required in order to jointly satisfy properties (iii) and (iv): i.e., memorylessness and self-consistency in the limit $\Delta\tau \downarrow 0$ (the latter of these entails that we cannot multiply the exponentials by some non-unit constant). That is, apart from the numerical underflow-avoiding ε (which is itself introduced in an obvious way), the form of (5) is completely dictated by the temporal digraph structure in concert with obviously desirable symmetries.⁷

The limit $\beta \rightarrow \infty$ evidently amounts to considering so-called *greedy walks* [14] (more generally, in the regime $\beta > 0$, “spatial” transitions are more likely than “temporal” transitions), and as we shall see the general construction embeds the temporal coherence of random paths in a more faithful and conservative way than

or from (w, a_*) to (w, a_m) would be exponentially small were it not for the ε term. While in principle this is not an issue, in numerical practice this leads to floating-point underflow. Taking $\varepsilon > 0$ avoids this problem without significant side effects.

⁶ The $\Delta\tau$ dependence is necessary and in the context of information flows is a plausible approximant (for small values) to the conditional Kolmogorov complexity of the intervening computation.

⁷ For a weighted DCN, normalizing so that the sum of outbound weights equals either d^+ or zero as appropriate and replacing the first case in (5) with the corresponding normalized weight gives an easy and consistent generalization.

the sorts of series of “projected snapshots” that have heretofore characterized attempts such as [3, 11, 13, 16] to map a DCN into a time series of graphs and/or provide a substrate for random walks.

The proposition above provides a mechanism for nominally optimizing the choice of a . Write $N := |\mathcal{C}|$, $M := |a|$, and suppose that $|\mathcal{C}|_{[a_m, a_{m+1}]}| \approx N/M$. Now $|V(T(\mathcal{C}|_{[a_m, a_{m+1}})))| \approx 2(n + N/M)$. Meanwhile, the complexity of computing $\mathcal{P}_{(\mathcal{C}, a, j)}^{(\beta, \varepsilon)}$ is dominated by a matrix division step of the form $(I - Q) \setminus R$, where Q is the block of $P_{(\mathcal{C}, a, j)}^{(\beta, \varepsilon)}$ whose rows and columns both correspond to transient states, and R is the block whose rows correspond to transient states but whose columns correspond to absorbing states. There are approximately $n + 2N/M$ transient states and exactly n absorbing states, so the complexity of computing $(I - Q) \setminus R$ is $O(n(n + 2N/M)^{\omega-1})$, where we take matrix multiplication and inversion to have complexity exponent $\omega > 2$ (in practice $\omega = 3$ for dense unstructured matrices). Since there are $M - 1$ multiplications, the overall complexity of the RHS of (6) is $O(Mn(n + 2N/M)^{\omega-1})$. It is easy to check that $\arg \min_M Mn(n + 2N/M)^{\omega-1} = 2(\omega - 2)N/n$, and taking this value for M yields a nominal complexity that is linear in N . In particular, the only reason to take $M \ll N/n$ is if the complexity of the other operations involved in the overall computation dominates the linear algebra complexity: it is cheaper to invert and multiply a lot of small matrices than to invert and multiply a few large matrices. Taking M to be at least comparable to N/n also has the obvious benefit of providing a more detailed picture of the dynamics of \mathcal{C} than a smaller value.

Before proceeding further, let us exhibit the basic construction:

Example 1. Consider yet again the DCN depicted in Fig. 1. Let $\tau_j = j$ for $1 \leq j \leq 4$, $\varepsilon \ll 1$, $a = \{0, 2, 5, 5\}$ and $a' = \{0, 5\}$. We then have that the entries of $P_{(\mathcal{C}, a', 1)}^{(\beta, \varepsilon)}$ are as shown in Fig. 2 and (using \cdot within matrices as a shorthand for 0)

$$\begin{aligned} \mathcal{P}_{(\mathcal{C}, a', 1)}^{(\beta, \varepsilon)} &= \begin{pmatrix} \frac{e^\beta + 1}{(e^{4\beta} + 1)(e^\beta + 1)} & \cdot & \frac{e^{5\beta}}{(e^{4\beta} + 1)(e^\beta + 1)} & \frac{e^{4\beta}}{(e^{4\beta} + 1)(e^\beta + 1)} & \cdot \\ \cdot & \frac{1}{e^{2\beta} + 1} & \cdot & \cdot & \frac{e^{2\beta}}{e^{2\beta} + 1} \\ \cdot & \cdot & 1 & \cdot & \cdot \\ \cdot & \cdot & \frac{e^\beta}{e^\beta + 1} & \frac{1}{e^\beta + 1} & \cdot \\ \cdot & \cdot & \frac{e^{2\beta}}{(e^\beta + 1)^2} & \frac{e^\beta}{(e^\beta + 1)^2} & \frac{e^\beta + 1}{(e^\beta + 1)^2} \end{pmatrix} \\ &= \begin{pmatrix} \frac{1}{e^{4\beta} + 1} & \cdot & \frac{e^{4\beta}}{e^{4\beta} + 1} & \cdot \\ \cdot & 1 & \cdot & \cdot \\ \cdot & \cdot & 1 & \cdot \\ \cdot & \cdot & \frac{1}{e^\beta + 1} & \cdot \end{pmatrix} \cdot \begin{pmatrix} 1 & \cdot & \cdot & \cdot & \cdot \\ \cdot & \frac{1}{e^{2\beta} + 1} & \cdot & \cdot & \frac{e^{2\beta}}{e^{2\beta} + 1} \\ \cdot & \cdot & 1 & \cdot & \cdot \\ \cdot & \cdot & \frac{e^\beta}{e^\beta + 1} & \frac{1}{e^\beta + 1} & \cdot \\ \cdot & \cdot & \cdot & \cdot & 1 \end{pmatrix} \\ &= \mathcal{P}_{(\mathcal{C}, a, 1)}^{(\beta, \varepsilon)} \cdot \mathcal{P}_{(\mathcal{C}, a, 2)}^{(\beta, \varepsilon)}, \end{aligned}$$

so we can see that as β increases (equivalently, the temperature decreases) the most likely transitions become those corresponding to temporally coherent paths

in which spatial arcs are greedily traversed. Meanwhile, consider the digraph D with edges $(s(c), t(c))$ for $c \in \mathcal{C}$ as well as loops (v, v) for $v \in [n]$: the adjacency matrix of D has nonzero entries in the $(2, 3)$ and $(2, 4)$ positions as well as in the sparsity pattern of $\mathcal{P}_{(\mathcal{C}, a', 1)}^{(\beta, \varepsilon)}$. These spurious $(2, 3)$ and $(2, 4)$ entries correspond to nonexistent temporally coherent paths in \mathcal{C} , highlighting that \mathcal{P} gives a more detailed and accurate picture of \mathcal{C} than D . \square

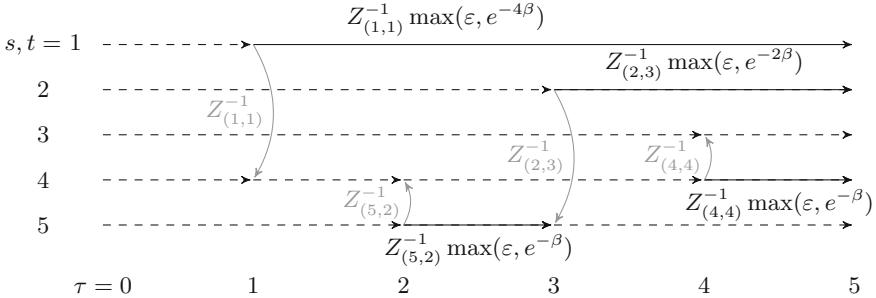


Fig. 2. Entries of $\mathcal{P}_{(\mathcal{C}, a', 1)}^{(\beta, \varepsilon)}$ not in $\{0, 1\}$ for $\mathcal{C} := \{(1, 4, 1), (5, 4, 2), (2, 5, 3), (4, 3, 4)\}$ and $a' = \{0, 5\}$ are indicated along with solid arcs (gray for spatial arcs; black for temporal arcs); unit entries correspond to dashed arcs.

3.1 Embeddability

Since a DCN lives in continuous time, it is natural to ask if the Markov chain $\mathcal{P}_{(\mathcal{C}, a, m)}^{(\beta, \varepsilon)}$ corresponds to a continuous-time Markov process, i.e., if there is a well-defined notion of instantaneous probability rate for all times. This is an instance of the so-called *Markov embeddability* problem [8], and this instance can be answered effectively for various situations of practical interest.

First suppose that no two contacts occur at once. Then the embeddability problem reduces to the case of a single contact for $n = 2$, and this in turn follows from the following readily verifiable identity for $p \in (0, 1)$:

$$\log \begin{pmatrix} 1-p & p \\ 0 & 1 \end{pmatrix} = \log(1-p) \cdot \begin{pmatrix} 1 & -1 \\ 0 & 0 \end{pmatrix}.$$

However, simultaneous contacts can obstruct embeddability. It suffices again to consider the case $n = 2$: here a stochastic matrix is embeddable iff its determinant is positive. But a quick calculation with $\mathcal{C} := \{(1, 2, 0), (2, 1, 0), (1, 2, \tau), (2, 1, \tau)\}$ and $a = \{-1, \tau/2, 2\tau\}$ shows that $\det \mathcal{P}_{(\mathcal{C}, a, 1)}^{(\beta, 0)} < 0$ for $\beta > 0$.

The following result follows immediately from Proposition IV.3 of [8] and takes both preceding cases into account. The algorithm of Sect. V.B of [8] can be used to estimate a Markov generator when it exists.

Proposition 2. *If $T(\mathcal{C})$ is acyclic, then $\mathcal{P}_{(\mathcal{C}, a, m)}^{(\beta, \varepsilon)}$ is embeddable.* \square

4 Data Reduction and Anomaly Detection

In this section, we show how the Markov model performs data reduction well enough to be used as a practical anomaly detector.

We used data from an adversarial challenge in DARPA’s Transparent Computing program. Specifically, we considered a DCN \mathcal{C} formed from $N \approx 3.4 \cdot 10^6$ read and write events spanning a period of four days and derived in turn from data produced by the CADETS tool [6]. We ignored process IDs, resulting in $n = 418$ files (of which 88 were processes). We curated the ground truth data set for this challenge to exclude events that were demonstrably not present in our input data (e.g., directory changes). This left 54 APT events, each given with an accurate timestamp with a precision of 1 s. To translate each APT event record into a form suitable for evaluation, we found the set of contacts that occur within ± 1 s (note that this is conservative), and extracted the subset of these for which either the source or target are a process or file in the APT event record. This yielded a set $\mathcal{G} \subset \mathcal{C}$ of 216 ground truth contacts distributed over the 54 APT events.

We set β to the average inter-contact time (as suggested in an earlier footnote), ε to the square root of machine epsilon, and a to 10-s windows (with $M = 28423$). Writing $\mathcal{P}(m) \equiv \mathcal{P}_{(C,a,m)}^{(\beta,\varepsilon)}$ and $(f \cup g)(X) := f(X) \cup g(X)$, we define for respective probability and relative frequency thresholds $\lambda, \mu \in (0, 1)$

$$\hat{\mathcal{I}}(m) := (\pi_1 \cup \pi_2) (\{(j, k) : [\mathcal{P}_{jk}(m) > \lambda] \wedge [|\{m' : \mathcal{P}_{jk}(m') > \lambda\}| / M < \mu]\}) \quad (7)$$

and

$$\mathcal{I}(m) := (s \cup t) (\{c \in \mathcal{G} : \tau(c) \in [a_m, a_{m+1}]\}) \quad (8)$$

That is, $\hat{\mathcal{I}}(m)$ is the set of indices corresponding to origins or destinations of information flows spanning the m th time window that are probable (as defined by λ), and for which this probability is also rare (as defined by μ); meanwhile, $\mathcal{I}(m)$ is the set of indices corresponding to sources or targets of ground truth events during the m th time window.⁸

From (7) and (8) we can define both Boolean and natural number versions of detection metrics. The Boolean version uses $\llbracket \top \rrbracket := 1$ and $\llbracket \perp \rrbracket := 0$ à la

$$\begin{aligned} \delta_{\text{bool}}^{\top+}(m) &:= \llbracket [\hat{\mathcal{I}}(m) \neq \emptyset] \wedge [\hat{\mathcal{I}}(m) \cap \mathcal{I}(m) \neq \emptyset] \rrbracket; \\ \delta_{\text{bool}}^{\perp+}(m) &:= \llbracket [\hat{\mathcal{I}}(m) \neq \emptyset] \wedge [\hat{\mathcal{I}}(m) \cap \mathcal{I}(m) = \emptyset] \rrbracket; \\ \delta_{\text{bool}}^{\perp-}(m) &:= \llbracket [\hat{\mathcal{I}}(m) = \emptyset] \wedge [\mathcal{I}(m) \neq \emptyset] \rrbracket; \\ \delta_{\text{bool}}^{\top-}(m) &:= \llbracket [\hat{\mathcal{I}}(m) = \emptyset] \wedge [\mathcal{I}(m) = \emptyset] \rrbracket. \end{aligned} \quad (9)$$

⁸ In many cases either the source or the target of a ground truth event does not exist. For example, the userspace commands `hostname` and `put/tmp/netrecon` correspond to the (*process name, filename*) pairs (`hostname, ∅`); and (`∅, /tmp/netrecon`). By way of comparison, the command `rm -f /tmp/netrecon.log` corresponds to the pair (`rm, /tmp/netrecon.log`).

The natural number analogues of (9) are (suppressing m for brevity)

$$\delta_{\text{nat}}^{\top+} := |\hat{\mathcal{I}} \cap \mathcal{I}|; \quad \delta_{\text{nat}}^{\perp+} := |\hat{\mathcal{I}} \cap \mathcal{I}^c|; \quad \delta_{\text{nat}}^{\perp-} := |\hat{\mathcal{I}}^c \cap \mathcal{I}|; \quad \delta_{\text{nat}}^{\top-} := |\hat{\mathcal{I}}^c \cap \mathcal{I}^c|. \quad (10)$$

From these we get in turn the usual detection metrics shown in Figs. 3 and 4, i.e. true positive rate (or recall) and false positive rate

$$\text{TPR} := \frac{\sum_m \delta^{\top+}(m)}{\sum_m \delta^{\top+}(m) + \sum_m \delta^{\perp-}(m)}; \quad \text{FPR} := \frac{\sum_m \delta^{\perp+}(m)}{\sum_m \delta^{\perp+}(m) + \sum_m \delta^{\top-}(m)}, \quad (11)$$

and positive predictive value (or precision) and negative predictive value

$$\text{PPV} := \frac{\sum_m \delta^{\top+}(m)}{\sum_m \delta^{\top+}(m) + \sum_m \delta^{\perp+}(m)}; \quad \text{NPV} := \frac{\sum_m \delta^{\top-}(m)}{\sum_m \delta^{\perp-}(m) + \sum_m \delta^{\top-}(m)}. \quad (12)$$

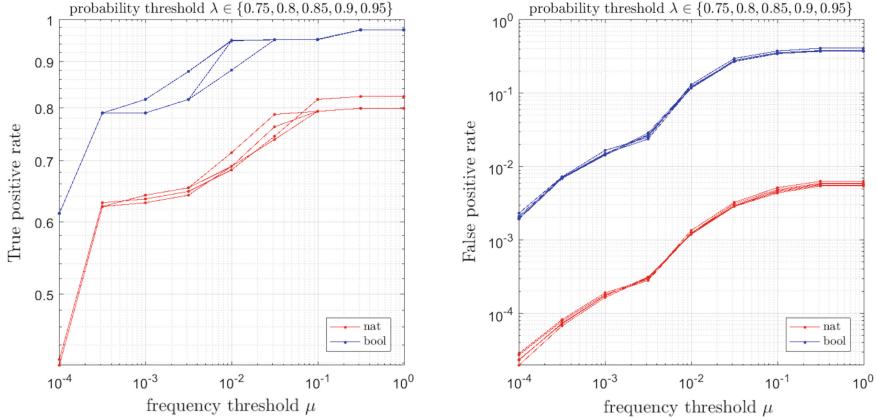


Fig. 3. True (left) and false (right) positive rates as defined in (11).

It is obvious from Figs. 3 and 4 that the results are broadly insensitive to the probability threshold λ , though it is easy to pick an optimal value using the technique of [5]. Similarly, a cursory analysis (not shown) indicates that the value of β is only important up to several orders of magnitude (this is because potential information flow probabilities strongly tend to be either very near or bounded away from unity in practice, a fact which we also exploit in setting the probability threshold λ). For the value $\mu = 10^{-3}$, we see that a clear majority of the APT events are detected (by either version of the metrics) with a false positive rate below 2% (again, by either version). The results indicate that the model is a sufficiently effective data reduction technique (in particular, the negative predictive value is essentially perfect) to be a useful anomaly detector.

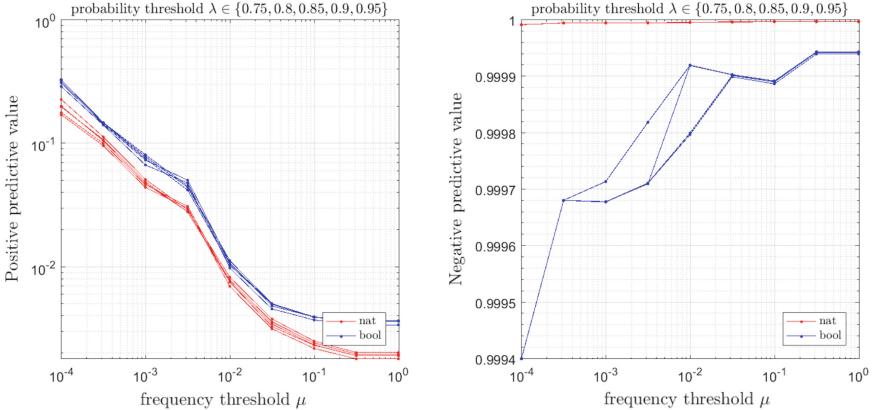


Fig. 4. Positive (left) and negative (right) predictive values as defined in (12).

In fact, of the 57 (out of 418) files which are targets of high-probability potential information flows in the model, 27 fall below the $\mu = 10^{-3}$ level and have backtracks with fewer than 20 (or for that matter, 90) vertices. From these 27, 6 (in 3 pairs) correspond to the 3 executables which the APT wrote to /tmp from its initial foothold.

5 Remarks

The Markov model depends in an essential way on the detailed structure of the underlying DCN. Experiments not detailed here have shown that inserting even 1% of random contacts seriously degrades the data reduction and anomaly detection characteristics of the model. This is a good thing, though: it indicates that the model captures the most important aspects of flows from contact data.

We can also run the model in reverse by simply applying the map $(s, t, \tau) \mapsto (t, s, -\tau)$ beforehand. However, this idea of analyzing reverse information flow remains to be fully explored, as does a related notion of a *taint calculus* proposed by George Cybenko.

An interesting perspective on the model is that it gives us a time-varying geometry. If d denotes an arbitrary metric on probability distributions, we get an induced metric for a given time window of the form $d(v, w) := d(\mathcal{P}_v, \mathcal{P}_w)$, where \mathcal{P}_v is the v th row of \mathcal{P} . This in turn allows us to do things like time-dependent hierarchical clustering. Analysis of the variation of information [10] between subsequent clusterings would give additional principled insight into dynamical behavior.

Acknowledgements. The author thanks Yingbo Song, Rob Ross, and Mike Weber for many helpful discussions as well as creating the summary and ground truth data used in Sect. 4, and George Cybenko for still more helpful discussions. This material is based upon work supported by the Defense Advanced Research Projects Agency

(DARPA) and the Air Force Research Laboratory (AFRL). Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of DARPA or AFRL.

References

1. Brémaud, P.: *Markov Chains: Gibbs Fields, Monte Carlo Simulation, and Queues*. Springer, Berlin (1999)
2. Chan, S.C., et al.: Expressiveness benchmarking for system-level provenance. In: TaPP (2017)
3. Grindrod, P., Higham, D.J.: A matrix iteration for dynamic network summaries. *SIAM Rev.* **55**, 118 (2013)
4. Holme, P.: Modern temporal network theory: a colloquium. *Eur. Phys. J. B* **88**, 234 (2015)
5. Huntsman, S.: Topological mixture estimation. In: ICML (2018)
6. Jenkinson, G., et al.: Applying provenance in APT monitoring and analysis. In: TaPP (2017)
7. King, S.T., Chen, P.M.: Backtracking intrusions. *ACM Trans. Comput. Syst.* **23**, 51 (2005)
8. Lencastre, P., et al.: From empirical data to continuous Markov processes: a systematic approach. *Phys. Rev. E* **93**, 032135 (2016)
9. Masuda, N., Lambiotte, R.: *A Guide to Temporal Networks*. World Scientific, Singapore (2016)
10. Meilă, M.: Comparing clusterings—an information based distance. *J. Multivar. Anal.* **98**, 873 (2007)
11. Perra, N., et al.: Random walks and search in time-varying networks. *Phys. Rev. Lett.* **109**, 238701 (2012)
12. Ramsey, N.F.: Thermodynamics and statistical mechanics at negative absolute temperatures. *Phys. Rev.* **103**, 20 (1956)
13. Rocha, L.E.C., Masuda, N.: Random walk centrality for temporal networks. *New J. Phys.* **16**, 063023 (2014)
14. Saramäki, J., Holme, P.: Exploring temporal networks with greedy walks. *Eur. Phys. J. B* **88**, 334 (2015)
15. Ser-Giacomi, E., et al.: Most probable paths in temporal weighted networks: an application to ocean transport. *Phys. Rev. E* **92**, 012818 (2015)
16. Starnini, M., et al.: Random walks on temporal networks. *Phys. Rev. E* **85**, 056115 (2012)

A General Model of Dynamics on Networks with Graph Automorphism Lumping

Jonathan A. Ward^(✉) and John Evans

University of Leeds, Leeds LS2 9JT, UK
j.a.ward@leeds.ac.uk

Abstract. In this paper we introduce a general Markov chain model of dynamical processes on networks. In this model, nodes in the network can adopt a finite number of states and transitions can occur that involve multiple nodes changing state at once. The rules that govern transitions only depend on measures related to the state and structure of the network and not on the particular nodes involved. We prove that symmetries of the network can be used to lump equivalent states in state-space. We illustrate how several examples of well-known dynamical processes on networks correspond to particular cases of our general model. This work connects a wide range of models specified in terms of node-based dynamical rules to their exact continuous-time Markov chain formulation.

Keywords: Dynamics on networks · Markov chains · Graph automorphisms · Lumping · Epidemic models · Opinion dynamics · Social physics

1 Introduction

Dynamical processes on networks are one of the main topics of network science [5, 51, 55]. The ubiquity of networks means that a wide range of phenomena have been modelled as dynamical processes on networks, including epidemics [41, 52], magnetism [24], opinion dynamics [23, 60, 61], diffusion of innovations [6, 46, 47, 64], rumour spread [17, 31, 38], meme popularity [29], cultural polarisation [1, 10], racial segregation [57, 58], stock market trading [40], cascading failures [27, 32, 50] and language evolution [4, 9, 14]. For example, the study of epidemics on networks includes research on the effect that network structure has on the epidemic threshold [8, 12], a variety of methods to approximate the dynamics [19, 53] and the effectiveness of vaccination strategies [54]. For thorough reviews of epidemic models on networks, see [41, 52]. As another example, the study of opinion dynamics makes use of simple models of voting behaviour based on binary [34], multi-state [62] and continuous [18, 33] opinions, where voters take into account the opinions of one [60] or more [7, 11, 23, 61] neighbours, with the possibility of zealots with fixed opinions [48]. Typically, the effect of network topology on the mean-time to reach consensus [60] and the identification of phase-transitions between ordered (consensus) and disordered (disagreement)

states [42, 49] are of interest. Of fundamental importance in the study of dynamics on networks is understanding the effect of network topology on dynamics.

General models have been proposed that attempt to capture the wide variety of dynamical processes on networks [2, 3, 21, 25, 26, 41]. From these models, one can derive high-accuracy approximations of the dynamics [21, 25, 26] and identify redundancies in state-space due to network symmetries [2, 41, 44, 59]. However, these models only allow one node in the network to change its state at any instant in time. We call such models *single vertex-state transition* models. In contrast, there are many models where multiple vertices change state at once. Examples include the naming game of language evolution [4], the Schelling model of segregation [57, 58], the Bonabeau model of hierarchy formation [9], the Twitter model of meme propagation [29], and the Sznajd [61] and majority rule [15, 23] models of opinion dynamics. We call such models *multi vertex-state transition* models. Understandably, analytical results are less common for multi vertex-state transition models.

In this paper, after presenting some background material in Sect. 2, we review five models in Sect. 3 that capture the main features of typical dynamical processes on networks. Then in Sect. 4 we introduce a general model, which accounts for multi vertex-state transitions, and in Sect. 5 we prove that the state-space of our model can be lumped using graph automorphisms. In Sect. 6 we connect our general model to those described in Sect. 3 and we conclude with a discussion of our work in Sect. 7.

2 Background

Let $\mathbb{S} = \{S_1, S_2, \dots, S_n\}$ be the *state-space* of a continuous-time Markov chain whose time-dependent probability distribution over state-space is

$$X(t) = (x_1(t), x_2(t), \dots, x_n(t))^T,$$

where $x_i(t)$ is the probability of being in state S_i at time t . The evolution of $X(t)$ is described by the *master equation*

$$\dot{X} = Q^T X,$$

also known as the forward Kolmogorov or differential Chapman-Kolmogorov equation. The infinitesimal generator Q is an n by n matrix whose ij th component describes the non-negative transition rate from the state S_i to the state S_j for $i \neq j$, and whose diagonal entries ensure the rows sum to zero (i.e. the magnitude of Q_{ii} is the transition rate out of state S_i). See, e.g. [39] for more details about Markov chains. A partition of state-space $\mathbb{L} = \{\mathbb{L}_1, \mathbb{L}_2, \dots, \mathbb{L}_r\}$ is called a *lumping* if it preserves the Markov property [37, 59]. A necessary and sufficient condition for lumping is that for all $i, j \in \{1, 2, \dots, r\}$ there exists an R_{ij} that satisfies

$$R_{ij} = \sum_{S_l \in \mathbb{L}_j} Q_{kl}, \text{ for all } S_k \text{ in } \mathbb{L}_i.$$

If $y_i = \sum_{S_k \in \mathbb{L}_i} x_k$, and $Y = (y_1, y_2, \dots, y_r)^T$, then the dynamics of the lumped probability distribution $Y(t)$ are described by

$$\dot{Y} = R^T Y .$$

A *network* or graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ consists of a set of *vertices* or nodes \mathcal{V} and a set of *edges* or links $\mathcal{E} \subset \mathcal{V} \times \mathcal{V}$. A pair of vertices $u, v \in \mathcal{V}$ are *neighbours* if $(u, v) \in \mathcal{E}$ and the *neighbourhood* of a vertex is its set of neighbours. An *automorphism* of a network \mathcal{G} is a bijection $g : \mathcal{V} \rightarrow \mathcal{V}$ such that $(u, v) \in \mathcal{E}$ if and only if $(g(u), g(v)) \in \mathcal{E}$. We use the shorthand $gv = g(v)$. The set of automorphisms of a network \mathcal{G} form a permutation group $G = \text{Aut}(\mathcal{G})$ called the automorphism group of \mathcal{G} [30].

3 Dynamical Processes on Networks

We now describe several examples of well-known dynamical processes on networks. These examples cover the main model features captured by the general Markov chain model that we introduce in Sect. 4.

The **SIS** (susceptible-infected-susceptible) model [41, 52, 53] is one of the most fundamental of epidemic models. Individuals in a finite population are considered to be in one of two possible states, either **Susceptible** to an infection, or **Infected**. Each susceptible individual is assumed to contract the infection with a rate proportional to the number of infected individuals they are in contact with. Where such contacts are described by a network, a susceptible node becomes infected at a rate $\beta m(t)$, where β is a positive rate constant and $m(t)$ is the number of infected neighbours they have at time t . In addition, each infected individual is assumed to become susceptible again at a constant rate γ .

The **SIR** (susceptible-infected-recovered) model [41, 52] is another fundamental model of epidemics that is similar in formulation to the SIS model. In the SIR model, individuals can also be in an additional **Recovered** state. As in the SIS model on a network, a susceptible individual becomes infected at a rate proportional to the number infected neighbours they have. In the SIR model however, individuals recover at a constant rate γ and then remain recovered.

The **voter** model [13, 16, 60] is a simple model of opinion dynamics that is specified in terms of a set of iterated rules. Individuals in a finite population are assumed to have one of two possible opinions, typically denoted by ± 1 . Individuals can be influenced by the opinions of others, and the structure of who influences whom is usually represented by a network. At each discrete time-step, an individual is chosen at random and they adopt the opinion of a randomly chosen neighbour. This is repeated until consensus is reached.

The **Twitter** model [29] is a model of meme propagation in on-line social networks such as Twitter. In a particular case of this model, each node in a network has a screen that is either empty or displays a meme. At each time-step, a node is selected at random and if they have a meme on their screen they ‘tweet’, i.e. the meme replaces whatever is on the screens of their neighbours. This is repeated until all screens display the meme.

The **Schelling** model illustrates the formation of segregation in communities. There are several variations of the original model [57, 58], but we describe the variant in [20], generalised to a network. Nodes in the network correspond to the locations at which people can live and there are edges between neighbouring locations. Locations can be unoccupied or occupied by two different types of people. A person of one type is unhappy if the proportion of occupied neighbouring locations of the same type is below some threshold. At each time-step, an unhappy person is selected at random and moves to an unoccupied location chosen at random. This is repeated a fixed number of times or until everyone is happy.

Of the models described above, the SIS and SIR models evolve in continuous-time, whereas the voter, Twitter and Schelling model evolve in discrete-time; the nodes in the SIS, voter and Twitter models have two possible states (i.e. binary), whereas nodes in the SIR and Schelling model have three possibilities; in the SIS, SIR and voter models, only one node changes state at any instant in time, whereas in the Twitter and Schelling model, multiple nodes can change state at once.

4 General Markov Chain Model of Dynamics on Networks

In this section we will develop a general model of Markov chain dynamics on networks that captures the main features of the models described in Sect. 3. We assume that the dynamics take place on a network $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with automorphism group G , and that at any point in time, each vertex in the network is associated with one of a finite number of *vertex-states*. We denote the set of all possible vertex-states by \mathcal{W} . Thus for the SIR model $\mathcal{W} = \{\text{S, I, R}\}$ and for the voter model $\mathcal{W} = \{+1, -1\}$. A *state* of the network \mathcal{G} is a map $S : \mathcal{V} \rightarrow \mathcal{W}$, and the *state-space* of \mathcal{G} over \mathcal{W} is the set of all possible states, denoted $\mathbb{S} = \mathcal{W}^{\mathcal{V}}$ (i.e. the set of all functions from \mathcal{V} to \mathcal{W}). Let M be the number of vertex-states and N be the number of vertices, then the number of states in state-space is M^N . Since the state-space is finite, we assume that it has been enumerated in some way, i.e. $\mathbb{S} = \{S_1, S_2, \dots, S_{M^N}\}$. For example, if $\mathcal{V} = \{1, 2, 3, 4\}$ and $\mathcal{W} = \{+1, -1\}$, then one of the $2^4 = 16$ possible states, say $S_i \in \mathbb{S}$, has $S_i(1) = S_i(3) = +1$ and $S_i(2) = S_i(4) = -1$.

Automorphisms of the network permute vertices, so we need to define how they act on state-space. Let G be the automorphism group of \mathcal{G} , then we define the *action* of $g \in G$ on a state $S_i \in \mathbb{S}$ to be

$$(gS_i)(v) = S_i(g^{-1}v) \quad \text{for all } v \in \mathcal{V}, \tag{1}$$

i.e. the vertex-state of v in gS_i is the same as the vertex-state of $g^{-1}v$ in S_i . It is easy to prove that this is indeed an action of G on \mathbb{S} in the group theoretic sense (see e.g. [22] for a precise definition of an action of a group).

We focus on dynamical processes that can be represented as Markov chains over the state-space \mathbb{S} , where the time-dependent probability distribution over

state-space is $X(t) \in \mathbb{R}^{M^N}$, as defined in Sect. 2. Here we assume that the dynamics evolve in continuous-time, but in Sect. 6 we illustrate how to choose the continuous-time transition rates such that the embedded Markov chain corresponds to a discrete-time model of interest.

In our model, transitions will only occur between pairs of states where the differing vertex-states are of particular types.

Definition 1. For $S_i, S_j \in \mathbb{S}$, the vertices $U \subset \mathcal{V}$ are called **transition vertices** if for every $u \in U$, $S_i(u) \neq S_j(u)$ and for every $v \in \mathcal{V} \setminus U$, $S_i(v) = S_j(v)$.

For notational convenience let $\mathcal{W} = \{W_1, W_2, \dots, W_M\}$, i.e. we assume that the vertex-states have been ordered in some way. For $1 \leq m \leq M$, a *sub-state* α is a vector in $\{0, 1, \dots, N\}^M$ whose m th component α_m is the number of transition vertices in the vertex-state W_m . We denote the set of *allowable sub-states* by $\mathbb{A} \subset \{0, 1, \dots, N\}^M$ and the set of *allowable sub-state transition pairs* by $\mathbb{T} \subset \mathbb{A} \times \mathbb{A}$. Of all possible sub-states, only a few need be allowable, and these must be identified by specific cases of our model. We use the set of allowable sub-state transition pairs to identify pairs of states between which transitions can occur.

Definition 2. We call $S_i, S_j \in \mathbb{S}$ a **transition pair**, denoted

$$S_i \xrightarrow{\alpha, \beta} S_j, \quad (2)$$

if $(\alpha, \beta) \in \mathbb{T}$ and for each $1 \leq m \leq M$, α_m is the number of vertices u such that $S_i(u) = W_m$ and $S_i(u) \neq S_j(u)$, and β_m is the number of vertices v such that $S_j(v) = W_m$ and $S_j(v) \neq S_i(v)$.

We assume that the transition rate between a transition pair $S_i \xrightarrow{\alpha, \beta} S_j$ depends only on α, β and a vector of metrics $\mu(t) \in \mathbb{R}^A$, for integer A . This assumption means that the transition rates do not depend on the particular set of transition vertices. We call such models *vertex homogeneous*. Furthermore, we assume that μ is invariant under network automorphisms. In more detail, if G is the automorphism group of the graph \mathcal{G} , then we assume there is a function $\eta_{\mathcal{G}} : \mathbb{S} \times \mathbb{S} \rightarrow \mathbb{R}^m$ that is a ‘structural measure’ [56], i.e. that satisfies $\eta_{\mathcal{G}}(S_i, S_j) = \eta_{\mathcal{G}}(gS_i, gS_j)$ for all $S_i, S_j \in \mathbb{S}$ and $g \in G$. As an example, if there is a single transition vertex then $\eta_{\mathcal{G}}$ might count the number of its neighbours in each vertex-state. We also assume that, for each transition pair $S_i \xrightarrow{\alpha, \beta} S_j$ with $\mu = \eta_{\mathcal{G}}(S_i, S_j)$, the transition rate is given by a known function $q_{\alpha, \beta}(\mu)$. Thus the ij th component of the infinitesimal generator is

$$Q_{ij} = \begin{cases} q_{\alpha, \beta}(\mu) & \text{if } S_i \xrightarrow{\alpha, \beta} S_j \text{ and } \mu = \eta_{\mathcal{G}}(S_i, S_j), \\ 0 & \text{otherwise.} \end{cases} \quad (3)$$

This completes the specification of our general model. To summarise, the model consists of a graph \mathcal{G} , a state-space \mathbb{S} over the graph’s vertices, a set of allowable sub-state transition pairs \mathbb{T} , a structural measure $\eta_{\mathcal{G}}$ and the transition rates $q_{\alpha, \beta}$ between all allowable sub-state transition pairs $(\alpha, \beta) \in \mathbb{T}$.

5 Graph Automorphism Lumping

In this section we prove that the automorphism group G of the network \mathcal{G} can be used to lump the state-space \mathbb{S} of the general model introduced in Sect. 4. First note that transition pairs persist under automorphisms.

Lemma 1. *If $S_i, S_j \in \mathbb{S}$ and $g \in G$, then $S_i \xrightarrow{\alpha, \beta} S_j$ if and only if $gS_i \xrightarrow{\alpha, \beta} gS_j$.*

This lemma is a direct consequence of Definition 2 and the action of the automorphism group on state-space (1), since $S_i(u) = (gS_i)(gu)$ for all $u \in \mathcal{V}$.

It follows from our definition (1) of the action of G on state-space that \mathbb{S} is a G -set [22]. This means that we can partition state-space into equivalence classes where a pair of states $S_i, S_j \in \mathbb{S}$ are equivalent if and only if there is a $g \in G$ such that $S_i = gS_j$. This partition is known as the *orbit partition* of state-space.

Theorem 1. *For the infinitesimal generator Q defined in (3), the orbit partition of state-space under the action of G is a lumping of \mathbb{S} .*

The proof of this theorem draws on that of Theorem 2.11 in [41]. Our proof is simplified by the fact that $\eta_{\mathcal{G}}$ is a structural measure, although it is generally easy to prove that particular forms of $\eta_{\mathcal{G}}$ satisfy this requirement.

Proof.. Let $\mathbb{L} = \{\mathbb{L}_1, \mathbb{L}_2, \dots, \mathbb{L}_r\}$ be the orbit partition of the state-space \mathbb{S} under the action of G . Suppose $S_k \in \mathbb{L}_i$ and $S_l \in \mathbb{L}_j$. For any $S_m \in \mathbb{L}_j$, we can find a $g \in G$ such that $S_m = gS_l$. Thus let $S_m = gS_k \in \mathbb{L}_i$. From (3) and Lemma 1, if S_k and S_l do not form a transition pair then $Q_{kl} = Q_{mn} = 0$, otherwise $S_k \xrightarrow{\alpha, \beta} S_l$ and $S_m \xrightarrow{\alpha, \beta} S_n$ for $(\alpha, \beta) \in \mathbb{T}$. By definition, $\eta_{\mathcal{G}}(S_k, S_l) = \eta_{\mathcal{G}}(S_m, S_n)$. If $\mu = \eta_{\mathcal{G}}(S_k, S_l)$, then $Q_{kl} = Q_{mn} = q_{\alpha, \beta}(\mu)$. Since \mathbb{S} is a G -set, it follows that g is a bijection on \mathbb{L}_j . Thus for all $S_k, S_m \in \mathbb{L}_i$ we have

$$\sum_{S_l \in \mathbb{L}_j} Q_{kl} = \sum_{S_n \in \mathbb{L}_j} Q_{mn},$$

and therefore \mathbb{L} is a lumping. □

6 Example Applications

In this section we show that the dynamical processes on networks introduced in Sect. 3 correspond to particular cases of our general model. We start by describing how to choose the transition rates so that the embedded Markov chain of our general model coincides with a particular discrete-time model of interest. Recall that when Q_{ij} is the transition rate from state S_i to S_j of a continuous-time Markov chain, the transition *probability* of the embedded discrete-time Markov chain is $P_{ij} = -Q_{ij}/Q_{ii}$ [39]. The only part of our general model that is specific to the continuous-time setting is the transition rate function $q_{\alpha, \beta}$. Thus to translate a *discrete-time* model of dynamics on a graph \mathcal{G} with state-space \mathbb{S} , allowable sub-state transition pairs \mathbb{T} and structural measure $\eta_{\mathcal{G}}$, as described in Sect. 4,

we must be able to identify transition probabilities $P_{\alpha,\beta}(\mu)$ between allowable sub-state transition pairs that depend only $\mu = \eta_G(S_i, S_j)$. Setting the transition rates to be proportional to the transition probabilities, i.e. $q_{\alpha,\beta} = \tau P_{\alpha,\beta}(\mu)$, results in a continuous-time Markov chain whose embedded discrete-time Markov chain is the discrete-time model of interest. Note that the constant of proportionality τ just sets the time-scale of the transition rates.

As an example, consider the voter model on a graph \mathcal{G} with N vertices. Recall that vertices in the network have opinions ± 1 and at each time-step a vertex is picked at random and adopts the state of a randomly chosen neighbour. Thus the set of vertex states is $\mathcal{W} = \{+1, -1\}$ and let $W_1 = +1$ and $W_2 = -1$. Since only one vertex ever changes vertex-state at any time-step, the set of allowable sub-states is $\mathbb{A} = \{(1, 0), (0, 1)\}$, i.e. a single $+1$ vertex or a single -1 vertex. If $\alpha = (1, 0)$ and $\beta = (0, 1)$, then the set of allowable sub-state transition pairs is $\mathbb{T} = \{(\alpha, \beta), (\beta, \alpha)\}$. Transition pairs are then all pairs of states that differ in vertex-state at only one vertex. The probability that a given vertex with vertex-state ± 1 is picked is $1/N$ and if it is picked then it changes vertex-state with probability n_{\mp}/k , where n_{\mp} is the number of neighbours it has with vertex-state ∓ 1 and k is its degree. Thus $\eta_G(S_i, S_j)$ should return (n_+, n_-, k) . Note that n_{\pm} and k are invariant under automorphisms, since automorphisms do not change the degree of a vertex and the action defined in (1) does not change the vertex-states of neighbours. Thus this choice of η_G is a structural measure. If we choose $\tau = N$, then the transition rates are $q_{\alpha,\beta} = n_-/k$ and $q_{\beta,\alpha} = n_+/k$.

The voter model is an example of a binary state-space model with single vertex-state transitions, and is a particular case of the general binary [25, 26] and multi-state [21, 41] models with single vertex-state transitions proposed in the corresponding references. In these models, $S_i, S_j \in \mathbb{S}$ are transitions pairs if they have a single transition vertex. If the set of vertex-states is $\mathcal{W} = \{W_1, W_2, \dots, W_M\}$, and the vertex-states of the transition vertex u are $S_i(u) = W_k$ and $S_j(u) = W_l$, then the transition rate from S_i to S_j is given by $f_{W_k, W_l}(n_1, n_2, \dots, n_M)$, where n_m is the number of neighbours of u whose vertex-states are W_m . Note that each n_m is invariant under automorphisms. Specific forms of the transition rate f_{W_k, W_l} that correspond to several examples of well known models of dynamics on networks are given in [21, 26]. We can easily identify the features of our general model from those in [21, 25, 26, 41]. The allowable sub-states are $\mathbb{A} = \{\alpha^k \mid 1 \leq k \leq M\}$, where α_m^k is one if $m = k$ and zero otherwise. The set of allowable sub-state transition pairs consists of all distinct pairs of α^k , i.e. $\mathbb{T} = \{(\alpha^k, \alpha^l) \mid 1 \leq k, l \leq M \text{ and } k \neq l\}$. The structural measure η_G returns $\mu = (n_1, n_2, \dots, n_M)$ and the transition rates are $q_{\alpha^k, \alpha^l}(\mu) = f_{W_k, W_l}(n_1, n_2, \dots, n_M)$.

The general model proposed in this paper also allows for multi vertex-state transitions. For the Twitter model described in Sect. 3, let the set of vertex-states be $\mathcal{W} = \{E, A\}$, where $W_1 = E$ denotes empty and $W_2 = A$ denotes a meme. Recall that a vertex is picked at random and if their vertex-state is A then they change the vertex-state of all of their neighbours to A . Thus the allowable sub-state transitions correspond to cases where groups of vertices all

in state E that share a neighbour in state A all change to state A . If k_{\max} is the largest degree of all vertices in \mathcal{V} , then the set of allowable sub-states is $\mathbb{A} = \{\alpha^k, \beta^k \mid 1 \leq k \leq k_{\max}\}$ where $\alpha^k = (k, 0)$ and $\beta^k = (0, k)$. The set of allowable sub-state transition pairs is $\mathbb{T} = \{(\alpha^k, \beta^k) \mid 1 \leq k \leq k_{\max}\}$. The transition vertices of a transition pair must share at least one neighbour that has vertex-state A , and the transition vertices are the only neighbours of these A vertices that have vertex-state E . Thus the transition probability is $P_{\alpha^k, \beta^k} = n_k/N$, where n_k is the number of vertices in vertex-state A whose only neighbours in vertex-state E are the k transition vertices. Since n_k only depends on the neighbourhoods of the transition vertices, it is invariant under automorphisms. If we choose the rate constant $\tau = N$ and structural measure $\eta_{\mathcal{G}}$ to return $\mu = n_k$, then the transition rates are $q_{\alpha^k, \beta^k}(n_k) = n_k$.

For the Schelling model, let the set of vertex-states be $\mathcal{W} = \{E, A, B\}$, where $W_1 = E$ denotes an unoccupied or empty site, and $W_2 = A$ and $W_3 = B$ are the different types of occupied vertices. Recall that a vertex in vertex-state A (B) is unhappy if the proportion of its neighbours in state A (B) is less than a threshold, denoted by ϕ . At each time-step, an unhappy vertex is selected at random and it chooses an empty location at random to move to, leaving its original vertex empty. Note that the total number of vertices in each vertex-state is conserved. The allowable sub-states are $\alpha = (1, 1, 0)$, i.e. one empty and one A vertex, and $\beta = (1, 0, 1)$, i.e. one empty and one B vertex. The set of allowable sub-state transitions are $\mathbb{T} = \{(\alpha, \alpha), (\beta, \beta)\}$. Note that a transition exchanges the vertex-states, but the sub-states remain the same. Note that, by definition, a pair of identical states can not form a transition pair. If an occupied vertex is unhappy, then it is selected at random with probability $1/n_u$, where n_u is the number of unhappy vertices. An unoccupied vertex is then selected at random with probability $1/n_E$, where n_E is the number of empty vertices. Let $\eta_{\mathcal{G}}$ return $\mu = (\delta, n_u)$, where $\delta = 1$ if the occupied transition vertex is unhappy and $\delta = 0$ otherwise. Note that δ and n_u only depend on the vertex-states in neighbourhoods, so are invariant under automorphisms. If we choose the rate constant $\tau = n_E$, then the transition rates are

$$q_{\alpha, \alpha}(\mu) = q_{\beta, \beta}(\mu) = \begin{cases} \frac{1}{n_u} & \text{if } \delta = 1, \\ 0 & \text{otherwise.} \end{cases}$$

The allowable sub-states and sub-state transition pairs, structural metrics and transition rates can be identified in a similar way for other models, such as the naming game, the Bonabeau model, the Sznajd model and majority rule model.

7 Discussion

In this paper we have introduced a general Markov chain model of dynamical processes on networks and connected it to a range of well known models. One of the most important assumptions of our model is vertex homogeneity,

i.e. that the dynamical behaviour is independent of the particular vertices involved. Relaxing this assumption would certainly weaken the state-space compression resulting from network symmetries, in addition to complicating the identification of transition pairs. However, it is our belief that such heterogeneity may not necessarily add qualitatively new dynamics. For example, in traffic modelling, the stability criteria of car-following models is qualitatively the same for both homogeneous and heterogeneous driver behaviour [63]. Our assumption that the transition rates depend only on a structural measure is fundamental to the ability to lump the state-space on the basis of graph automorphisms. While this may seem restrictive, if it were not true, the model would necessarily ignore connectivity structure and negate the point of posing the model on a network in the first place.

The work presented here focuses on the formulation of dynamical processes on networks as Markov chains. It's clear that for networks of even fairly small sizes (e.g 40 vertices) with binary vertex-states, working with the full state-space is prohibitive. However, there has recently been increased interest in the analysis of dynamics on small networks [35, 43] and exploiting symmetries can have a significant impact on the size of networks that can be studied [36]. It would also be interesting to understand what can be said about the dynamics on the full state-space from the 'microscopic' vertex transition rules, e.g. whether the dynamics are ergodic or the nature of any absorbing states. It has recently been shown that real-world networks have a surprising amount of symmetry [45]. In a future publication, we will explore the impact of such symmetries on lumping for the general model introduced here.

Exact analyses of Markov chain dynamics on networks are rare since authors typically resort to mean-field approximations. These can do surprisingly well, but it's unclear what model and network features ensure this [28]. The simplest mean-field approximation corresponds to the 'well-mixed' case, i.e. the fully connected network or complete graph. The simplicity of this case results from the symmetry present in the complete graph [41, 59]. It would be interesting to identify other graphs where symmetry gives rise to significant lumping and better dynamical approximations. However, our experience is that finding such graphs is extremely difficult as they necessarily have to have a large amount of symmetry, and this makes computing the lumped state-space difficult. Alternatively, there is scope to develop more accurate mean-field approximations, similar to those in [21, 25, 26], using the general model proposed here.

In summary, we have presented a general model of dynamics on networks. One of the main purposes of this general model is to show how a wide range of dynamical processes on networks can be formulated as Markov chains. Our hope is that this will help to formalise the analytical treatment of more complex and more realistic models, ultimately improving our understand of the phenomena that they represent.

References

1. Axelrod, R.: The dissemination of culture: a model with local convergence and global polarization. *J. Confl. Resolut.* **41**(2), 203–226 (1997)
2. Banisch, S., Lima, R.: Markov chain aggregation for simple agent-based models on symmetric networks: the voter model. *Adv. Complex Syst.* **18**(03n04), 1550011 (2015)
3. Banisch, S., Lima, R., Araújo, T.: Agent based models and opinion dynamics as markov chains. *Soc. Netw.* **34**(4), 549–561 (2012)
4. Baronchelli, A., Felici, M., Loreto, V., Caglioti, E., Steels, L.: Sharp transition towards shared vocabularies in multi-agent systems. *J. Stat. Mech.* **2006**(06), P06014 (2006)
5. Barrat, A., Barthelemy, M., Vespignani, A.: *Dynamical Processes on Complex Networks*. Cambridge University Press, Cambridge (2008)
6. Bass, F.M.: A new product growth for model consumer durables. *Manag. Sci.* **15**(5), 215–227 (1969)
7. Bernardes, A.T., Stauffer, D., Kertész, J.: Election results and the sznajd model on Barabasi network. *Eur. Phys. J. B* **25**(1), 123–127 (2002)
8. Boguná, M., Pastor-Satorras, R.: Epidemic spreading in correlated complex networks. *Phys. Rev. E* **66**(4), 047104 (2002)
9. Bonabeau, E., Theraulaz, G., Deneubourg, J.L.: Phase diagram of a model of self-organizing hierarchies. *Phys. A* **217**(3–4), 373–392 (1995)
10. Castellano, C., Marsili, M., Vespignani, A.: Nonequilibrium phase transition in a model for social influence. *Phys. Rev. Lett.* **85**(16), 3536 (2000)
11. Castellano, C., Muñoz, M.A., Pastor-Satorras, R.: Nonlinear q-voter model. *Phys. Rev. E* **80**(4), 041129 (2009)
12. Castellano, C., Pastor-Satorras, R.: Thresholds for epidemic spreading in networks. *Phys. Rev. Lett.* **105**(21), 218701 (2010)
13. Castellano, C., Vilone, D., Vespignani, A.: Incomplete ordering of the voter model on small-world networks. *Eur. Lett.* **63**(1), 153 (2003)
14. Castelló, X., Eguíluz, V.M., San Miguel, M.: Ordering dynamics with two non-excluding options: bilingualism in language competition. *New J. Phys.* **8**(12), 308 (2006)
15. Chen, P., Redner, S.: Majority rule dynamics in finite dimensions. *Phys. Rev. E* **71**(3), 036101 (2005)
16. Clifford, P., Sudbury, A.: A model for spatial conflict. *Biometrika* **60**(3), 581–588 (1973)
17. Daley, D.J., Kendall, D.G.: Epidemics and rumours. *Nature* **204**(4963), 1118 (1964)
18. Deffuant, G., Neau, D., Amblard, F., Weisbuch, G.: Mixing beliefs among interacting agents. *Adv. Complex Syst.* **3**(01n04), 87–98 (2000)
19. Eames, K.T., Keeling, M.J.: Modeling dynamic and network heterogeneities in the spread of sexually transmitted diseases. *Proc. Natl. Acad. Sci.* **99**(20), 13330–13335 (2002)
20. Epstein, J.M., Axtell, R.: *Growing Artificial Societies: Social Science from the Bottom Up*. Institution Press, Brookings (1996)
21. Fennell, P.G., Gleeson, J.P.: Multistate dynamical processes on networks: analysis through degree-based approximation frameworks. arXiv preprint [arXiv:1709.09969](https://arxiv.org/abs/1709.09969) (2017)
22. Fraleigh, J.B.: *A First Course in Abstract Algebra*. Pearson Education, India (2003)

23. Galam, S.: Minority opinion spreading in random geometry. *Eur. Phys. J. B* **25**(4), 403–406 (2002)
24. Glauber, R.J.: Time-dependent statistics of the Ising model. *J. Math. Phys.* **4**(2), 294–307 (1963)
25. Gleeson, J.P.: High-accuracy approximation of binary-state dynamics on networks. *Phys. Rev. Lett.* **107**(6), 68701 (2011)
26. Gleeson, J.P.: Binary-state dynamics on complex networks: pair approximation and beyond. *Phys. Rev. X* **3**(2), 021004 (2013)
27. Gleeson, J.P., Hurd, T., Melnik, S., Hackett, A.: Systemic risk in banking networks without Monte Carlo simulation. *Advances in Network Analysis and its Applications*, pp. 27–56. Springer, Berlin (2012)
28. Gleeson, J.P., Melnik, S., Ward, J.A., Porter, M.A., Mucha, P.J.: Accuracy of mean-field theory for dynamics on real-world networks. *Phys. Rev. E* **85**(2), 026106 (2012)
29. Gleeson, J.P., Ward, J.A., Osullivan, K.P., Lee, W.T.: Competition-induced criticality in a model of meme popularity. *Phys. Rev. Lett.* **112**(4), 048701 (2014)
30. Godsil, C., Royle, G.F.: *Algebraic Graph Theory*, vol. 207. Springer Science & Business Media, Berlin (2013)
31. Goldenberg, J., Libai, B., Muller, E.: Talk of the network: a complex systems look at the underlying process of word-of-mouth. *Mark. Lett.* **12**(3), 211–223 (2001)
32. Haldane, A.G., May, R.M.: Systemic risk in banking ecosystems. *Nature* **469**(7330), 351 (2011)
33. Hegselmann, R., Krause, U.: Opinion dynamics and bounded confidence models, analysis, and simulation. *J. Artif. Soc. Soc. Simul.* **5**(3) (2002)
34. Holley, R.A., Liggett, T.M.: Ergodic theorems for weakly interacting infinite systems and the voter model. *Ann. Probab.* 643–663 (1975)
35. Holme, P.: Shadows of the susceptible-infectious-susceptible immortality transition in small networks. *Phys. Rev. E* **92**(1), 012804 (2015)
36. Holme, P., Tupikina, L.: Epidemic extinction in networks: insights from the 12,110 smallest graphs. arXiv preprint [arXiv:1802.08849](https://arxiv.org/abs/1802.08849) (2018)
37. Kemeny, J.G., Snell, J.L., et al.: *Finite Markov Chains*, vol. 356. van Nostrand, Princeton (1960)
38. Kempe, D., Kleinberg, J., Tardos, É.: Maximizing the spread of influence through a social network. In: *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 137–146. ACM (2003)
39. Kijima, M.: *Markov processes for stochastic modeling*, vol. 6. CRC Press, Boston (1997)
40. Kirman, A.: Ants, rationality, and recruitment. *Q. J. Econ.* **108**(1), 137–156 (1993)
41. Kiss, I.Z., Miller, J.C., Simon, P.L.: *Mathematics of Epidemics on Networks*. Springer, Berlin (2017)
42. Lambiotte, R.: How does degree heterogeneity affect an order-disorder transition? *Eur. Lett.* **78**(6), 68002 (2007)
43. López-García, M.: Stochastic descriptors in an SIR epidemic model for heterogeneous individuals in small networks. *Math. Biosci.* **271**, 42–61 (2016)
44. MacArthur, B.D., Sánchez-García, R.J.: Spectral characteristics of network redundancy. *Phys. Rev. E* **80**(2), 026117 (2009)
45. MacArthur, B.D., Sánchez-García, R.J., Anderson, J.W.: Symmetry in complex networks. *Discret. Appl. Math.* **156**(18), 3525–3531 (2008)
46. Mellor, A., Mobilia, M., Redner, S., Rucklidge, A.M., Ward, J.A.: Influence of Luddism on innovation diffusion. *Phys. Rev. E* **92**(1), 012806 (2015)

47. Melnik, S., Ward, J.A., Gleeson, J.P., Porter, M.A.: Multi-stage complex contagions. *Chaos* **23**(1), 013124 (2013)
48. Mobilia, M., Petersen, A., Redner, S.: On the role of zealotry in the voter model. *J. Stat. Mech.* **2007**(08), P08029 (2007)
49. Mobilia, M., Redner, S.: Majority versus minority dynamics: phase transition in an interacting two-state spin system. *Phys. Rev. E* **68**(4), 046,106 (2003)
50. Motter, A.E., Lai, Y.C.: Cascade-based attacks on complex networks. *Phys. Rev. E* **66**(6), 065102 (2002)
51. Newman, M.: Networks. Oxford University Press, Oxford (2010)
52. Pastor-Satorras, R., Castellano, C., Van Mieghem, P., Vespignani, A.: Epidemic processes in complex networks. *Rev. Mod. Phys.* **87**(3), 925 (2015)
53. Pastor-Satorras, R., Vespignani, A.: Epidemic spreading in scale-free networks. *Phys. Rev. Lett.* **86**(14), 3200 (2001)
54. Pastor-Satorras, R., Vespignani, A.: Immunization of complex networks. *Phys. Rev. E* **65**(3), 036104 (2002)
55. Porter, M.A., Gleeson, J.P.: Dynamical Systems on Networks. Frontiers in Applied Dynamical Systems: Reviews and Tutorials, vol. 4. Springer, Berlin (2016)
56. Sanchez-Garcia, R.J.: Exploiting symmetry in network analysis. arXiv preprint [arXiv:1803.06915](https://arxiv.org/abs/1803.06915) (2018)
57. Schelling, T.C.: Models of segregation. *Am. Econ. Rev.* **59**(2), 488–493 (1969)
58. Schelling, T.C.: Dynamic models of segregation. *J. Math. Sociol.* **1**(2), 143–186 (1971)
59. Simon, P.L., Taylor, M., Kiss, I.Z.: Exact epidemic models on graphs using graph-automorphism driven lumping. *J. Math. Biol.* **62**(4), 479–508 (2011)
60. Sood, V., Redner, S.: Voter model on heterogeneous graphs. *Phys. Rev. Lett.* **94**(17), 178701 (2005)
61. Sznajd-Weron, K., Sznajd, J.: Opinion evolution in closed community. *Int. J. Mod. Phys. C* **11**(06), 1157–1165 (2000)
62. Vazquez, F., Krapivsky, P.L., Redner, S.: Constrained opinion dynamics: freezing and slow evolution. *J. Phys. A* **36**(3), L61 (2003)
63. Ward, J.A.: Instability in heterogeneous traffic. In: Proceedings of the Traffic Flow Theory and Characteristics Committee (2010)
64. Watts, D.J.: A simple model of global cascades on random networks. *Proc. Natl. Acad. Sci.* **99**(9), 5766–5771 (2002)

EvoNRL: Evolving Network Representation Learning Based on Random Walks

Farzaneh Heidari() and Manos Papagelis

York University, Toronto, ON M3J1P3, Canada
[{farzanah, papaggel}@eecs.yorku.ca](mailto:{farzanah,papaggel}@eecs.yorku.ca)

Abstract. Large-scale network mining and analysis is key to revealing the underlying dynamics of networks. Lately, there has been a fast-growing interest in learning random walk-based low-dimensional continuous representations of networks. While these methods perform well, they can only operate on static networks. In this paper, we propose a random-walk based method for learning representations of evolving networks. The key idea of our approach is to maintain a set of random walks that are consistently valid with respect to the updated network topology. This way we are able to continuously learn a new mapping function from the new network to the existing low-dimension network representation. A thorough experimental evaluation is performed that demonstrates that our method is both accurate and fast, for a varying range of conditions.

Keywords: Network representation · Evolving networks
Random walks

1 Introduction

Large-scale network mining and analysis is key to revealing the underlying dynamics of networks, not easily observable before. Traditional approaches to network mining and analysis inherit a number of limitations; typically algorithms do not scale well (due to ineffective representation of network data) and require domain-expertise. More recently and to address the aforementioned limitations, there has been a growing interest in learning *low-dimensional representations of networks*, also known as *network embeddings*. A comprehensive coverage can be found in the following surveys [5, 9, 19]. A family of these methods is based on performing random walks on a network. Random-walk based methods, inspired by the word2vec’s skip-gram model of producing word embeddings [13], establish an analogy between *a network* and *a document*. While a document is an ordered sequence of words, a network can effectively be described by a set of random walks (i.e., ordered sequences of nodes). Typical examples of these algorithms include DeepWalk [15] and node2vec [7]. In this work, we collectively refer to these random-walk based methods as StaticNRL. A typical StaticNRL method, is operating in two steps:

- (i) a set of random walks, *walks*, is collected by performing r random walks of length l starting at each node in the network (e.g., $r = 10$, $l = 80$).
- (ii) *walks* are provided as input to an optimization problem that is solved using variants of Stochastic Gradient Descent using a deep neural network [3]. The outcome is a set of d -dimensional representations, one for each node.

A major shortcoming of these network representation learning methods is that they can only be applied on *static networks*. However, in real-world, networks are continuously evolving, as nodes and edges are added or deleted over time. As a result, any previously obtained network representation will now be outdated having an adverse effect on the accuracy of the data mining task at stake.

The main objective of this paper is to develop methods for *learning representations of evolving networks*. The focus of our work is on random-walk based methods that are known to scale well. The naive approach to address this problem is to re-apply the random-walk based network representation learning method of choice every time there is an update to the network. But this approach has serious drawbacks. First, it will be very inefficient, because the embedding method is computationally expensive and it needs to run again and again. Then, the data mining results obtained by the subsequent network representations are not directly comparable to each other, due to the differences involved between the previous and the new set of random walks (lack of stability), as well as, the non-deterministic nature of the learning process (Sect. 2). Therefore the naive approach would be inadequate for learning representations of evolving networks.

In contrast to the naive approach, we propose a novel algorithm, EvoNRL, for Evolving Network Representation Learning based on random walks. The key idea of our approach is to design efficient methods that are incrementally updating the original set of random walks in such a way that it always respects the changes that occurred in the evolving network (Sect. 3). As a result, we are able to design a strategy for continuously learning a new mapping from the evolving network to a low-dimension network representation, by only updating a small number of random walks required to re-obtain the network embedding (Sect. 4). A thorough experimental evaluation on synthetic and real data sets demonstrates that the proposed method offers substantial time performance gains without loss of accuracy (Sect. 5). In addition, the method is generic, so it can accommodate a variant of random-walk based methods and the needs of different domains and applications.

2 Instability of StaticNRL Methods

In this paragraph, we present a systematic evaluation of the stability of the StaticNRL methods, similar to the one presented in [2]. The evaluation aims to motivate our approach to address the problem of interest. Intuitively, a *stable* embedding method is one in which successive runs of it on the same network would learn the same (or similar) embedding. StaticNRL methods are to a great degree dependent on two random processes: (i) the set of random walks collected, and (ii) the random initialization of the parameters of the optimization method.

Both factors can be a source of instability for the StaticNRL method. Comparing two embeddings can happen either by measuring their similarity or by measuring their distance. Let us introduce the following measures of instability:

- Embedding Similarity: Cosine similarity is a popular similarity measure for real-valued vector space models [8, 11]. Formally, given the vector representations \mathbf{n}_i and \mathbf{n}'_i of the same node n_i in two network embeddings obtained at two different times, their cosine similarity is: $sim(\mathbf{n}_i, \mathbf{n}'_i) = \cos(\theta) = \frac{\mathbf{n}_i \cdot \mathbf{n}'_i}{\|\mathbf{n}_i\| \|\mathbf{n}'_i\|}$. We can extend the similarity to two network embeddings \mathcal{E} and \mathcal{E}' by summing and normalizing over all nodes: $sim(\mathcal{E}, \mathcal{E}') = \sum_{i \in V} sim(\mathbf{n}_i, \mathbf{n}'_i) / |V|$.
- Embedding Distance: Given a graph $G = (V, E)$, a network embedding is a mapping $f : V \rightarrow \mathbb{R}^d$, where $d \ll |V|$. Let $F_t(V) \in \mathbb{R}^{|V| \times d}$ be the matrix of all node representations at time t . Then, similarly to the approach in [6], the distance of two embeddings \mathcal{E} , \mathcal{E}' is: $distance(\mathcal{E}, \mathcal{E}') = \|F(V) - F'(V)\|_F$.

Experimental Scenario: We design a controlled experiment on two real-world networks, namely Protein-Protein-Interaction (PPI) [4] and a collaboration network (dblp) [18] that aims to evaluate the effect of the two random processes (set of random walks, initialization weights) in the network embeddings. In these experiments, we compare three settings. For each setting, we run StaticNRL on a network (using parameter values: $r = 10$, $l = 10$, $k = 5$) two times (while there have been no change in the network), and compute the *cosine similarity* and the *matrix distance* of the two embeddings \mathcal{E} , \mathcal{E}' . We repeat the experiment 10 times and report averages. The three settings are:

- StaticNRL: independent set of random walks; random initialization weights.
- StaticNRL-i: independent set of random walks; same initialization weights.
- StaticNRL-rw-i: same set of random walks; same initialization weights.

Results and Implications: The results of the experiment are shown in Fig. 1a (cosine similarity) and Fig. 1b (matrix distance). They show that when we employ the same set of random walks and the same initialization in consecutive runs of StaticNRL, we are able to obtain the same embedding (as depicted by the $sim(\cdot, \cdot) = 1$ in Fig. 1a or $distance(\cdot, \cdot) = 0$ in Fig. 1b). However, when random walks and/or random initialization are employed in consecutive runs of a StaticNRL method, then the embedding can be shifted (lack of stability) despite the fact that there is no actual change in the network topology. Most of similar work in the literature correct this problem by applying an *alignment method* [8] that aims to rotationally align two embeddings. While alignment methods can bring independent embeddings closer and eliminate the effect of different embeddings, this approach won't work well in random walk based models. The main reason for that is that as we have showed in the experiment, consecutive runs suffer from instability that is introduced by both random processes. Therefore, changes that occur in the evolving network topology will not be easily interpretable by the changes observed in the network embedding (since differences might incorporate changes due to the two random processes). These observations highlight the challenges of employing StaticNRL methods for learning representations of evolving networks, which is the focus of this work.

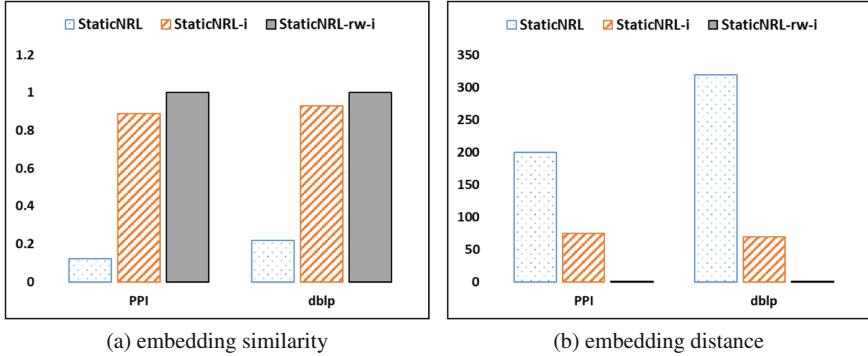


Fig. 1. Instability of the StaticNRL methods when applied on the same network.

3 Dynamic Random Walks Framework

In Sect. 2, we established the instability of random-walk based methods when they are repeatedly applied to the same static network. This motivated our key idea to address the problem: maintain a set of random walks that are consistently valid with respect to the network topology changes. If we are able to do so, we can effectively eliminate the effect of the random processes by, first, preserving, as much as possible, the original random walks that haven't been affected by the network changes. Then, by initializing the model with a previous run's initialization [11]. The main advantage of this approach is that any changes observed in the embedding of an evolving network will be more interpretable. Stemming from our key idea, in this Section we describe a general framework and a novel method that allows to incrementally update a set of random walks obtained on a network $G_t(V_t, E_t)$ at time t so that they remain valid at time $t' > t$ and network $G_{t'}(V_{t'}, E_{t'})$. The most common change in an evolving network consists of either *adding a new edge* or *deleting an existing edge*. While our method is able to handle both cases (with minor variations), for the rest of the manuscript we discuss only the case of edge addition, due to space limitations. Therefore, $G_{t'}$ represents an augmentation of G_t , where $V_{t'} = V_t$ and $E_{t'} = E_t \cup E^+$, and E^+ represents the set of new edges in $G_{t'}$.

Dynamic Updates of Random Walks: Given a network $G_t = (V_t, E_t)$ at time t , we employ a standard StaticNRL method¹ to simulate random walks. By default, this method is configured to perform $r = 10$ random walks per node, each of length $l = 80$. Let RW_t be the set of random walks obtained, where $|RW_t| = |V_t| \times r$. We store the random walks in memory, using a data structure that provides random access to its elements (i.e., a 2-D numpy matrix). Now, assume that a single new edge $e_{ij} = (node_i, node_j)$ arrives in the network at time $t + 1$, so $E_{t+1} = E_t \cup (node_i, node_j)$. There are two operations that need

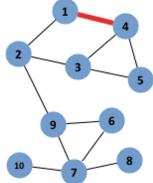
¹ node2vec; code is available at <https://github.com/aditya-grover/node2vec>

to take place in order to properly update the set RW_t of the random walks in hand:

- Operation 1: contain the new edge to existing random walks in RW_t .
- Operation 2: discard obsolete parts of random walks of RW_t and replace them with new random walks to form the new RW_{t+1} .

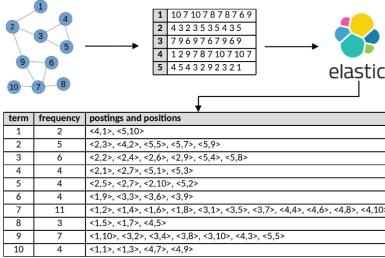
Operation 1: We want to update the set RW_t to contain the new edge $(node_i, node_j)$. We explain the update process for $node_i$; the same process is followed for $node_j$. First, we need to find all the random walks $walks_i \in RW_t$ that include $node_i$. Then, we need to update them so as to reflect the existence of the new edge $(node_i, node_j)$. In practice, the new edge offers a new possibility for each random walk in G_{t+1} that reaches $node_i$ to traverse $node_j$ in the next step. The number of these random walks that include $(node_i, node_j)$ depends on the node degree of $node_i$ and it is critical for correctly updating random walks in RW . Formally, if the node degree of $node_i$ in G_t is d_t then in G_{t+1} it will be incremented by one, $d_{t+1} = d_t + 1$. Effectively, a random walk that visits $node_i$ in G_{t+1} would have a probability $\frac{1}{d_{t+1}}$ to traverse $node_j$. This means that if there are $freq_i$ occurrences of $node_i$ in RW_t , then $\frac{freq_i}{d_{t+1}}$ edges $(node_i, node_j)$ need to be contained, by setting the next node of $node_i$ to be $node_j$, in the current random walk. If $node_i$ is the last node in a random walk then, there is no need to update the new edge in that random walk. The *naive approach* to perform the updates is to visit all $freq_i$ occurrences of $node_i$ in $walks_i \in RW$ and for each of them to decide whether to perform an update of the random walk (or not), by setting the next node to be $node_j$. The decision is based on tossing a biased coin, where with probability $p_{success} = \frac{1}{d_{t+1}}$ we update the random walk, and with probability $p_{failure} = 1 - p_{success}$ we do not. While this method is accurate, it is not efficient as all occurrences of $node_i$ need to be examined, when only a portion of them needs to be updated. A *faster approach* is to find all the $freq_i$ occurrences of $node_i$, and then to uniformly at random sample $\frac{freq_i}{d_{t+1}}$ of them and update them by setting the next node to be $node_j$. While this method will be faster than the naive approach, it still resides on finding all the $freq_i$ occurrences of $node_i$ in the set of random walks RW , which is an expensive operation. We employ the faster approach for the updates. We will soon describe how this method can be accelerated by using an efficient indexing library that allows for fast querying and retrieval of all occurrences of a node in random walks.

Operation 2: Once a new edge $(node_i, node_j)$ is contained in an existing random walk, it renders the rest of it obsolete. We replace the remainder of the random walk by simulating a new random walk on the updated network G_{t+1} . The random walk starts at $node_j$ with length $l_{sim} = l - (Ind_i + 1)$, where $0 \leq Ind_i \leq l - 1$ is the index of $node_i$ in the currently updated random walk. Figure 2a presents an illustrative example of how updates of random walks work. First, a set of random walks RW_t are obtained (say 5 as illustrated by the upper lists of random walks). Let us assume that a new edge $(1, 4)$ arrives. Note that now, the degree of node 1 and node 2 will increase by 1 ($d_{t+1} = d_t + 1$). Because of the new edge, some random walks need to be updated to account for the change



(a) Example addition of a new edge (1,4). Random walks need to be updated to adhere to the updates of the network.

RW #1 (original)	10 7 10 7 8 7 8 7 6 9
RW #1 (updated)	10 7 10 7 8 7 8 7 6 9
RW #2 (original)	4 3 2 3 5 3 5 4 1 2
RW #2 (updated)	4 3 2 3 5 3 5 4 1 2
RW #3 (original)	7 9 6 9 7 6 7 9 6 9
RW #3 (updated)	7 9 6 9 7 6 7 9 6 9
RW #4 (original)	1 2 9 7 8 7 10 7 10 7
RW #4 (updated)	1 4 3 4 3 5 4 1 4 3
RW #5 (original)	4 5 4 3 2 9 2 3 2 1
RW #5 (updated)	4 5 4 1 4 5 3 4 5 3



(b) Example inverted random walk index. Given a graph, five random walks are performed and indexed.

Fig. 2. Illustrative examples of the update method in both naive and faster approach.

in the topology. To perform the updates, we first search for all occurrences of i , $freq_i$. Then, we uniformly at random sample $\frac{freq_i}{d_{t+1}} = 2/2 = 1$ of them to determine where to contain the new edge. In the example, node 4 is listed after node 1 (i.e., the second node of the 4th random walk is now updated). The rest of the current random walk is obsolete, so it needs to be replaced. To perform the replacement a new random walk is simulated on the updated network G_{t+1} that starts at node 4 and has a length of $l_{sim} = l - (Ind_1 + 1) = 10 - (0 + 1) = 9$. The same process is repeated for node 4 of the added edge (1,4) (see the updates on the 2nd and 5th random walk, respectively). The details of the proposed algorithm are described in Algorithm 1. Lines 2 and 12 of the algorithm invoke a *Query* operator. This operator is responsible for searching and retrieving information about all the occurrences of $node_i$ in the set of the random walks RW_t . In addition, lines 11 and 19 of the algorithm invoke a *UpdateRandomWalks* operator. This operator is responsible for updating any obsolete random walks of RW_t with the updated ones to form the new set of random walks RW_{t+1} , valid to G_{t+1} . However, these operators are very computationally expensive, especially for larger networks. In the next paragraph, we describe how these two slow operators, *Query* and *UpdateRandomWalks*, can be replaced by similar operators offered off-the-shelf by high performance indexing and searching open-source technologies. In addition, so far, we have relied on maintaining the set of random walks RW_t in memory. The indexing technologies we will employ scale well to very large number of random walks.

Efficient Query and Update of Random Walks: Updating random walks methods presented in the previous paragraph are accurate. However, they depend on operators *Query* and *UpdateRandomWalks* that are computationally expensive and are not scalable. The most expensive operation is to search and update the random walks RW_t with the occurrences of $node_i$ and $node_j$ of the new edge ($node_i, node_j$). To address these shortcomings, our framework relies on popular open-source indexing and searching technologies. We build an *inverted random walk index*, I^{RW} . I^{RW} is an index data structure that stores a mapping from

nodes (terms) to *random walks* (documents). The purpose of I^{RW} is to enable *fast querying* of nodes in random walks, and *fast updates* of random walks that can inform Algorithm 1. Figure 3 provides an example of a small inverted random walk index. We also describe how to create the index and use it in our setting.

Algorithm 1 Update RW

```

1: procedure UpdateWalks
2:    $walks_i \leftarrow \text{Query}(node_i)$ 
3:    $p_i \leftarrow \frac{1}{d^l}$ 
4:    $p_j \leftarrow \frac{1}{d^l}$ 
5:    $s_i \leftarrow \text{Sample}(walks_i, p_i)$ 
6:   if  $\text{len}(s_i) > 0$  then
7:     for  $wk$  in  $s_i$  do
8:        $Ind_i \leftarrow \text{Position}(node_i, wk)$ 
9:        $l_{sim} = l - (Ind_i + 1)$ 
10:       $wk[Ind_i+1:] \leftarrow \text{SimulateWalk}(node_j, l)$ 
11:    UpdateRandomWalks()
12:     $walks_j \leftarrow \text{Query}(node_j)$ 
13:     $s_j \leftarrow \text{Sample}(walks_j, p_j)$ 
14:    if  $\text{len}(s_j) > 0$  then
15:      for  $wk$  in  $s_j$  do
16:         $Ind_j \leftarrow \text{Position}(node_j, wk)$ 
17:         $l_{sim} = l - (Ind_j + 1)$ 
18:         $wk[Ind_j+1:] \leftarrow \text{SimulateWalk}(node_i, l)$ 
19:    UpdateRandomWalks()

```

Indexing Random Walks: We obtain the initial set of random walks RW_t at time t by performing random walks on the original network, similarly to the process followed in standard Static-NRL methods. Each random walk is transformed to a document by properly concatenating the ids of the nodes in the walk. For example, a short walk ($x \rightarrow y \rightarrow z$) will be represented as a document with content “x y z”. These random walks are indexed to create I^{RW} . It is important to note that once an index is available, there is no need to maintain the random walks in memory any more.

Querying Random Walks: We rely on the index I^{RW} to perform any *Query* operation. Besides searching and retrieving all random walks that contain a specific $node_i$, the index I^{RW} can provide useful quantities of interest like the frequency of $node_i$ in a set of random walks, and the position of $node_i$ in a random walk.

Updating Random Walks: We rely on the index I^{RW} for any *UpdateRandomWalks* operation. While querying an inverted index is a fast process, updating it is slower. Therefore, the performance of our methods is dominated by the number of random walk updates required. Still, our methods would perform multitude of times faster than StaticNRL methods. A detailed analysis of this issue is provided in Sect. 5. Additional optimizations are available as a result of employing an inverted index. For instance, we can take advantage of *bulk updates*. This will sacrifice accuracy for speed that might be preferable in some applications.

4 Evolving Network Representation Learning

So far, we have described our framework for maintaining an always valid set of random walks RW_t at time t . Recall that our final objective is to be able to

learn a representation of the evolving network. The process begins by obtaining an initial network embedding. For that, we resort to the StaticNRL method of choice. EvoNRL has the overhead of first indexing the set of initial random walks RW . At this time only ($t = 0$), we initialize the skip-gram model, but we store and re-use the same initialization weights for the needs of subsequent embeddings. As new edges are arriving, EvoNRL performs the necessary updates as described earlier. At each time t a valid set of random walks RW_t is available that can be used to obtain an updated network embedding. While re-embedding the network at every time t will result in more accurate embeddings, we only need to do this once in a while. The timing of the embedding is domain-specific and relates to the trade-off between *accuracy* and *performance* that an application can accommodate. In Sect. 5 we present experiments that can support the decision making process. For completeness, we describe next how to obtain an embedding of the evolving network at time t , given a set of random walks RW_t .

Learning Embeddings Given RW_t : Given a network $G_t = (V_t, E_t)$, our goal is to learn the network representation $F(V_t)$ using the skip-gram model. $F(V_t)$ is a $|V_t| \times d$ matrix where each row is the vector representation of a node and d is the vector's dimension. The *context* of each node n_i is found using the valid RW_t set, similar to works [7, 15]. To obtain an embedding we optimize the skip-gram with negative sampling objective, using stochastic gradient decent, so that: $Pr(n_j|n_i) \propto \exp(\mathbf{n}_j^T \mathbf{n}_i)$, where \mathbf{n}_i is the vector representation of a node n_i ($F(n_i) = \mathbf{n}_i$). $Pr(n_j|n_i)$ is the probability of the observation of neighbor node n_j , within the window-size given that the window contains n_i .

5 Experimental Evaluation

In this section, we experimentally evaluate the performance of our dynamic random walk framework and EvoNRL. We aim to answer the following questions:

- **Q1** How the importance of a new edge affects the random walks update time?
- **Q2** How the network topology affects the number of random walks updated?
- **Q3** What is the time performance of EvoNRL?
- **Q4** What is the accuracy performance of EvoNRL?

Environment: All experiments are conducted on a workstation with 8x Intel(R) Core(TM) i7-7700 CPU @ 3.60 GHz and 64 GB memory. For the embeddings, we employ the gensim implementation of the skip-gram model². We set the context-size $k = 5$ and the number of dimensions $d = 128$. We use Python 3.6.

Data: We experiment with real networks coming from a Protein-to-Protein interaction network (*PPI* [4]) and a social network of bloggers (*BlogCatalog* [16]). In addition, we experiment with a set of synthetic networks of different topologies, obtained using the Watts-Strogatz model [14]. In particular, we set the model's rewiring probability parameter p , so as to obtain a *Lattice* ($p = 0.0$), a *Small-world* ($p = 0.5$) and an *Erdos-Reyni* ($p = 1.0$) network, respectively. Details are shown in Table 1.

² <https://github.com/RaRe-Technologies/gensim>

Table 1. Description of the network data sets employed in the experimental evaluation

Name	Type	# Nodes	# Edges	Description
PPI	real	3,890	76,584	50 different labels
BlogCatalog	Real	10,312	333,983	39 different labels
{Lattice, Small-world, ER}	Synthetic	10,000	70,000	$p = \{0, 0.5, 1.0\}$, respectively

Q1 Effect of New Edge Importance: It is easy to see that even a single new edge can have a dramatic effect in the number of random walks affected. Apparently, that number controls the time needed to update the affected random walks in our framework. In this set of experiments we perform a systematic analysis of the effect of the importance of an arriving edge to the time required for the update. Importance of an incoming edge $e_{ij}^{t+1} = (n_i, n_j)$ at time $t + 1$ in a network can be defined in different ways. Here, we define three metrics of edge importance, based on properties of the nodes n_i and n_j that form the **endpoints** of an arriving edge. These include the *sum of frequencies* of edge endpoints in RW_t , *sum of the node degrees* of edge endpoints in G_t and *sum of the node-betweenness* of edge endpoints in G_t . Results are presented in Fig. 3. The first observation is that important incoming edges are more expensive to update, up to three or four times (1.6sec vs 0.4sec). This is expected, as more random walks need to be updated. However, the majority of the edges are of least importance (lower left dense areas in Fig. 3a, b, c), so fast updates are more common. Finally, the behavior of sum of frequencies (Fig. 3a) and sum of degrees (Fig. 3b) of the edge endpoints are correlated. This is because the node degree is known to be directly related to the number of random walks that traverse it. On the other hand, node-betweenness demonstrates more unstable behavior since it is related to shortest paths and not just paths (which are traversed in randw3om walks).

Q2 Effect of Network Topology: We evaluate the effect of randomly adding new edges in networks of different topologies, but same size ($|V| = 10,000$). For each case, we report the number of the random walks that need to be updated. Fig. 4 shows the results, where it becomes clear that as more new edges are added, more random walks are affected. The effect is more stressed in the case of the *Small-world* and *Erdos-Reyni* networks; these networks have small diameter, therefore every node is easily accessible from any other node. In contrast, *Lattices* have larger diameter and nodes are more equally distributed in random walks.

Q3 Time Performance of EvoNRL: To evaluate the time performance of EvoNRL we run experiments on two *Small-world* networks (Watts-Strogatz ($p = 0.5$)) of different size ($|V| = \{1000, 10000\}$). We evaluate EvoNRL against a standard StaticNRL method from the literature [7]. Both algorithms start with the same set of random walks RW . As new edges are arriving, StaticNRL needs to learn a new network representation by re-simulating a new set of walks every time. On the other hand, EvoNRL has the overhead of first indexing the set of initial random walks RW . Then, for every new edge it just needs to perform

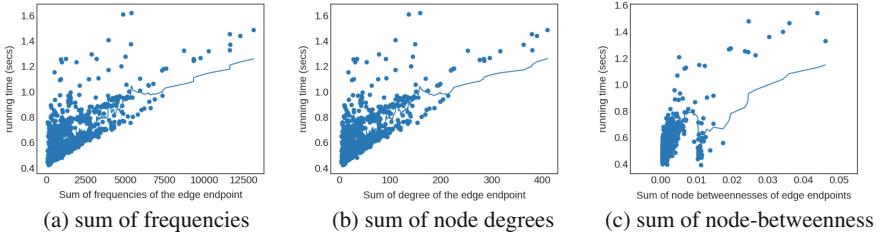


Fig. 3. Effect of the new edge importance to the running time of updates (PPI is used).

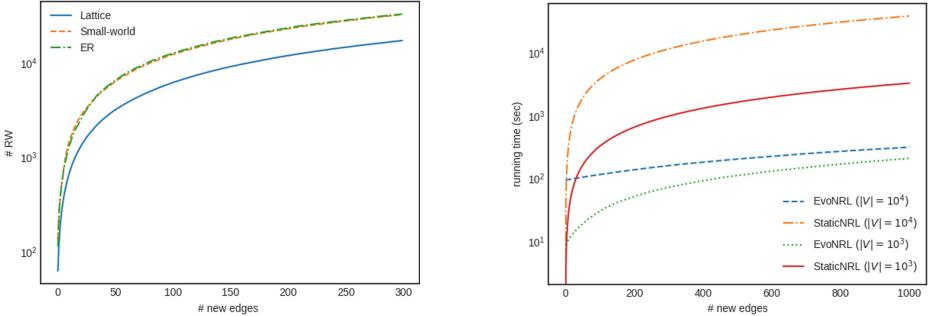


Fig. 4. Effect of network topology.

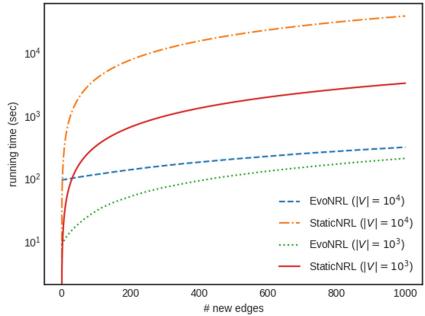


Fig. 5. Time performance of EvoNRL.

the necessary updates as described earlier. Figure 5 shows the results, where it can be seen that the performance of StaticNRL is linear to the number of new edges. At the same time, EvoNRL can accommodate the changes more than 100 times faster than staticNRL. This behavior is even more stressed in the larger network (where the number of nodes is larger). By increasing the number of nodes, running StaticNRL becomes significantly slower, because it needs to simulate a larger amount of random walks. On the other hand, EvoNRL has a larger initialization overhead, but after that it can easily accommodate new edges. This is because every update is only related to the number of random walks affected and not the size of the network. This is an important observation, as it means that the benefit of EvoNRL will be more stressed in larger networks.

Q4 Accuracy Performance of EvoNRL: We run experiments that demonstrate that EvoNRL has a similar accuracy to that obtained by StaticNRL, when it is run again and again on instances of an evolving network (*Blog-Catalog*, *PPI*) and evaluate the accuracy on a downstream data mining task: *multi-label classification*. In our multi-label classification experiments, we see 50% of nodes and their labels in the training phase and the goal is to predict labels of the rest of the nodes. We use node vector representations as input to a one-vs-rest logistic regression classifier with L2 regularization. Adding new edges

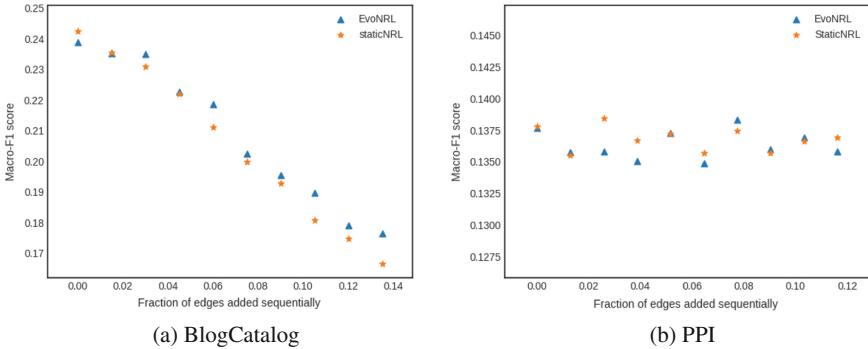


Fig. 6. Accuracy performance of EvoNRL for the *BlogCatalog* and the PPI network.

will impact the network embedding and thus the overall accuracy of the classification results. It is important to note here that we only care about observing similar trends in the accuracy results of both methods, and not about the actual accuracy values. For both experiments we report the $\text{Macro-}F_1$ accuracy of the multi-label classification task as a function of the fraction of new edges added. For StaticNRL, since it is sensitive to the new set of random walks, we run it 10 times and report the averages. Figure 6 shows the results. We observe that the Macro- F_1 accuracy of EvoNRL follows the same trend as the one of StaticNRL in both the *BlogCatalog* (Fig. 6a) and the *PPI* (Fig. 6b) networks. It can be seen that the accuracy of the two methods remains similar as more edges are added. This provides strong evidence that our random walk updates are accurate and able to reproduce the accuracy results obtained by applying a StaticNRL method on multiple instance of the evolving network.

6 Related Work

A comprehensive coverage of methods for *learning network representations of static networks* can be found in [5, 9, 19]. Work on *learning representations of dynamic networks* often apply static methods to snapshots of an evolving network [8]. Similarly, graph factorization approaches attempt to learn the embedding of dynamic graphs by smoothing over consecutive snapshots [1]. DANE [12] is a dynamic representation framework that focuses on attributed networks. Know-Evolve [17] proposes an entity embedding method of an evolving knowledge-graph based on multivariate event detection. EvoNRL does not need to operate on snapshots of the evolving network; instead, it directly learns the evolving network representations by monitoring the changes in the topology. Our work is also related to work on *dynamic random walks*. For instance, READS[10] is an indexing scheme for Simrank computation in dynamic graphs. EvoNRL has different semantics, sampling strategy and application focus to READS.

7 Conclusions

Our focus in this paper is on learning representations of evolving networks. To extend static random walk based network representation methods to evolving networks, we proposed a general framework for updating random walks as new edges are arriving. The updated random walks leverage time and space efficiency of inverted indexing methods. Our proposed method, EvoNRL, utilizes the continuously valid set of random walks to obtain new network representations that respect the changes that occurred in the network. We demonstrated that our proposed method, EvoNRL, is both **accurate** and **fast**. Therefore, it can be successfully employed in a number of predictive tasks that arise in the study of networks that evolve over time.

References

1. Ahmed, A., Shervashidze, N., Narayananamurthy, S., Josifovski, V., Smola, A.J.: Distributed large-scale natural graph factorization. ACM (2013)
2. Antoniak, M., Mimno, D.: Evaluating the stability of embedding-based word similarities. TACL **6**, 107–119 (2018)
3. Bengio, Y., Courville, A., Vincent, P.: Representation learning: a review and new perspectives. IEEE TPAMI **35**(8), 1798–1828 (2013)
4. Breitkreutz, B.J., et al.: The biogrid interaction database: 2008 update. Nucleic Acids Res. **36**(suppl 1), D637–D640 (2007)
5. Cai, H., Zheng, V.W., Chang, K.: A comprehensive survey of graph embedding: problems, techniques and applications. IEEE TKDE (2018)
6. Goyal, P., Kamra, N., He, X., Liu, Y.: Dyngem: deep embedding method for dynamic graphs. In: IJCAI Workshop on Representation Learning for Graphs (2018)
7. Grover, A., Leskovec, J.: node2vec: Scalable feature learning for networks. In: KDD, pp. 855–864 (2016)
8. Hamilton, W.L., Leskovec, J., Jurafsky, D.: Diachronic word embeddings reveal statistical laws of semantic change. CoRR [abs/1605.09096](https://arxiv.org/abs/1605.09096) (2016)
9. Hamilton, W.L., Ying, R., Leskovec, J.: Representation learning on graphs: methods and applications. IEEE Data Eng. Bull. **40**(3), 52–74 (2017)
10. Jiang, M., Fu, A.W.C., Wong, R.C.W.: Reads: a random walk approach for efficient and accurate dynamic simrank. Proc. VLDB Endow. **10**(9), 937–948 (2017)
11. Kim, Y., Chiu, Y., Hanaki, K., Hegde, D., Petrov, S.: Temporal analysis of language through neural language models. CoRR [abs/1405.3515](https://arxiv.org/abs/1405.3515) (2014)
12. Li, J., Dani, H., Hu, X., Tang, J., Chang, Y., Liu, H.: Attributed network embedding for learning in a dynamic environment. In: CIKM, pp. 387–396 (2017)
13. Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. In: NIPS, pp. 3111–3119(2013)
14. Newman, M.E.: The structure and function of complex networks. SIAM Rev. **45**(2), 167–256 (2003)
15. Perozzi, B., Al-Rfou, R., Skiena, S.: Deepwalk: online learning of social representations. CoRR [abs/1403.6652](https://arxiv.org/abs/1403.6652) (2014)
16. Reza, Z., Huan, L.: Social computing data repository. <http://socialcomputing.asu.edu/>

17. Trivedi, R., Dai, H., Wang, Y., Song, L.: Know-evolve: deep temporal reasoning for dynamic knowledge graphs. *ICML* **70**, 3462–3471 (2017)
18. Yang, J., Leskovec, J.: Defining and evaluating network communities based on ground-truth. *KAIS* **42**(1), 181–213 (2015)
19. Zhang, D., Yin, J., Zhu, X., Zhang, C.: Network representation learning: a survey. *IEEE Trans. Big Data* (2018)



Distributed PI Control for Multi-agent Consensus Tracking of Heterogeneous Networks with Heterogeneous Uncertainties

Yuting Feng¹(✉), Zhisheng Duan¹, and Guanrong Chen²

¹ Department of Mechanics and Engineering Science, College of Engineering, Peking University, Beijing, China

{fengyuting,duanzs}@pku.edu.cn

² Department of Electronic Engineering, City University of Hong Kong, Hong Kong SAR, China

eegchen@cityu.edu.hk

Abstract. This paper considers the multi-agent consensus tracking problem of heterogeneous networks with parameter uncertainties, unmodeled dynamics, bounded external disturbances and Lipschitz-type disturbances. By linearly parameterizing the control input of the leader and the unknown dynamics of all followers, two distributed proportional-integral (PI) protocols with dynamic and static coupling gains respectively are proposed to ensure tracking consensus. A simulation example is provided to verify the theoretical results.

Keywords: Heterogeneous networks · Distributed PI control
Robustness

1 Introduction

In the recent decade, collective behaviors of multi-agent systems have attracted increasing attention from the control community since it is a fundamental problem in networked systems [1–3]. As one of the most important and basic issues in cooperative control, consensus has been extensively studied [4–6]. In general, the consensus problem can be classified as leaderless consensus and leader-follower consensus. In most existing literature, the agents are considered to be identical and do not have any inner connections with each other [7–9]. However, many social networks have heterogeneous dynamics due to, for example, different shapes or load abilities [10–12]. Meanwhile, real-world networks are usually coupled with hard-wired links, which make the inner topology and the control topology coexisting in a complex network [13, 14].

In this paper, we address the multi-agent consensus tracking problem of heterogeneous networks with heterogeneous uncertainties including parameter uncertainties, unmodelled dynamics and bounded external disturbances. In [15],

the tracking problem is solved for agents with external disturbances and uncertainties from a neural network approach. In [16], distributed tracking protocols are developed for the integrator-type systems with unknown disturbances. Recently, in [17], an adaptive consensus protocol is proposed for general linear multi-agent systems based on parameterizations of the unknown dynamics. It should be noted that the above-mentioned references are all concerned about the consensus of homogeneous multi-agent systems. To reach output consensus, an internal model principle is presented based on the matching condition in [19]. To achieve static consensus of a heterogeneous network with disturbances, PI control was utilized in [11, 20], where different control layers are used in proportional and integral actions. In [21, 22], the tracking problem of a heterogeneous network is solved by introducing a virtual leader, where the communication graph is directed.

Inspired by the previous works, we focus on the consensus tracking problem for heterogeneous systems with inner connections under heterogeneous uncertainties, which are more common in applications but also more challenging. The main contributions of this paper lie in the following aspects. First, we propose two PI control protocols with dynamic and static coupling gains respectively in the presence of inner physical connections and heterogeneous uncertainties. It is notable that the PI control protocol is fully distributed in the dynamic coupling case. Second, contrarily to the work in [23] where the control input of the leader is inaccessible to the followers, in this paper the input is parameterized linearly and identified by the followers. Furthermore, the adaptive control protocol is robust against heterogeneous uncertainties without using the bounds of the parameter uncertainties. Finally, the handling of both inner physical connections and the communication topology in this scenario is more complex and technically challenging. Compared with the work in [24], this paper considers a more general situation where both heterogeneous dynamics and heterogeneous uncertainties exist.

The rest of this paper is organized as follows. Some preliminaries and the model formulation are given in Sect. 2. Section 3 presents the main results, in which two distributed PI control protocols are proposed. Numerical example is given in Sect. 4. The last section concludes the paper.

2 Preliminaries and Problem Statement

2.1 Graph Theory

Let $R^{N \times M}$ be the set of $N \times M$ real matrices, I_N be the $N \times N$ identity matrix, $\mathbf{1}_N$ be the N -dimensional column vector with each entry being 1. Notation $diag(\delta_1, \dots, \delta_N)$ represents a diagonal matrix with diagonal entries δ_1 to δ_N . Let σ_{max} denote the maximal singular value of a matrix. The Kronecker product of matrices A and B is denoted by $A \otimes B$. For a matrix X , $\|X\|$ represents its spectral norm, i.e., $\|X\| = \sqrt{\lambda_{max}(X^T X)}$. For a vector $x \in R^p$, $\|x\|_1$ and $\|x\|_\infty$ represent the 1-norm and ∞ -norm respectively. $sgn(x) = [sgn(x_1), \dots, sgn(x_p)]^T$, where $sgn(\cdot)$ is the signum function.

An undirected simple graph $\tilde{\mathcal{G}} = (\tilde{\mathcal{V}}, \tilde{\mathcal{E}})$ is composed of a set of nodes $\tilde{\mathcal{V}} = v_1, \dots, v_N$, a set of edges $\tilde{\mathcal{E}} \subset \tilde{\mathcal{V}} \times \tilde{\mathcal{V}}$. Associated adjacency matrix $\mathcal{A} = [a_{ij}] \in R^{N \times N}$ of the graph $\tilde{\mathcal{G}}$ is defined by $a_{ii} = 0$, and $a_{ij} = a_{ji} = 1$ if i and j are neighbors; $a_{ij} = 0$, otherwise. Correspondingly, the Laplacian matrix $\mathcal{L} = [\mathcal{L}_{ij}] \in R^{N \times N}$ is defined by $\mathcal{L}_{ii} = \sum_{j \neq i} a_{ij}$, $\mathcal{L}_{ij} = -a_{ij}$, for $i \neq j$.

Lemma 1 [1]. *Zero is an eigenvalue of L with $\mathbf{1}$ as a right eigenvector, and all nonzero eigenvalues have positive real parts. Furthermore, zero is a simple eigenvalue of L if and only if \mathcal{G} has a directed spanning tree.*

Lemma 2 (Barbalet's Lemma). *If the differentiable function $f(t)$ has a finite limit as $t \rightarrow \infty$, and if $\dot{f}(t)$ is uniformly continuous, then $\dot{f}(t) \rightarrow 0$ as $t \rightarrow \infty$.*

2.2 Problem Formulation

Consider a class of nonlinear multi-agent systems with N linearly coupled heterogeneous agents indexed with $1, \dots, N$, and one leader indexed with 0,

$$\begin{aligned} \dot{x}_i &= A_i x_i + \sum_{j=1}^N H a_{ij} (x_i - x_j) + u_i + b_i + w_i + f(x_i, t) \\ \dot{x}_0 &= A_0 x_0 + u_0 + f(x_0, t) \end{aligned} \quad (1)$$

where $x_i \in R^n$ and $u_i \in R^n$ denote the state and control input of the follower i ($i = 1, \dots, N$), x_0 and u_0 denote the state and control input of the leader, $A_i \in R^{n \times n}$ is the dynamics matrix, a_{ij} is the (i, j) th entry of the adjacency matrix \mathcal{A} associated with the inherent fixed connection graph \mathcal{G}_1 among agents, L is the corresponding Laplacian matrix, H is the inner coupling matrix describing the interconnections among components of $(x_i - x_j)$, b_i is some time-varying disturbances including the unknown external disturbances and unmodeled dynamics.

Assumption 1. There exists positive constants μ_i such that $|b_i| \leq \mu_i, i = 1, \dots, N$.

Assumption 2. The nonlinear function $f(\cdot, t)$ is assumed to satisfy the Lipschitz condition

$$\|f(x, t) - f(y, t)\| \leq \eta \|x - y\|, \quad \forall x, y \in R^n, \quad \forall t \geq 0, \quad (2)$$

where η is a nonnegative constant.

For the leader-follower system, control input u_0 and the disturbances w_i are linearly parameterized as

$$u_0 = W_0 \phi_0, \quad w_i = W_i \phi_i, \quad (3)$$

where $\phi_0, \phi_i \in R^p$ are the base function vectors and $W_0, W_i \in R^{(n \times p)}$ are unknown constant matrices to be adaptively updated.

Remark 1. The model proposed in this paper considering the unmodeled dynamics, parametric uncertainties and external disturbances together. Compared with [17], we focus on the heterogeneous dynamics with heterogeneous uncertainties, which is much closer to reality but also much challenging. Moreover, instead of constant disturbances assumed in [20], this paper allows the general time-varying disturbances which are only assumed to be bounded.

The objective here is to design distributed adaptive robust protocols such that each state of the follower converges to the state of the leader:

$$\lim_{t \rightarrow \infty} \|x_i(t) - x_0(t)\| = 0, \quad \forall i = 1, \dots, N.$$

3 Main Results

3.1 Distributed PI Control Design

Based on the relative states of neighboring agents, we propose a distributed PI control protocol for each follower, as follows:

$$\begin{aligned} u_i &= -\sigma_P \sum_{j=0}^N a'_{ij}(x_i - x_j) - \sigma_I \int_0^t e_i(\tau) \theta(e_i(\tau)) d\tau + \hat{W}_{i0} \phi_0 - \hat{W}_i \phi_i - \gamma \operatorname{sgn} \left(\sum_{j=0}^N a'_{ij}(x_i - x_j) \right) \\ d_i &= \|e_i\|, \quad i = 1, \dots, N \end{aligned} \tag{4}$$

where $e_i \triangleq \sum_{j=0}^N a'_{ij}(x_i - x_j)$, a'_{ij} is the (i, j) th entry of the adjacency matrix \mathcal{A}' associated with the communication graph \mathcal{G}_2 , and L_2 is the corresponding Laplacian matrix, σ_P and σ_I represent the control strengths of the proportional and integral actions respectively, γ is positive constants to be determined, \hat{W}_{i0} and \hat{W}_i are parameter estimates with respect to W_0 and W_i . The updated adaptive law of the estimates are

$$\dot{\hat{W}}_{i0} = - \sum_{j=0}^N a'_{ij}(x_i - x_j) \phi_0^T, \quad \dot{\hat{W}}_i = \sum_{j=0}^N a'_{ij}(x_i - x_j) \phi_i^T \tag{5}$$

The function $\theta(\cdot)$ is defined as

$$\theta(x) = \begin{cases} \frac{x}{\|x\|} & \text{if } \|x\| \neq 0 \\ 0 & \text{if } \|x\| = 0 \end{cases}$$

Assumption 3. The underlying graph of the communication network \mathcal{G}_2 is connected , and at least one follower has access to the leader.

It is assumed that the leader has no neighbors (i.e., does not receive information from any other agent). Under Assumption 1, the Laplacian matrix \mathcal{L}_2 associated with \mathcal{G}_2 can be partitioned as $\mathcal{L}_2 = \begin{pmatrix} 0 & 0_{1 \times N} \\ l & L_1 \end{pmatrix}$, where $L_1 \in R^{N \times N}$.

Before moving forward, define the consensus error $\delta_i = x_i - x_0$. Then, the error can be written as

$$\dot{\delta}_i = A_i \delta_i + (A_i - A_0)x_0 + b_i + W_i \phi_i + u_i - u_0 + f(x_i) - f(x_0) \quad (6)$$

Let $e = (e_1^T, \dots, e_N^T)^T$, $\delta = (\delta_1^T, \dots, \delta_N^T)^T$. Then, one has $e = (L_1 \otimes I_n)\delta$. It is clear that consensus is achieved if and only if the consensus error δ converges to zero. Substituting (3) and (4) into (6), it follows that

$$\begin{aligned} \dot{\delta} &= \hat{A}\delta + (\hat{A} - I_N \otimes A_0)(\mathbf{1} \otimes x_0) + F(x) - F(x_0) + L \otimes H - \sigma_P L_1 \otimes I_n \delta + \tilde{W}_0 \Phi_0 - \tilde{W} \Phi \\ &\quad + B - \sigma_I (D \otimes I_n) \Theta(e) - \gamma \operatorname{sgn}[(L_1 \otimes I_n)\delta] \end{aligned} \quad (7)$$

where $\hat{A} = \operatorname{diag}(A_1, \dots, A_N)$, $F(x_0) = (f^T(x_0), \dots, f^T(x_0))^T \in R^{nN}$, $F(x) = (f^T(x_1), \dots, f^T(x_N))^T$, $\tilde{W}_0 = \operatorname{diag}(\tilde{W}_{10}, \dots, \tilde{W}_{N0})$, $\tilde{W} = \operatorname{diag}(\tilde{W}_1, \dots, \tilde{W}_N)$ and $\tilde{W}_{j0} = \hat{W}_{j0} - W_0$, $\tilde{W}_j = \hat{W}_j - W_j$, $j = 1, \dots, N$, $\Phi_0 = (\phi_0^T, \dots, \phi_0^T)^T \in R^{Np}$, $\Phi = (\phi_1^T, \dots, \phi_N^T)^T$, $B = (b_1, \dots, b_N)^T$, $D = \operatorname{diag}(d_1, \dots, d_N)$, $\Theta = (\theta(e_1)^T, \dots, \theta(e_N)^T)^T$.

The following theorem provides a sufficient condition to achieve consensus of agents in (1).

Theorem 1. Suppose Assumptions 1, 2 and 3 hold. Then the leader-follower consensus problem of the N heterogeneous agents in (1) can be solved under the PI protocol (4) by choosing $\sigma_I > 0$, $\sigma_p > \frac{1}{\sigma_{\min}(L_1)} \left[\max_k \sigma_{\max}(A_k) + \frac{1}{2} (a + \frac{\eta^2}{a}) \right] + \sigma_{\max}(LL_1^{-1} \otimes H) + \frac{\varepsilon}{2}$, and $\gamma > \iota$, where $a, \varepsilon > 0$, and $\iota = \|x_0\|_\infty$.

Proof. Consider the following Lyapunov function:

$$V = \frac{1}{2} \delta^T (L_1 \otimes I_n) \delta + \frac{1}{2} \sum_{i=1}^N (\sigma_I d_i - \mu_i)^2 + \frac{1}{2} \sum_{i=1}^N \operatorname{tr}(\tilde{W}_{i0}^T \tilde{W}_{i0}) + \frac{1}{2} \sum_{i=1}^N \operatorname{tr}(\tilde{W}_i^T \tilde{W}_i) \quad (8)$$

The time derivative of V along the trajectory of (7) is given by

$$\begin{aligned} \dot{V} &= \delta^T (L_1 \otimes I_n) \dot{\delta} + \sum_{i=1}^N (\sigma_I d_i - \mu_i) \dot{d}_i + \sum_{i=1}^N \operatorname{tr}(\tilde{W}_{i0}^T \dot{\tilde{W}}_{i0}) + \sum_{i=1}^N \operatorname{tr}(\tilde{W}_i^T \dot{\tilde{W}}_i) \\ &= e^T \hat{A} \delta + e^T (\hat{A} - I_N \otimes A_0)(\mathbf{1} \otimes x_0) + e^T (F(x) - F(x_0)) + e^T L \otimes H x \\ &\quad - \sigma_P e^T (D \otimes I_n) \Theta(e) + e^T (\tilde{W}_0 \Phi_0 - \tilde{W} \Phi) - \gamma e^T \operatorname{sgn}[(L_1 \otimes I_n)\delta] + e^T B \\ &\quad - \sigma_I e^T (D \otimes I_n) \Theta(e) + \sum_{i=1}^N \operatorname{tr}(\tilde{W}_{i0}^T \dot{\tilde{W}}_{i0}) + \sum_{i=1}^N \operatorname{tr}(\tilde{W}_i^T \dot{\tilde{W}}_i) + \sum_{i=1}^N (\sigma_I d_i - \mu_i) \|e_i\| \end{aligned} \quad (9)$$

Since

$$e^T \hat{A} \delta = e^T \hat{A} (L_1 \otimes I_n)^{-1} e, \quad (10)$$

$$\begin{aligned} &e^T (\hat{A} - I_N \otimes A_0)(\mathbf{1} \otimes x_0) - \gamma e^T \operatorname{sgn}[(L_1 \otimes I_n)\delta] \\ &\leq \|e\|_1 \|(\hat{A} - I_N \otimes A_0)(\mathbf{1} \otimes x_0)\|_\infty + \gamma \|e\|_1 \leq (\iota - \gamma) \|e\|_1 \end{aligned} \quad (11)$$

where it is assumed that $\|\hat{A} - I_N \otimes A_0\|_\infty \leq 1$ without loss of generality, and $\iota = \|x_0\|_\infty \geq \|(\hat{A} - I_N \otimes A_0)(\mathbf{1} \otimes x_0)\|$ is the norm bound of x_0 .

Observe that

$$e^T L \otimes Hx = e^T (L \otimes H)[(L_1 \otimes I_n)^{-1} e + \mathbf{1} \otimes x_0] = e^T (LL_1^{-1} \otimes H)e, \quad (12)$$

$$\begin{aligned} e^T (\tilde{W}_0 \Phi_0 - \tilde{W} \Phi) &= \sum_{i=1}^N \left(\sum_{j=1}^N L_{ij} \delta_j^T \right) \tilde{W}_{i0} \phi_0 - \sum_{i=1}^N \left(\sum_{j=1}^N L_{ij} \delta_j^T \right) \tilde{W}_i \phi_i \\ &= \sum_{i=1}^N \text{tr}[\tilde{W}_{i0}^T e_i \phi_0^T] - \sum_{i=1}^N \text{tr}[\tilde{W}_i^T e_i \phi_0^T] \end{aligned} \quad (13)$$

and

$$\begin{aligned} -\sigma_I e^T (D \otimes I_n) \Theta(e) + e^T B &= \sum_{i=1}^N (-\sigma_I d_i e_i^T \theta(e_i) + e_i^T b_i) \\ &= \sum_{i=1}^N (-\sigma_I d_i \|e_i\| + e_i^T b_i) \leq \sum_{i=1}^N (-\sigma_I d_i + \mu_i) \|e_i\| \end{aligned} \quad (14)$$

Thus, with (10)–(14), it follows from (9) that

$$\begin{aligned} \dot{V} &\leq e^T \hat{A} (L_1 \otimes I_n)^{-1} e + e^T (LL_1^{-1} \otimes H)e - \sigma_P e^T e \\ &\quad + e^T (F(x) - F(x_{x_0})) - (\gamma - \iota) \|e\|_1 \end{aligned} \quad (15)$$

Let $U \in R^{N \times N}$ be a unitary matrix such that $U^T L_1 U = \Lambda = \text{diag}(\lambda_1, \dots, \lambda_N)$, where $0 < \lambda_1 \leq \dots \leq \lambda_N$ are the eigenvalues of L_1 . Let $\bar{\delta} = (\bar{\delta}_1^T, \dots, \bar{\delta}_N^T)^T = (U^T \otimes I_n) \delta$. According to the Lipschitz condition (2), it follows from (15) that

$$\begin{aligned} e^T (F(x) - F(x_0)) &= \delta^T (L_1 \otimes I_n) (F(x) - F(x_0)) \\ &= \bar{\delta}^T (U^T \otimes I_n) (L_1 \otimes I_n) (F(x) - F(x_0)) = \bar{\delta}^T (\Lambda U^T \otimes I_n) (F(x) - F(x_0)) \\ &= \bar{\delta}^T (\sqrt{a} \Lambda \sqrt{\Xi} \otimes I_n) \left(\frac{1}{\sqrt{a}} \sqrt{\Xi}^{-1} U^T \otimes I_n \right) (F(x) - F(x_0)) \\ &\leq \frac{1}{2} a \bar{\delta}^T (\Lambda \Xi \Lambda \otimes I_n) \bar{\delta}^T + \frac{1}{2a} [(U \otimes I_n) (F(x) - F(x_0))]^T \times (\Xi^{-1} \otimes I_n) \\ &\quad [(U \otimes I_n) (F(x) - F(x_0))] \leq \frac{1}{2} a \bar{\delta}^T (\Lambda \Xi \Lambda \otimes I_n) \bar{\delta}^T + \frac{\eta^2}{2a} \bar{\delta}^T (\Xi^{-1} \otimes I_n) \bar{\delta} \end{aligned} \quad (16)$$

where $\Xi > 0$ is a diagonal matrix and a is a positive scalar. By choosing $\Xi = \Lambda^{-1}$, it follows from (16) that

$$\begin{aligned} e^T (F(x) - F(x_0)) &\leq \frac{1}{2} a \bar{\delta}^T (\Lambda \otimes I_n) \bar{\delta}^T + \frac{\eta^2}{2a} \bar{\delta}^T (\Lambda \otimes I_n) \bar{\delta} \\ &= \frac{1}{2} \left(a + \frac{\eta^2}{a} \right) \bar{\delta}^T (\Lambda \otimes I_n) \bar{\delta} = \frac{1}{2} \left(a + \frac{\eta^2}{a} \right) \delta^T (U \otimes I_n) (\Lambda \otimes I_n) (U^T \otimes I_n) \delta \\ &= \frac{1}{2} \left(a + \frac{\eta^2}{a} \right) \delta^T (L_1 \otimes I_n) \delta = \frac{1}{2} \left(a + \frac{\eta^2}{a} \right) e^T (L_1 \otimes I_n)^{-1} e \end{aligned} \quad (17)$$

Substituting (17) into (15) and choose $\sigma_p > \frac{1}{\sigma_{\min}(L_1)} \left[\max_k \sigma_{\max}(A_k) + \frac{1}{2} \left(a + \frac{\eta^2}{a} \right) \right] + \sigma_{\max}(LL_1^{-1} \otimes H) + \frac{\varepsilon}{2}$, where ε is a small positive scalar, and choose $\gamma > \iota$ to yield

$$\begin{aligned} \dot{V} &\leq e^T (\hat{A}(L_1 \otimes I_n)^{-1}) e + e^T (LL_1^{-1} \otimes H) e - \sigma_P e^T e + \frac{1}{2} \left(a + \frac{\eta^2}{a} \right) e^T (L_1 \otimes I_n)^{-1} e - (\gamma - \iota) \|e\|_1 \\ &\leq \frac{\max_k \sigma_{\max}(A_k) + \frac{1}{2} \left(a + \frac{\eta^2}{a} \right)}{\sigma_{\min}(L_1)} e^T e + \sigma_{\max}(LL_1^{-1} \otimes H) e^T e - \sigma_P e^T e - (\gamma - \iota) \|e\|_1 \leq 0 \end{aligned} \quad (18)$$

Therefore, V is bounded, and so are the consensus error δ and the integral part d . From (8), one has that $\|\tilde{W}_0\|, \|\tilde{W}\| \in L_\infty$. Also, with the condition that Φ_0 and Φ are uniformly bounded, according to Lemma 2, it follows from (7) that $\dot{\delta} \in L_\infty$.

Since V is non-increasing and bounded from below by zero, it has a finite limit: $\lim_{t \rightarrow \infty} V(t) = V(\infty)$. Integrating the second inequality of (18), one can obtain that

$$V(0) - V(\infty) \geq \int_0^\infty \frac{\varepsilon}{2} e^T e dt$$

which implies $\delta \in L_2$, due to the boundedness of δ and $\dot{\delta}$. So, by Barbalat's lemma, $\delta \rightarrow 0$ as $t \rightarrow \infty$. Thus, the tracking error converges to zero. This completes the proof.

Remark 2. Theorem 1 shows that the distributed PI control protocol (4) can achieve consensus of N heterogeneous agents in (1), where the integral term plays a crucial role in eliminating the steady-state error. Unlike [17], in which the followers have identical dynamics, the linear multi-agent systems studied here can have heterogeneous dynamics.

3.2 Distributed Adaptive PI Control Design

In this section, we propose the following adaptive controller with time-varying coupling gains:

$$\begin{aligned} u_i &= -\sigma_{P_i} \sum_{j=0}^N a'_{ij} (x_i - x_j) - \sigma_I \int_0^t e_i(\tau) \theta(e_i(\tau)) d\tau + \hat{W}_{i0} \phi_0 - \hat{W}_i \phi_i + \gamma_i \operatorname{sgn} \left(\sum_{j=0}^N a'_{ij} (x_i - x_j) \right) \\ \sigma_{P_i} &= \kappa_i e_i^T e_i, \quad \gamma_i = \varrho_i \|e_i\|_1, \quad d_i = \|e_i\|, \quad i = 1, \dots, N \end{aligned} \quad (19)$$

where σ_{P_i} and γ_i are time-varying gains associated with the i th follower, κ_i , ϱ_i and σ_I are positive constants. d_i is the integral part. The proposed adaptive protocol is independent of any global information of the communication graph and thereby is fully distributed.

Consequently, the dynamics of the consensus error can be written as

$$\begin{aligned} \dot{\delta} &= \hat{A}\delta + (\hat{A} - I_N \otimes A_0)(\mathbf{1} \otimes x_0) + F(x) - F(x_0) + L \otimes H - \sigma_P L_1 \otimes I_n \delta + \hat{W}_0 \Phi_0 - \hat{W} \Phi \\ &\quad + B - \sigma_I (D \otimes I_n) \Theta(e) + \hat{\gamma} \operatorname{sgn}[(L_1 \otimes I_n)\delta] \end{aligned} \quad (20)$$

where $\sigma_P = \text{diag}(\sigma_{P_1}, \dots, \sigma_{P_N})$, $\hat{\gamma} = \text{diag}(\gamma_1, \dots, \gamma_N)$.

Theorem 2. Suppose Assumptions 1, 2 and 3 hold. Then, the leader-follower consensus problem of the N heterogeneous agents in (1) can be solved under the PI adaptive protocol (19). Moreover, the coupling gains σ_{P_i} , γ_i and the integral part d_i will converge to some finite steady-state values.

Proof. Consider the following Lyapunov function

$$\begin{aligned} V_1 = & \frac{1}{2} \delta^T (L_1 \otimes I_n) \delta + \frac{1}{2} \sum_{i=1}^N (\sigma_I d_i - \mu_i)^2 + \frac{1}{2} \sum_{i=1}^N \text{tr}(\tilde{W}_{i0}^T \tilde{W}_{i0}) + \frac{1}{2} \sum_{i=1}^N \text{tr}(\tilde{W}_i^T \tilde{W}_i) \\ & + \sum_{i=1}^N \frac{1}{2\kappa_i} (\sigma_{P_i} - \sigma_0)^2 + \sum_{i=1}^N \frac{1}{2\varrho_i} (\gamma_i - \gamma_0)^2 \end{aligned} \quad (21)$$

where σ_0 and ϱ_0 are positive constants.

The time derivative of V_1 along the trajectories of (20) is given by

$$\begin{aligned} \dot{V}_1 = & e^T \hat{A} \delta + e^T (\hat{A} - I_N \otimes A_0) (\mathbf{1} \otimes x_0) + e^T (F(x) - F(x_0)) \\ & + e^T L \otimes Hx - \delta^T (L_1 \hat{\sigma}_P L_1 \otimes I_n) \delta + e^T (\tilde{W}_0 \Phi_0 - \tilde{W} \Phi) \\ & + \delta^T (L_1 \hat{\gamma} \otimes I_n) \text{sgn}[(L_1 \otimes I_n) \delta] + e^T B - \sigma_I e^T (D \otimes I_n) \Theta(e) + \sum_{i=1}^N \text{tr}(\tilde{W}_{i0}^T \dot{\tilde{W}}_{i0}) \\ & + \sum_{i=1}^N \text{tr}(\tilde{W}_i^T \dot{\tilde{W}}_i) + \sum_{i=1}^N (\sigma_I d_i - \mu_i) \|e_i\| + \sum_{i=1}^N (\sigma_{P_i} - \sigma_0) e_i^T e_i + \sum_{i=1}^N (\gamma_i - \gamma_0) \|e_i\|_1 \end{aligned} \quad (22)$$

Note that

$$-\delta^T (L_1 \hat{\sigma}_P L_1 \otimes I_n) \delta + \sum_{i=1}^N (\sigma_{P_i} - \sigma_0) e_i^T e_i = -\delta^T (\sigma_0 L_1^2 \otimes I_n) \delta \quad (23)$$

and

$$\begin{aligned} & e^T (\hat{A} - I_N \otimes A_0) (\mathbf{1} \otimes x_0) + \sum_{i=1}^N (\gamma_i - \gamma_0) \|e_i\|_1 + \delta^T (L_1 \hat{\gamma} \otimes I_n) \text{sgn}[(L_1 \otimes I_n) \delta] \\ & = e^T (\hat{A} - I_N \otimes A_0) (\mathbf{1} \otimes x_0) - \gamma_0 \|e_i\|_1 \leq -(\gamma_0 - \iota) \|e\|_1 \end{aligned} \quad (24)$$

Substituting (12)–(14), (23)–(24) and (17) into (22), it follows that

$$\begin{aligned} \dot{V}_1 \leq & e^T \hat{A} (L_1 \otimes I_n)^{-1} e + e^T (L L_1^{-1} \otimes H) e - \sigma_0 e^T e \\ & + \frac{1}{2} \left(a + \frac{\eta^2}{a} \right) e^T (L_1 \otimes I_n)^{-1} e - (\gamma_0 - \iota) \|e\|_1 \end{aligned} \quad (25)$$

By selecting σ_0 and γ_0 large enough such that $\sigma_0 > \frac{1}{\sigma_{\min}(L_1)} \left[\max_k \sigma_{\max}(A_k) + \frac{1}{2} \left(a + \frac{\eta^2}{a} \right) \right] + \sigma_{\max}(L L_1^{-1} \otimes H)$ and $\gamma_0 > \iota$, one

obtains that $\dot{V}_1 \leq 0$. The following proof is similar to that for Theorem 1. Since the consensus error converges to zero, it follows from the dynamics of the integral states d_i and the adaptive coupling gains σ_{P_i} and γ_i in (19) that \dot{d}_i , $\dot{\sigma}_{P_i}$ and $\dot{\gamma}_i$ will converge to zero. In addition, they are bounded, therefore the integral states d_i and the adaptive gains γ_i and σ_{P_i} will converge to some finite steady state values.

Remark 3. Compared with the PI control protocol (4), which requires the eigenvalues of the Laplacian matrix be associated with the communication graph, the adaptive distributed protocol (19) proposed in this section only requires relative states of neighboring agents, and thus can be implemented in a fully distributed way.

4 Simulation

In this section, a simulation example is provided to illustrate the theoretical results. Consider a network of five followers and one leader. System matrices in (1) are chosen as $A_0 = \begin{bmatrix} -0.1 & 0.1 \\ -0.1 & -0.2 \end{bmatrix}$, which is obviously stable, $A_1 = A_4 = \begin{bmatrix} 0 & -0.5 \\ 0.1 & 0 \end{bmatrix}$ (oscillatory), $A_2 = A_5 = \begin{bmatrix} -0.3 & 0 \\ -0.2 & -0.5 \end{bmatrix}$ (stable), $A_3 = \begin{bmatrix} 0.3 & 0.4 \\ 0 & 0.5 \end{bmatrix}$ (unstable), $H = \begin{bmatrix} -2 & 3 \\ 4.5 & -1 \end{bmatrix}$. The disturbances b_i are time-varying, given as

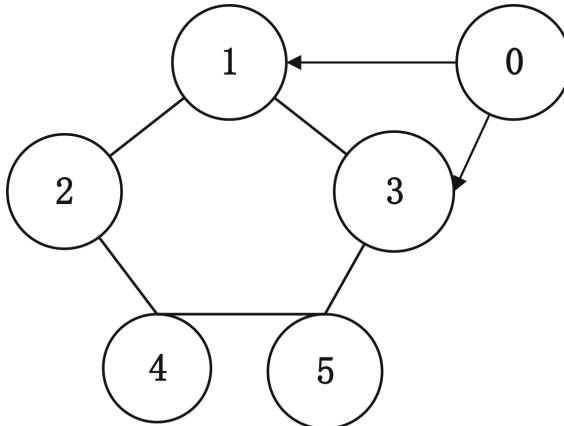


Fig. 1. Leader-follower communication topology

$b_1 = \begin{bmatrix} 0 \\ \sin(0.2t) \end{bmatrix}$, $b_2 = \begin{bmatrix} 2\cos(0.4*t) \\ 0 \end{bmatrix}$, $b_3 = \begin{bmatrix} e^{-t} \\ -2\sin(t) \end{bmatrix}$, $b_4 = \begin{bmatrix} 0 \\ \cos(0.5t) \end{bmatrix}$,
 $b_5 = \begin{bmatrix} 1 \\ 2e^{-0.5t} \end{bmatrix}$. The basis functions in the linearly parameterized term w_i are defined by $\phi_0 = [sint, cost]^T$, $\phi_i = [\sin((i+1)t), \cos((i+1)t)]^T$, $i = 1, \dots, N$. The nonlinear function f is chosen as $f(x_i, t) = [-\sin(x_{i,1}), -\sin(x_{i,2})]^T$. The topologies of L_1 and L_2 can be seen from Fig. 1.

Select $\sigma_I = 1$, $\varrho_i = 0.02$, $\kappa_i = 0.01$. The state trajectories of the agents under adaptive PI control (21) are depicted in Fig. 2, demonstrating that consensus is achieved. The integral action and the coupling gains are shown in Fig. 3, which are obviously bounded.

5 Conclusion

In this paper, we have addressed the consensus tracking problem of heterogeneous networks with heterogeneous uncertainties. Two novel distributed proportional-integral (PI) control protocols with dynamic and static coupling gains respectively are proposed for reaching consensus in the presence of disturbances, which thus are robust. Furthermore, an adaptive PI control protocol independent of the communication graph is proposed, which also is fully distributed.

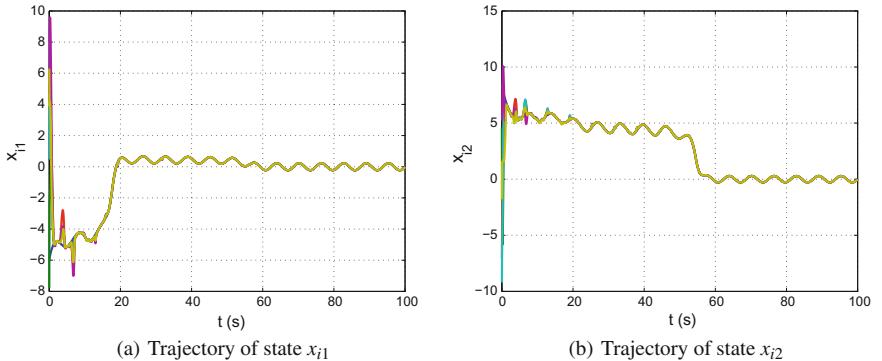
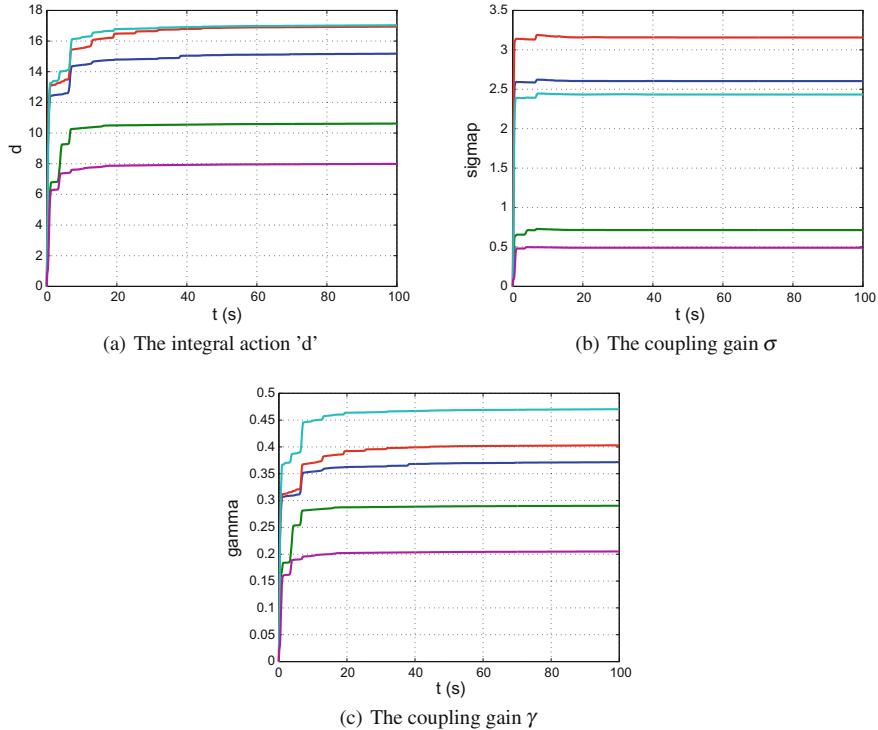


Fig. 2. Trajectory of the state

**Fig. 3.** The parameters

Acknowledgement. This work was supported by the National Key R&D Program of China under grant 2017YFB1301001 and National Natural Science Foundation of China under Grant U1713223, 11332001, 61673026, and the Hong Kong Research Grants Council under the GRF Grant CityU11200317.

References

1. Ren, W., Beard, R.W.: Distributed Consensus in Multi-vehicle Cooperative Control. Springer, New York (2008)
2. Olfati-Saber, R., Fax, A., Murray, R.: Consensus and cooperation in networked multi-agent systems. Proc. IEEE **95**(1), 215–233 (2007)
3. Sorrentino, F.: Synchronization of hypernetworks of coupled dynamical systems. New J. Phys. **14**(3), 033035 (2012)
4. Li, Z., Duan, Z., Chen, G., Huang, L.: Consensus of multiagent systems and synchronization of complex networks: a unified viewpoint. IEEE Trans. Circuits Syst. I, Regul. Pap. **57**(1), 213–224 (2010)
5. Ren, W., Beard, R.W.: Consensus algorithms for double-integrator dynamics. In: Theory and Applications, Distributed Consensus in Multi-vehicle Cooperative Control, pp. 77–104 (2008)
6. Yu, W., Chen, G., Cao, M.: Some necessary and sufficient conditions for second-order consensus in multi-agent dynamical systems. Automatica **46**(6), 1089–1095 (2010)

7. Scardovi L., Sepulchre R.: Synchronization in networks of identical linear systems. In: Decision and Control, 47th IEEE Conference, IEEE, pp. 546–551 (2008)
8. Yu, W., DeLellis, P., Chen, G., et al.: Distributed adaptive control of synchronization in complex networks. *IEEE Trans. Autom. Control* **57**(8), 2153–2158 (2012)
9. Zhao, Y., Liu, Y., Li, Z., et al.: Distributed average tracking for multiple signals generated by linear dynamical systems: an edge-based framework. *Automatica* **75**, 158–166 (2017)
10. Mei, J., Ren, W., Chen, J.: Distributed consensus of second-order multi-agent systems with heterogeneous unknown inertias and control gains under a directed graph. *IEEE Trans. Autom. Control* **61**(8), 2019–2034 (2016)
11. Lombana, D.A.B., Di Bernardo, M.: Distributed PID control for consensus of homogeneous and heterogeneous networks. *IEEE Trans. Control Netw. Syst.* **2**(2), 154–163 (2015)
12. He, W., Qian, F., Lam, J., et al.: Quasi-synchronization of heterogeneous dynamic networks via distributed impulsive control: error estimation, optimization and design. *Automatica* **62**, 249–262 (2015)
13. Duan, Z., Wang, J., Chen, G.: Stability analysis and decentralized control of a class of complex dynamical networks. *Automatica* **44**, 1028–1035 (2008)
14. Razeghi-Jahromi, M., Seyed, A.: Stabilization of networked control systems with sparse observer-controller networks. *IEEE Trans. Autom. Control* **60**(6), 1686–1691 (2015)
15. Hou, Z., Cheng, L., Tan, M.: Decentralized robust adaptive control for multiagent system consensus problem. *IEEE Trans. Syst. Man, Cybern. Syst.* **39**(3), 636–647 (2009)
16. Das, A., Lewis, F.L.: Cooperative adaptive control for synchronization of second-order systems with unknown nonlinearities. *Int. J. Robust Nonlinear Control* **21**(13), 1509–1524 (2011)
17. Sun, J.Y., Geng, Z.Y.: Adaptive consensus tracking for linear multi-agent systems with heterogeneous unknown nonlinear dynamics. *Int. J. Robust Nonlinear Control* **26**(1), 154–173 (2015)
18. Ai, X., Yu, J., Jia, Z., et al.: Adaptive robust consensus tracking for nonlinear second-order multi-agent systems with heterogeneous uncertainties. *Int. J. Robust Nonlinear Control* **27**(18), 5082–5096 (2017)
19. Wieland, P., Sepulchre, R., Allgower, F.: An internal model principle is necessary and sufficient for linear output synchronization. *Automatica* **47**(5), 1068–1074 (2011)
20. Lombana, D.A.B., Di Bernardo, M.: Multiplex PI control for consensus in networks of heterogeneous linear agents. *Automatica* **67**, 310–320 (2016)
21. Lv, Y., Li, Z., Duan, Z.: Fully distributed adaptive PI controllers for heterogeneous linear networks. *IEEE Trans. Circuits Syst. II, Exp. Briefs.* <https://doi.org/10.1109/TCSII.2017.2783893>
22. Lv, Y., Li, Z., Duan, Z.: Distributed PI control for consensus of heterogeneous multiagent systems over directed graphs. *IEEE Trans. Syst. Man, Cybern. Syst.* **vol. 99** (2018)
23. Zhang, H., Lewis, F.L., Qu, Z.: Optimal design for synchronization of cooperative systems: state feedback, observer and output feedback. *IEEE Trans. Autom. Control* **56**(8), 1948–1952 (2011)
24. Feng, Y., Duan, Z., Ren, W., et al.: Consensus of multi-agent systems with fixed inner connections. *Int. J. Robust Nonlinear Control* **28**(1), 154–173 (2018)



Time Granularity in System-of-Systems Simulation of Infrastructure Networks

Mateusz Iwo Dubaniowski^(✉) and Hans R. Heinimann

Singapore-ETH Centre, Future Resilient Systems, 1 CREATE Way, #06-01
CREATE Tower, Singapore 138602, Singapore
iwo.dubaniowski@frs.ethz.ch

Abstract. Because of their extreme complexities, a system-of-systems (SoS) approach is often used for simulating infrastructure systems. This allows the user to integrate models of various systems into one simulation. However, this integration presents several issues because individual simulations are often designed for only a specific purpose and context. This leads to variations among space granularities and proposes a challenge when selecting an appropriate time granularity for the overall SoS simulation. To explore how this granularity might affect the outcome of simulations, we designed and developed a prototype system of three infrastructure simulation networks that were then combined into one SoS simulation using High Level Architecture (HLA) implementation. We then performed a series of experiments to investigate the response of the simulation to varying time granularities. Our examination included a propagation of disruptions among constituent simulations to estimate how this was affected by the frequency of updates between those simulations, i.e. time granularity. Our results revealed that the size of the simulated disruption decreased with increasing time granularity and that the simulated recovery time was also affected. In conducting this project, we also identified several ideas for future research that focus on a wider range of disruption generators and infrastructure systems in those SoS simulations.

Keywords: Time granularity · Infrastructure system · System-of-systems (SoS) · High level architecture (HLA) · Interdependency study
Synchronization

1 Introduction

As technology advances, infrastructure systems are becoming more interdependent, requiring inputs and outputs from and to other systems so that their functions are performed properly. At the same time, disruptions to these systems are increasing both in frequency and in the extent of the impact. This is especially visible within the

ETH Zurich, Future Resilient Systems at the Singapore-ETH Centre (SEC), which was established collaboratively between ETH Zurich and National Research Foundation (NRF) Singapore (FI 370074011), under the auspices of the NRF's Campus for Research Excellence and Technological Enterprise (CREATE) programme.

context of urban settings, where various intertwined systems must work perfectly to ensure smooth operation of systems nearby [1]. Consequently, designing these systems to be highly resilient and ensuring that they respond adequately to any disruptions are major concerns that must be addressed. Planners and managers have to recognize how disruptions that emerge in one system can affect other systems. Currently, however, there is a lack of understanding about how time granularity in distributed modeling environments affects the outcomes of simulation experiments, including the propagation of disruptions between constituent systems.

Although models are being developed to examine interdependencies among infrastructure systems [2–5], they have not addressed the issue of time granularity. Their applications include models of traffic simulation [6] or networks of infrastructure systems [7]. However, those models are not designed to be accurate when investigating the impact of disruptions on infrastructure. There exists a model simulating interactions between infrastructure systems under disruption [8]. However, in this model the individual infrastructure systems are not run independently of each other, and hence time granularity of synchronizing the constituent infrastructure systems simulations is not analyzed.

The objective of our study was to conduct prototype experiments that assessed the impact of time granularity on the propagation of disruptions between systems in a system-of-systems (SoS) simulation. Specifically, we examined whether simulating the same event among the same infrastructure systems could produce widely differing results under various time granularities. We limited our experiments to one abstract set of infrastructure system networks and a single event rather than exploring a wider range of events. Therefore, our test results would not necessarily apply to any particular real-life system.

2 Model Development

Frameworks and methodologies have been established to model individual infrastructure systems. Such infrastructure systems include power supply grids, transportation networks, water supply networks, emergency services, financial systems etc. The SoS approach can be used to illustrate interactions among interdependent systems [7–9]. In such models, the constituent systems operate on their own, being guided by their unique mechanics, but can be combined to exchange certain information (data) based on their interdependencies. In this approach, autonomous infrastructure systems interact with each other, users, operators, observers, and disruption generators. All of these components can be represented as individual systems within the overarching SoS simulation. Synchronization of these systems means that interdependencies between them are encoded in the simulation design.

One framework used for modeling SoS is High Level Architecture (HLA) [10, 11], which originated in military applications. However, HLA can also be used with civil infrastructure systems [7]. This framework has three components. First, the Interface Specification determines how and where constituent systems within HLA, so-called ‘federates’, communicate with the real-time infrastructure (RTI). The second component, the Object Model Template (OMT), defines what information is exchanged

between constituent simulations. As the third component, Rules specify what the federates must obey if they are to comply with the HLA overarching simulation, or ‘federation’.

Within the context of SoS simulations, federates can be infrastructure systems, operators, observers, disruption generators, or patterns of user services. These independently operating federates are connected with the RTI to exchange data and form an HLA federation. Such a system is shown in Fig. 1. In general, this framework can work with any simulation methodology that includes exchange of information with each other. However, in the context of our model, the simulation programs are network simulations that correspond to infrastructure systems networks. These networks can be adaptive to the variable nature of infrastructure systems that they represent.

Although the HLA is useful for depicting the scope and method by which information is exchanged between constituent systems, it does not solve the issue of finding an adequate time granularity for the SoS. Time granularity defines when the exchange of information and, thereby, inter-system synchronization happens. Specifications for HLA include time management [12, 13], which ensures that timing between federates is synchronized. However, the issue of how often to synchronize constituent federate simulations remains unanswered because it can vary between different types of simulation.

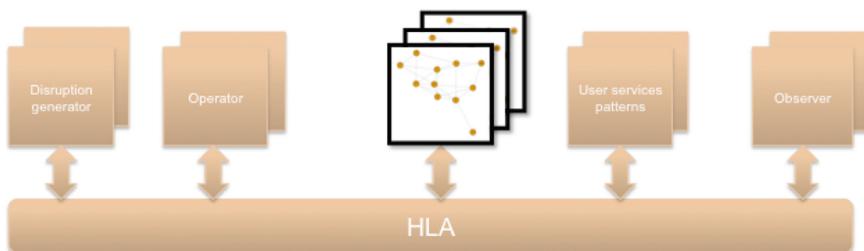


Fig. 1. HLA system-of-systems, showing components connected to real-time infrastructure to ensure exchange of information.

Time granularity is crucial when disruptions of a particular size and at a particular time are introduced by a generator into the simulation. These entry points and sizes, as well as the effects of such disruptions, can vary greatly, depending upon the time granularity of the simulation. We developed a prototype SoS of infrastructure system networks that could operate both independently and interdependently with each other. The prototype utilized an observer that could visualize the results and progress of the simulations, along with evaluating system performance. In this study, we also included a disruption generator and varied time granularity to learn how different granularities might influence the outcomes when the rest of the parameters of the simulation were held constant. These collected data included the size of the disruption and the length of recovery time.

In each network, nodes correspond to units that perform operations and interact with other nodes in the same network as well as with corresponding nodes in other

federates. Edges correspond to transfer links between operational units within each network. The mechanics of each network differ and are abstract. Although each node has its own intrinsic, internal performance, it also takes inputs from the incoming edges of its network and from its corresponding nodes in the other two networks. These conditions are then combined and transformed to determine the total performance of that particular node. Performance is then propagated to the next nodes through outgoing edges. In doing so, we designed a working process for each constituent federate to simulate an individual infrastructure system network.

We also connected a disruption generator to the HLA RTI, following Poisson processes to introduce disruptions into one of the constituent networks by communicating the message that certain nodes had been affected and were to be incapacitated. This reduced to ‘zero’ the intrinsic performance of the affected nodes. The framework components were linked with the HLA RTI to allow for the exchange of information at synchronized points. After a certain time period, recovery occurred, the disruption was removed, and performance by those nodes was restored to its original level.

2.1 Model Implementation

Our simulation was developed in C++ and Python, and the HLA modules were conceived in C++ v11 [14] through Portico 2.0.2 HLA implementation to define the interfaces between infrastructure systems, communication between them at various time granularities, and communication with the disruption generator. For graph operations, we used igraph library version 0.7.1 [15]. The disruption generator was also developed in C++ v11. Infrastructure systems were created in Python 3.5 [16], under an Anaconda 2.4.0 distribution [16], with the aid of the igraph library for graph operations and representation, and also with numpy library version 1.10.1 [17] for linear algebra and numerical operations. All results were observed with an interface web page developed in JavaScript and HTML, using the CanvasJS [18] library to visualize the performance of the simulation. This SoS simulation was evaluated and run on a Mac OS Yosemite 10.10.5 operating system.

Implementation of the system followed a natural, logical pattern. First, the infrastructure system networks and their required inputs and outputs were developed in parallel with the disruption generator and observer. Afterward, interfaces with the HLA RTI were created for each of the components.

The system was evaluated and tested with several sets of parameters and different constituent networks under varying time granularities. It performed well and was able to mimic certain real-life scenarios. We then conducted multiple tests with positive outcomes to ensure that the system propagated and communicated disruptions as expected.

3 Simulation Experiments

We measured the impact of time granularity by running a small-scale simulation of three interdependent infrastructure system networks (Fig. 2). Each network consisted of a certain number of nodes connected with edges. These nodes had equivalents in

other networks with which they communicated by exchanging information at synchronization points. The experimental networks had the following parameters: (1) Network 1, 20 nodes connected with 75 edges; (2) Network 2, 21 nodes connected with 77 edges; and (3) Network 3, 22 nodes connected with 77 edges. They corresponded to real-life infrastructure systems such as a power grid, water supply, or transportation. The interdependencies of those networks were defined according to how each utilized the information exchanged among them.

In our simulation, a Poisson generator was used to introduce disruption into the system. The process of disruption and recovery, and the current state of the system under simulation, was examined in real-time by an observer connected to the system.

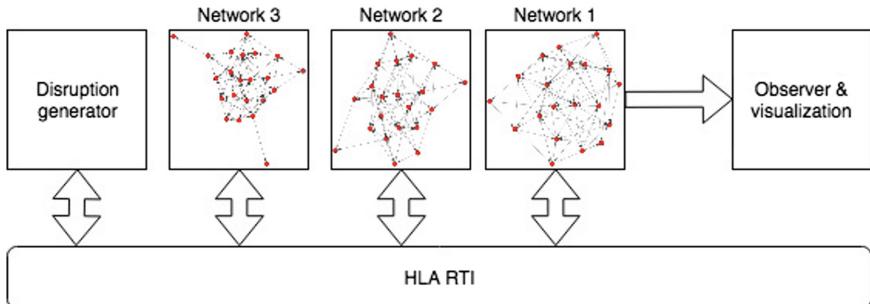


Fig. 2. Setup of experimental system involving 3 network simulations and disruption generator connected through HLA RTI. Impact of disruption on Network 3 was simulated. Propagation and related impacts were evaluated by observer and visualization module connected to Network 1.

3.1 Experimental Layout

Our goal was to determine whether time granularity has an impact on simulation outcomes. To achieve this, we kept all of the simulation parameters constant (single-factor experiment) and varied only the granularity factor across 12 levels (Table 1). Recovery times were measured for the different granularities along with the maximum disruption.

Table 1. Single-factor layout.

	Timesteps											
Time granularity	1	2	3	6	9	12	15	18	21	24	27	30
Recovery time	18											

Various experimental scenarios were tested with the following parameters. Network 3 was disrupted, Network 1 was measured, and Network 2 was not affected. In Network 3, disruption was of size 16, corresponding to 16 nodes being incapacitated when the disruption was introduced. The recovery time was 18, meaning that the disruption was recovered after 18 time steps. That is, the affected nodes in Network 3 were

restored to their normal performance after 18 steps. This system was swept through time granularities from 1 to 30, in increments of 3, but also included time granularities of 1 and 2. We measured the disruption size of Network 1 and measured its recovery time to within >99% of the original performance.

3.2 Simulation Result

The results from this prototype simulation confirmed our expectations that changing the time granularity would have an impact on the outcome. After assessing the performance of Network 1, it was apparent that both disruption size and the recovery process were affected by granularity.

Disruption size. The main metric used to assess simulation performance was simulated disruption size, which was measured in Network 1 at each level of time granularity. Here, disruption size decreased as granularity increased (Fig. 3). After initially decreasing rather slowly, when time granularity exceeds the actual recovery time, we learned that the disruption was not captured at all because its size fell to '0'.

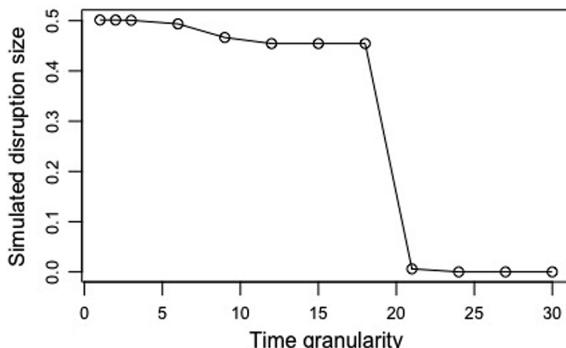


Fig. 3. Relationship between disruption size and time granularity. As granularity increased, disruption size decreased. When time granularity exceeded recovery time (18), disruption was no longer visible in simulation.

Recovery time. Simulated recovery time was expressed in time steps, beginning when the disruption was introduced in Network 1 and ending when 99% performance level was again reached. This metric allowed us to see which simulated recovery time was the closest to the actual recovery time, i.e., 18. Figure 4 illustrates how the simulated recovery time continued to increase until it arrived at half of the actual recovery time. At that point, the simulated recovery time declined before rising again, at a slower but constant rate, until the actual recovery time was reached. At a point higher than the actual recovery time, the simulated recovery time became '0' because the system no longer recognized the disruption. That is, Network 3 was repaired before the disruption could propagate to other networks where it would have been registered. This scenario presented a potentially large issue when defining an adequate time granularity in a simulation.

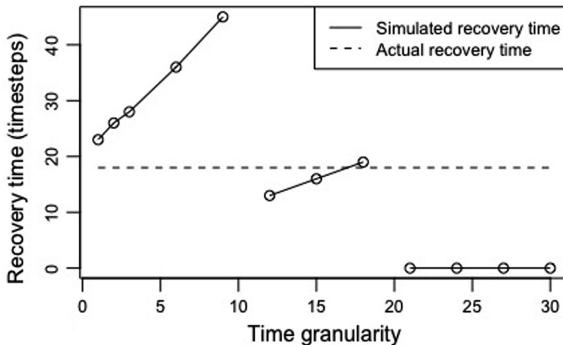


Fig. 4. Relationship between simulated recovery time and time granularity, based on time required for performance to return to 99% of original level when compared with actual recovery time. Three simulated regimes were revealed: up to half of actual recovery time (1–9), from half to actual recovery time (9–18), and above actual recovery time (18–30). The second regime proved to be most representative of actual recovery time.

4 Conclusions

Our study objective was to run simulation experiments with three federates and a disruption generator that would mimic the interdependencies among several infrastructure systems. We then investigated how time granularity might affect the outcome of those simulations.

These experiments yielded the following major results:

- As time granularity increased, the simulated disruption size was decreased.
- As time granularity increased, the simulated recovery time increased at different rates within three distinct granularity regimes: up to half of the actual recovery time, from half up to the actual recovery time, and above the actual recovery time.
- When time granularity exceeded the actual recovery time, the disruptions were no longer registered in the system.
- Time granularity has to be smaller than the actual recovery time of key events in order for the propagation of those events to be registered and visible in a SoS simulation.

Our analysis of how time granularity influences the simulation of infrastructure systems is novel. Although time management of SoS simulations has already been investigated within the context of HLA simulations [12, 19], no one had previously attempted to examine time granularity within the context of modeling interdependent infrastructure systems. Although Eusgeld and Nan [3], Rinaldi [20], and Dubaniowski and Heinemann [8] had studied interdependent infrastructure systems, their research did not consider the impact of time granularity on such modeling. In the model described by Dubaniowski and Heinemann, individual infrastructure systems are combined together in one, large multi-layer network simulation, rather than multiple individual independent, distributed simulations connected together to exchange information. Thus time granularity of synchronization cannot be investigated adequately in their model.

Similarly, HLA has been utilized in modeling various SoS simulations, e.g., aircraft [21], defense [9, 22], or individual infrastructure [23]. However, none of the earlier research had introduced the concept of disruptions in those evaluations. Therefore, the results of our innovative experiments will assist scientists in developing better models of infrastructure and investigating how disruptions can affect the interdependencies of infrastructure systems.

One limitation, revealed in our tests, was the incomplete exploration of the experimental space because we varied only time granularity without considering other factors. In addition, the nature of our included networks was abstract and generally would vary with the location and type of infrastructure. To address these challenges, future studies could involve multi-factorial experiments with a greater number of variables. Moreover, real-life networks could be applied to evaluate the system within the context of a real-life SoS. Finally, one could incorporate a wider range of disruption types and generators, as well as more system networks.

Acknowledgements. This work is an outcome of the Future Resilient Systems project at the Singapore-ETH Centre (SEC), which is funded by the National Research Foundation of Singapore (NRF) under its Campus for Research Excellence and Technological Enterprise (CREATE) programme.

References

1. Heinemann, H.R., Hatfield, K.: Infrastructure Resilience Assessment, Management and Governance—State and Perspectives in Resilience and Risk, pp. 147–187. Springer, Dordrecht (2017)
2. Boer, C.: Distributed Simulation in Industry. Erasmus University, Rotterdam (2005)
3. Eusgeld, I., Nan, C.: Creating a simulation environment for critical infrastructure interdependencies study. In: 2009 IEEE International Conference on Industrial Engineering and Engineering Management, pp. 2104–2108 (2009)
4. Rinaldi, S.M., Peerenboom, J.P., Kelly, T.K.: Identifying, understanding, and analyzing critical infrastructure interdependencies. *IEEE Contl. Syst. Mag.* **21**(6), 11–25 (2001)
5. Ouyang, M.: Review on modeling and simulation of interdependent critical infrastructure systems. *Reliab. Eng. Syst. Saf.* **121**, 43–60 (2014)
6. Kelin, U., Schulze, T., Straßburger, S.: Traffic simulation based on the high level architecture. In: Proceedings of the 30th Winter Simulation Conference (WSC), Los Alamitos, CA, USA, pp. 1095–1104 (1998)
7. Eusgeld, I., Nan, C., Dietz, S.: ‘System-of-systems’ approach for interdependent critical infrastructures. *Reliab. Eng. Syst. Saf.* **96**(6), 679–686 (2011)
8. Dubaniowski, M.I., Heinemann, H.R.: A framework modeling flows of goods and services between businesses, households, and infrastructure systems. In: Resilience The 2nd International Workshop on Modelling of Physical, Economic and Social Systems for Resilience Assessment , 2017, vol. I, pp. 182–190, Ispra, Italy, 14–16 Dec 2017
9. Gao, J., Buldyrev, S.V., Stanley, H.E., Havlin, S.: Networks formed from interdependent networks. *Nat. Phys.* **8**(1), 40–48 (2012)
10. Dahmann, J.S., Fujimoto, R.M., Weatherly, R.M.: The department of defense high level architecture. In: Proceedings of the 29th Conference on Winter Simulation, pp. 142–149, Atlanta, GA, USA (1997)

11. Topçu, O., Oğuztüzün, H.: High level architecture. In: Topçu, O., Oğuztüzün, H. (eds.) Guide to Distributed Simulation with HLA, pp. 29–78. Springer International Publishing, Cham (2017)
12. Fujimoto, R.M.: Time management in The high level architecture. *Simulation* **71**(6), 388–400 (1998)
13. Fujimoto, R.M., Weatherly, R.M.: Time management in the DoD high level architecture. In: Proceedings of the Tenth Workshop on Parallel and Distributed Simulation, pp. 60–67, Washington, DC, USA (1996)
14. Josuttis, N.M.: The C++ standard library: a tutorial and reference. Addison-Wesley Professional (2012)
15. igraph.org.: python-igraph 0.7.1. 04-Feb-2014. <http://igraph.org/2014/02/04/igraph-0.7-python.html>. Accessed 07 Aug 2018
16. Python.org.: Python Release Python 3.5.0, Python.org, 13-Sep-2015. <https://www.python.org/downloads/release/python-350/>. Accessed 07-Aug-2018
17. SciPy.org.: NumPy Reference—NumPy v1.10 Manual, 18-Oct-2015. <https://docs.scipy.org/doc/numpy-1.10.1/reference/>. Accessed 07 Aug 2018
18. Fenopix, Inc.: CanvasJS 1.8.5 beta, CanvasJS. <https://canvasjs.com/> (2013). Accessed 07 Aug 2018
19. Jansen REJ, Huiskamp W, Boomgaardt JJ, Brasse M.: Real-time scheduling of HLA simulator components. TNO Fysisch en Elektronisch Laboratorium, Jan 2004
20. Rinaldi, S.M.: Modeling and simulating critical infrastructures and their interdependencies. In: Proceedings of the 37th Annual Hawaii International Conference on System Sciences, 2004, Big Island, HI, USA (2004)
21. Gervais, C., Chaudron, J.B., Siron, P., Leconte, R., Saussie, D.: Real-time distributed aircraft simulation Through HLA. In: Proceedings of the 2012 IEEE/ACM 16th International Symposium on Distributed Simulation and Real Time Applications, pp. 251–254, Washington, DC, USA (2012)
22. Ferenci, S.L., Choi, M., Evans, J., Fujimoto, R.M., Alspaugh, C., Legaspi, A.K.: Experiences integrating NETWARS with the naval simulation system using the high level architecture. In: IEEE MILCOM 2004. Military Communications Conference, vol. 3, pp. 1395–1401 (2004)
23. Jain, A., Robinson, D., Dilkina, B., Fujimoto, R.: An approach to integrate inter-dependent simulations using HLA with applications to sustainable urban development. In: 2016 Winter Simulation Conference (WSC), pp. 1218–1229, Washington, DC, USA (2016)

TTPROF: A Weighted Threshold Model for Studying Opinion Dynamics in Directed Temporal Network

Eeti Jain^{1(✉)}, Anurag Singh¹, and Rajesh Sharma²

¹ Department of Computer Science and Engineering, National Institute of Technology Delhi, New Delhi 110040, India

{eetijain,anuragsg}@nitdelhi.ac.in

² Institute of Computer Science, University of Tartu, Tartu, Estonia
rajesh.sharma@ut.ee

Abstract. Information flows continuously in a network of individuals, which are connected with each other as individuals tend to learn from each other through sharing of views or information. For example, an individual's perception or ratings or reviews of a product might change with time with an effect of the opinions (representing new information) of its acquaintances. Various models have been proposed in the past to study opinion dynamics in complex settings. However, most of these models considered networks as static. In this work, we propose a directed and weighted Temporal Threshold Page Rank Opinion Formation (TTPROF) model, for studying the opinion dynamics using temporal networks. The term dynamics in this work is related to two aspects, firstly, dynamics of the network i.e., the network structure is time varying (temporal) in nature. Secondly, dynamics on the network i.e., opinions of individuals propagating on the network change with time. Opinion of every node is weighted by a factor of its page rank. A node is affected with its neighbour's weighted opinion. A concept of threshold is added to limit the neighbours from which opinion can be shared. A parameter of the fraction of top page ranked nodes is introduced to consider the opinions of influential nodes irrespective to threshold. We simulated our model using random networks with temporal effect, which shows that as the threshold value or the fraction of top ranked nodes increases, opinions start converging faster and consensus is achieved sooner. But if any of these parameters is decreased, convergence time increases or opinions converge into multiple clusters.

Keywords: Temporal network · Top ranked nodes
Opinion dynamics · Threshold value · Page rank · Timestamp

1 Introduction

Information diffusion is an omnipresent phenomena. Sometimes it happens intentionally and sometimes unconsciously when we are exposed to different views or

opinions (or information) through social contacts and media. A decade back, mass media, e.g., television, newspaper, radio, magazines etc., were the popular medium for spreading the information among the people. However, with the advancement of web, social platforms have become popular medium for spreading the information and forming the opinions among the people due to ease of use and accessibility. People share their information and exchange their opinions in their day to day life through various offline and online interactions. For example, people may discuss about any brand to frame the opinion related to it, spreading of fake news related to any issue or product on social platforms can also affect the opinions of the people [1]. Sometimes, politicians start affecting the opinions of the voters by bringing influential people in their political organization with an aim to affect the opinions of the people so that they cast their votes for them [2]. Most of the research on opinion dynamics has been done exploiting static complex network [3,4], where people are represented as nodes and interactions among them as edges. By static, we mean that the structure of the network remains the same, that is, the number of edges as well their arrangement remains the same. However, the static network settings are in contrast with the real world scenario, which are temporal and dynamic in nature. These two terms are compliment in nature as with time, nodes and edges may appear or disappear at different time instances, which signify the dynamic nature of networks.

A static network is a collection of nodes and edges, $G(V, E)$ where, nodes V are connected through edges E . A temporal network can be represented by $G(V, E, t)$ where V and E are the set of nodes and edges at time stamp t_i , where $t \in [t_i : i \in \{1, 2, \dots, max\}]$, where t_{max} is the total observation period. If a network restructures at a time stamp t , opinions of individuals also get reframed signifying that individuals have come in contact with new individuals. In addition, opinions of individuals are continuous and are not necessarily binary (0 or 1) in nature. For example, if some individuals are having opinions related to a product, only 0 (thinking extreme negative) and 1 (thinking extreme positive) are not the only feasible opinions. There can be other options as well like thinking little worst or average or little better which relates to continuous opinion.

Many models have been proposed for understanding the dynamics on network which specify how opinions can be framed among the individuals in a static network [4]. PROF (Page Rank Opinion Formation) model [5] was proposed by Kandiah and Shepelyansky where weighted binary opinions are shared among the individuals by considering that not all neighbours have the same influence on their neighbours. To incorporate this phenomena, they used the page rank algorithm to rank the neighbors and weight the opinions. However, they did not include the effect of temporal behaviour, threshold and continuous opinion in their model. Deffuant et al. [3] have worked on threshold parameter not on page rank during opinion sharing which will be further explained in related work. They consider only consensus(1 cluster) or polarization (2 clusters). They have worked on static graphs but real time network evolves with time. Many researchers

[6–8] have worked on opinion dynamics on evolving network with fixed number of nodes, only interaction among them is changing. They do not consider the page rank factor as discussed in static graph.

In this paper, we propose a directed and weighted Temporal Threshold Page Rank Opinion Formation (TTPROF) model which is a major improvement compared to previous works as this model considers dynamicity and temporal aspects of the network. The model can also work on directed networks. The concept of threshold on difference in opinion values of adjacent nodes is introduced in the proposed model along with the sharing of weighted opinions. It suggests that opinions are shared only from those nodes whose opinion is close to the individual's opinion in a weighted manner. A parameter, top ranked nodes is considered in this work which is calculated using page rank. Opinions of these nodes are always considered during sharing of opinions without checking the threshold value as these are the most important nodes of a network, taking our model one more step closer to reality. In our model, the opinions may converge to more than 2 values signifying fragmentation (multiple clusters of opinions) over polarization (2 clusters of opinions). In the results, even if the value of threshold is fixed, if the fraction of top ranked nodes changes, convergence of opinions also changes which is not found in the previous works [6–8]. In particular, the proposed model is domain independent and can be applied for studying opinion dynamics in various fields such as marketing, politics etc. We evaluated our model using temporal random network datasets with temporal nature of opinions.

The rest of the paper is organized as follows. In Sect. 2 current state of the art is discussed, in particular, we discuss various models of opinion sharing. In Sect. 3, proposed model related to the work is discussed along with the algorithms. In Sect. 4, results have been shown for different size of network at different time stamps. It is also shown that how the opinion is getting converged or split into different clusters. Finally, we conclude the paper in Sect. 5, where we also discuss some future research directions.

2 Related Work

Interactions among agents play an important role in the formation of network structure for information, technological and social phenomena propagation. Online social platforms are playing an important role in increasing the interaction among individuals to a large extent on a global level. The interactions in social network can play a vital role for framing opinions among the individuals. Individuals try to take the advice from their acquaintances which influence their opinions accordingly. Opinion estimation from the acquaintances of an individual has more tendency to get near the true value in comparison with the individual opinion itself. As an individual can think only in positive or negative direction, taking opinion from different people of different thinking, individual can reach to the more optimized opinion. This phenomenon is termed as wisdom of crowd

effect [9]. The interaction among individuals is a way of shaping the opinions about a product, brand or a topic.

There can be various questions that can arise, how the interaction leads to consensus/polarization or how the individuals should interact to share their opinions in the most optimal way which are answered by various researches. Rigorous work has been done in framing different models for opinion sharing which include the following features, biased conformity, compromise, and stubbornness [10]. In conformity models, individuals adopt the opinion of their neighbours without considering their own opinions. Clifford and Sudbury [11] proposed the discrete conformity opinion model called voting model in which individual adopts the opinion of individual from its neighbours, chosen uniformly at randomly. This model always leads to consensus which is not a practical situation as it is not necessary that all the nodes reach to the same opinion. There can be different clusters of opinions. A variant of voter model was proposed known as LPA (Label Propagation Algorithm) [12] in which individual updates the opinion with the majority opinions among its neighbours. In this model, polarization is achieved instead of consensus. Polarization means to reach to two set of clusters of opinions who disagree with each other in their opinions.

Instead of taking opinion from a neighbour directly, opinions of the neighbours can be averaged which leads to the averaging models. Degroot has given an averaging and compromise model [13] in which user adopts an opinion which is the average of its own opinion as well as the average of the opinions of its neighbours. This model leads to the consensus. There can be biased conformity in opinion formation which means to assign more weight to the neighbour with more similar opinions leading to the flocking behavior [14]. Considering the flocking model along with the averaging model, Hegselman and Krause [4] proposed a model in which a confidence region for an individual is find out which is the set of neighbours whose difference in opinions with the individual is in the limit of bounded threshold. Individual updates its opinion with the average value of neighbour's opinions in the confidence region, known as bounded confidence of that individual. Suppose an individual i has opinion y_i . Considering a fixed threshold value, μ , the confidence region of the individual k is denoted by the set: $S_i(y) = \{k : |y_k - y_i| \leq \mu\}$. Deffuant et al. [3] proposed a model for opinion sharing in which every pair of individuals randomly interact in a network. If two individuals have the opinions x and y such that $|x - y| < d$ (threshold), they readjust the opinions.

To include the effect of importance of nodes/ page rank, PROF model was proposed [5] which uses the page rank as the weighted factor to give some of the nodes more weightage than the others with less page rank. Directed network is considered here. But they do not investigate about how opinion is getting framed that is converging or getting polarized. They do not consider the threshold value, continuous opinion.

All the work discussed above has been done on static network. In real world, networks are temporal in nature that is they change in terms of structure at different timestamp. Recently researchers have analysed various aspects of temporal network by giving an insight into the day to day changing social networks [15–17]. Researchers have worked on opinion formation on evolving/ temporal networks. Vazquez and Federico [6] have made a survey on the voter model and threshold model for static and evolving network. Kozma et al. [8] have implemented deffuant model [3] on static and adaptive network. By adaptive they mean that nodes can break their connection and get linked to other nodes. Results show that in adaptive network, consensus is difficult to achieve compared to static network. Maity et al. [7] have studied opinion dynamics on time varying data set in which interval of variation is different. These works on evolving networks consider one on one interaction between nodes. Weighted opinions are not taken into consideration while sharing the opinions.

3 Mathematical Modeling of the Proposed Work

In the proposed work, we model our approach by implementing the directed, weighted and temporal aspects. In real world, networks grow with time, for example, new individuals are introduced in the network and new connections are formed among individuals. However, in our work, the number of nodes remains fixed and only the connection among nodes changes with time. It may happen that with time, some of the old connections might disappear but generally the rate of forming new connections is always greater than the disappearance of old connections with time. This phenomenon is also visible in infrastructure networks such as railway network, airport network [18]. Thus, taking inspiration from the real world, network is growing in terms of edges in our model. In other words, at every new time stamp, the probability of adding new edges is higher than the deletion of the edges. By weighted, we mean that the opinions shared among the nodes are weighted according to their importance in network. An individual do not adopt complete opinion from its neighbours rather fraction of opinion is shared. Indegree of the nodes is considered as neighbours of the nodes from which it will share the opinion. Initially each node i ($i \in [1, N]$, where N represents total number of the nodes in a network), is assigned a continuous random uniform opinion value (denoted by Op_i) from the range $[0,1]$. This opinion value keeps on evolving with the effect of neighbour's opinion until becomes stabilized after which there is no change in the values of opinion. At each time stamp t , opinion value of a node i is denoted by $Op_i(t)$. A set of opinion values for all the nodes in a network at timestamp t is denoted by $Op(t) = (Op_1(t), \dots, Op_N(t))$

Update Rules for the Opinion Sharing Among the Nodes in Temporal Network

At every timestamp, opinion of an individual might get changed as it gets influenced from its neighbours opinion. Proposed model for opinion exchange is described next and is summarized with the help of the Algorithm 1.

A threshold value, α is set for exchanging the opinion values among the nodes at each timestamp which is related to the difference between the opinions of the node and its neighbouring node. If the difference in the opinions of the pair of adjacent nodes is less than the threshold value then they share the opinions. A node will get the opinion from its neighbouring nodes in weighted manner according to their importance. To find the importance of the nodes, we use page rank algorithm [19] which finds the page rank of the nodes by, $PR(i) = \frac{1-d}{N} + d \sum_{j \in Ne(i)} \frac{PR(j)}{L(j)}$ where N , is the total number of nodes in the network, d is the damping factor having range $\in [0,1]$, $L(j)$ is the outgoing links from the node j , $Ne(i)$ is the set of the neighbouring nodes from where there are incoming links towards i . To calculate the page rank in the given network, if there is an out-link from the node i to node j for information flow, then an in-link is created towards i from j i.e. edges has been considered in reverse manner in which node j tries to request information from node i or try to follow node i . So, the edges in the network have been reversed and page rank of nodes is calculated.

Nodes are ranked in the decreasing order of the page rank $PR(i)$ and stored as a sorted set *sorted_rank[]* according to which they are assigned an integer number as $rank_i$. It specifies the order of the nodes as per their importance. A score $score_i$ is assigned to all the nodes by using its respective $rank_i$ (line number 6–8, Algorithm 1),

$$score_i = \left[1 - \frac{rank_i - 1}{N} \right] \quad (1)$$

where, score, opinion and rank $\in \mathbb{R}^N$.

Score at time t is a set of scores of all the nodes at the respective time,

$$score(t) = [score_1(t), score_2(t), \dots, score_N(t)] \quad (2)$$

In real world, there are some individuals who are considered important and it is somehow difficult to ignore their opinions even if the difference in their opinion and its neighbours opinion is high. For example, in a working environment we are often obliged to follow the opinions of superiors. Thus, there might be some individuals whose opinions have to be considered without considering the threshold value. To model this real world scenario, we use the concept of top ranked nodes using page rank. Initially the number of top ranked nodes Num_Top is found with the help of F_Top which is the fraction (1%, 2% or 10%) of top ranked nodes. Nodes are arranged in decreasing order according to page rank in set *sorted_rank[]*. A set T i.e set of top ranked nodes consists of the Num_Top starting nodes of the set *sorted_rank[]* (line 16–17, Algorithm 1).

Algorithm 1 Evolving_Opinions($Op, \alpha, t, N, E, rank_i, score_i$)

1: Assign opinion values to the nodes according to the uniform random distribution between [0,1]
 2: Fix a threshold value α for the network.
 3: Initialize $k = 1$
 4: **top:**
 5: Compute page rank for all the nodes.
 6: Nodes are ranked in the decreasing order of the page rank $PR(i)$ and stored in a set $sorted_rank[]$
 7: Assignment of an integer number $rank_i$ according to the decreasing order of the nodes.
 8: Assign a score to all the nodes i that is $score_i$ according to their rank by:- $\left[1 - \frac{rank_i - 1}{N}\right]$
 9: Call Function **CalculateOpinions()** to update opinions (Op_i) for all the nodes $i \in N$.
 10: Return the updated opinions (Op_i) for all the nodes.
 11: Break if $k = t$
 12: Addition and deletion of the edges E leaving it a growing network for the next time stamp.
 13: $k = k + 1$
 14: **goto top.**
 15: **function** CALCULATEOPINIONS($Ne(i), Op, score, A_{xy}$)
 16: Initialize $Num_Top = int(\lceil F_Top \times N \rceil)$
 17: $T.append(sorted_rank[: Num_Top])$
 18: $X = < Op, score >$
 19: Initialize $i = 1$
 20: **top:**
 21: **for** $Ne(i)_j$ **do**
 22: **if** $|Op_i - Op_{Ne(i)_j}| < \alpha$ or $Ne(i)_j \in T$ **then**
 23: $Op_i = X_i + A_{Ne(i)_j, i} \times X_{Ne(i)_j}$
 24: $j=j+1$
 25: Final_ $Op_i = Op_i$
 26: $Value_i = \sum_{j=1}^{k_i} score_{Ne(i)_j}$
 27: $Normalized_Op_i = Final_Op_i / Value_i$
 28: $i = i + 1$
 29: Break if $i = N$
 30: **goto top**

A node i ' neighbouring node set is represented by $Ne(i)_j$, where $i \in [1, N]$, $j \in [1, k_i]$, where k_i is the neighbour of the node i which is the indegree of the node i . Node i can take opinion from a node in its neighbouring set if anyone of the following cases will be satisfied (line 22, Algorithm 1),

- **Case 1:** If difference in the opinion of the node i and the neighboring node $Ne(i)_j$ exist in the threshold limit. $|Op_i - Op_{Ne(i)_j}| < \alpha$ where $\alpha \in \mathbb{R}[0, 1]$

- **Case 2:** If the neighboring node is in its set S , $Ne(i)_j \in T$

Opinion of a node i gets changed according to the dot product of a set of opinion and score. Opinion of a node is multiplied with its respective score which is considered as X , an intermediate value for further calculating the opinion by, $X = \langle \mathbf{Op}, \mathbf{score} \rangle$

A set X for all the nodes in a network at timestamp t is denoted as in Eq. (3) which changes at every timestamp. Set X at next time stamp $t+1$ can be calculated by the dot product of opinion at previous timestamp t and score at time $t+1$ as in Eq. (4) which is further used to calculate final opinion at timestamp $t+1$.

$$X(t) = [X_1(t), X_2(t), \dots, X_N(t)] \quad (3)$$

$$X(t+1) = Op(t) \cdot score(t+1) \quad (4)$$

where, $\langle \cdot, \cdot \rangle$ is a dot product, $\mathbf{Op} = Op_i \in \mathcal{R} | 0 < Op_i < 1, \forall i : i \in [1, N]$, $\mathbf{score} = score_i \in \mathcal{R} | 0 < score_i < 1, \forall i : i \in [1, N]$.

Here, score is calculated using rank as described in Eq. (1) which changes at every time-stamp as network restructures.

$Final_Op_i$ is the complete opinion received by a node at any time-stamp considering threshold value and the fraction of the top ranked nodes (line 25, Algorithm 1),

$$Final_Op_i = X_i + A_{Ne(i)_j, i} \times X_{Ne(i)_j} \quad (5)$$

where, $A_{Ne(i)_j, i}^T$ is the adjacency matrix between $Ne(i)_j$ and i , $Ne(i)_j \in$ case 1 or case 2.

$Final_Op$ at next time-stamp $t+1$ is calculated as,

$$Final_Op(t+1) = X(t+1) + A(t+1) \times X(t+1) \quad (6)$$

Update of opinions is sequential with deterministic node selection.

With the help of Fig. 1, we exemplify how the proposed method can model the opinion dynamics. Let Num_Top is 5 for the considered network and 0.2 is the threshold value. Considering the examples of opinion sharing, difference between opinion of node 1 and node 4 is 0.6 which is higher than threshold, but $rank_4$ is 1 which shows that it is top ranked node. Therefore, node 1 will take the opinion from node 4. Difference between the opinion of the node 1 and node 5 is 0.1 which is in the limit of threshold, therefore can share the opinion. Difference between the opinion of node 1 and node 2 is 0.5 which is higher than threshold, and page rank of node 2 is 6 which means this node is not included in the top ranked nodes. Therefore, node 1 will not take opinion from node 2.

While sharing the opinions, opinions of the neighboring nodes are multiplied by their page score which is then normalized by the page score of the nodes included instead of directly taking average of the opinion of the nodes. $Normalized_Op_i$ is the final updated value of the opinion of a node with normalizing factor at any time stamp (line 26–27, Algorithm 1). This procedure of opinion update to find the opinion at next time stamp iterates as long as the opinions converge.

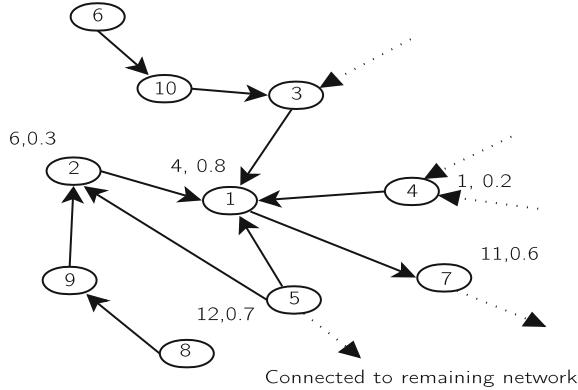


Fig. 1. An example showing different cases of Opinion Sharing with inside number as node id and outside tuple as $rank_i$ first and then Op_i

4 Results and Analysis

The proposed model is evaluated using random network with temporal effect with 500 and 1000 nodes with 0.2 as the probability of having an edge between the nodes. To make it temporal, new edges are added and existing are deleted. 10% of the edges are added and the 5% of the edges are deleted at every timestamp on the network keeping p , 0.2. TS is the timestamp at which opinions converge at some specific threshold value and F_Top value (fraction of top ranked nodes).

4.1 Impact of Threshold Value on the Convergence of Opinions

Simulations have been performed on a network of 500 nodes to analyze how the opinions are getting updated at every timestamp. Threshold values keep on changing during the simulation keeping F_Top fixed at 10%. When threshold value is decreased then either timestamp or the number of clusters in which the opinions converge increased. Threshold value acts as a tuning parameter for deciding the number of timestamp and the clusters in which opinions converge. When the threshold value is 1, opinions converge in 4 timestamps to a single cluster as in Fig. 2a which increases to 5 when threshold value is decreased to 0.5 as in Fig. 2b. Similarly convergence of opinions increases to 7 and 8 respectively as threshold value is decreased to 0.15 and 1 as in Fig. 2c, d respectively. Finally it is observed that when threshold value is increased, then opinions start converging quickly as more nodes start sharing the opinions. After getting the convergence point, there is no further change in opinions of nodes as they reach to a stable state. For example, in social network, low threshold means people are not so eager to listen the others which might leads to emergence of many groups of people or convergence of opinions become slower.

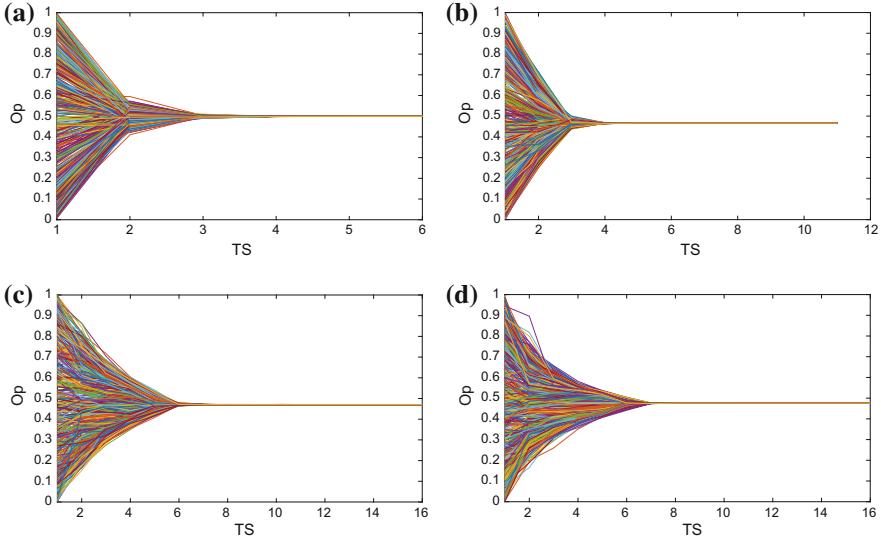


Fig. 2. Convergence of opinions with change in threshold value (α) as (a) 1 (b) 0.5 (c) 0.15 (d) 0.1 keeping fraction of top ranked nodes (F_Top) fixed at 10%. Different colors signify different opinions (Op) considered

4.2 Impact of Fraction of Top Ranked Nodes on the Convergence of Opinions

There is one more factor to be considered for opinion sharing among the nodes that is the fraction of top ranked nodes (F_Top) which we kept constant in our previous evaluation strategy. With the change in the F_Top , there is effect on the convergence of opinions e.g. Fig. 3 keeping threshold value fixed at 0.15. When F_Top is decreased then either timestamp or the number of clusters in which the opinions converge increased. When F_Top is decreased to very small value (0.001), opinions converge into multiple clusters. Hence, F_Top acts as another tuning parameter for deciding the number of timestamp and the clusters in which opinions converge. In Fig. 3a, opinions converge in 1 cluster in 10 timestamps when F_Top is 10%. With decrease in F_Top to 2%, opinions converge in 1 cluster in 24 timestamps as in Fig. 3b which increases to clusters 2 and decreases to timestamps 10 when F_Top is considered very small as in Fig. 3d. In Fig. 3d, opinions become stable to 0.22 and 0.69 at timestamp 10. For example in real world, there are some top ranked individuals which have very high influence among their neighbours. Neighbours have to accept the opinion of these highly influential individuals for example, due to their position in the society, even if there is a gap in their opinions. Opinions of such influential individual make their neighbour's opinion to come close to their own opinions quickly which results in less number of clusters or less convergence timestamp.

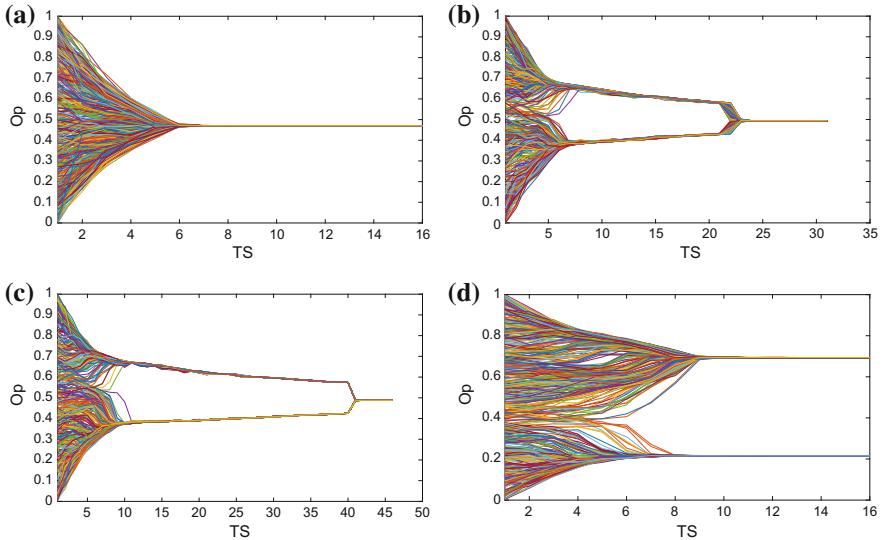


Fig. 3. Convergence of opinions with change in fraction of top ranked nodes (F_Top) as (a) 10% (b) 2% (c) 1% (d) 0% with threshold value (α) 0.15

4.3 Combined Impact of Threshold Value and Fraction of the Top Ranked Nodes on the Number of Clusters and Timestamp in Which Opinions Converge Along with Change in Number of Nodes

It is to be noted that while opinion sharing and update, opinions converge into clusters of different interest depending on the threshold value and the F_Top . With decrease in these parameters, either timestamp or the number of clusters in which opinions converge is increased. Therefore, these parameters can be tuned accordingly and desired number of clusters can emerge. In real world, desired number of clusters of opinions can be obtained which signifies different directions in which a society thinks like positive, neutral or negative. Opinions of the individuals in these clusters can be changed completely from positive to negative or vice versa by targeting influential nodes in the respective clusters. The values of threshold and the F_Top are assumed to vary simultaneously as mentioned in Table 1, where number of nodes considered is 500 and 1000. Here, even if the threshold value is same, convergence of opinions may change with change in parameter F_Top . In case of threshold value 1, all the nodes exchange the opinions and there is no need of mentioning the F_Top . Timestamp or the number of clusters in which opinions converge may increase or remains the same as the number of nodes in the network changes from 500 to 1000 in most of the cases which signifies that as the network size increases, either large number of clusters of opinions are emerged or comparatively higher threshold value has to be considered to decrease the number of clusters in which we want opinions to

converge. By higher threshold value, we mean that the people whose difference in opinions is higher can share their opinion in large size network in comparison to the small size network.

Table 1. Combined effect of change in threshold (α) and the fraction of the top ranked nodes (F_Top) on the convergence timestamp (TS) and the convergence clusters (C) along with change in number of nodes (N)

Fraction of top ranked nodes	Threshold value	No. of clusters (500 nodes)	No. of TimeStamp (500 nodes)	No. of clusters (1000 nodes)	No. of TimeStamp (1000 nodes)
10	0.1	1	8	1	8
2	0.1	1	32	1	40
1	0.1	1	74	3	30
0.001	0.1	4	10	5	9
10	0.15	1	7	1	7
2	0.15	1	24	1	21
1	0.15	1	42	2	30
0.001	0.15	2	10	2	11
10	0.3	1	7	1	7
2	0.3	1	7	1	7
1	0.3	1	7	1	7
0.001	0.3	1	7	1	7
10	0.5	1	5	1	5
2	0.5	1	5	1	5
1	0.5	1	5	1	5
0.001	0.5	1	5	1	5
-	1	1	4	1	4

5 Conclusions and Future Work

In this work, we have proposed a directed and weighted model for opinion dynamics. In addition, we have taken various parameters into consideration while exchanging the opinions, e.g., page rank, threshold value, giving priority to the top ranked nodes according to page rank. The underlying topology of the network is temporal and continuous opinions are considered. Various simulations and analysis are made to understand how opinions converge by tuning different parameters. We observed that with the increase in the threshold value and fraction of top ranked nodes, either timestamp or the number of clusters in which they converge decreases. Therefore, threshold value (α) and fraction of top ranked nodes (F_Top) act as tuning parameters for deciding the number of timestamp and the clusters in which opinions converge. One more thing is observed that even if the value of threshold is same, with change in fraction of top ranked nodes, convergence of opinions change. As the network size increases, timestamp or the number of clusters in which opinions converge also increases.

In our future work, instead of random opinion assignment to nodes, we plan to consider biased assignment. In addition, we will consider the addition and deletion of nodes in the network. We would also like to incorporate trust factor of the nodes while exchanging the opinions. We would also like to find the influential nodes among different clusters after the convergence of opinions. By influencing the opinion of these nodes, opinion of the cluster can be reframed. We would also like to perform analysis on scale-free, and small world networks along with some real datasets.

Acknowledgements. This work is supported by H2020 framework project, SoBig-Data.

References

1. Del Vicario, M., et al.: The spreading of misinformation online. *Proc. Ntl Acad. Sci.* **113**(3), 554–559 (2016)
2. Zollo, F., Quattrociocchi, W.: Misinformation spreading on facebook. arXiv preprint [arXiv:1706.09494](https://arxiv.org/abs/1706.09494) (2017)
3. Deffuant, G., Neau, D., Frederic, A., Weisbuch, G.: Mixing beliefs among interacting agents. *Adv. Complex Syst.* **3**(1–4), 87–98 (2000)
4. Zhang, Y.H., Liu, Q.P., Zhang, S.Y.: Opinion formation with time-varying bounded confidence. *PloS One* **12**(3), e0172982 (2017)
5. Kandiah, V., Shepelyansky, D.L., Vivek Kandiah and: Pagerank model of opinion formation on social networks. *Phys. A Statist. Mech. Appl.* **391**(22), 5779–5793 (2012)
6. Vazquez, F.: Opinion dynamics on coevolving networks. In: *Dynamics on and of Complex Networks*, Vol. 2, pp. 89–107. Springer (2013)
7. Maity, S.K., Manoj, T.V., Mukherjee, A.: Opinion formation in time-varying social networks: the case of the naming game. *Phys. Rev. E* **86**(3), 036110 (2012)
8. Kozma, B., Barrat, A.: Consensus formation on adaptive networks. *Phys. Rev. E* **77**(1), 016102 (2008)
9. Galton, F.: Vox populi (the wisdom of crowds). *Nature* **75**(7), 450–451 (1907)
10. Das, A., Gollapudi, S., Munagala, K.: Modeling opinion dynamics in social networks. In: *Proceedings of the 7th ACM International Conference on Web Search and Data Mining*, pp. 403–412 (2014)
11. Clifford, P., Sudbury, A.: A model for spatial conflict. *Biometrika* **60**(3), 581–588 (1973)
12. Yildiz, M.E., Pagliari, R., Ozdaglar, A., Scaglione, A.: Voting models in random networks. In: *Information Theory and Applications Workshop (ITA)*, pp. 1–7. IEEE (2010)
13. DeGroot, M.H.: Reaching a consensus. *J. Am. Statist. Assoc.* **69**(345), 118–121 (1974)
14. Hegselmann, R., Krause, U., et al.: Opinion dynamics and bounded confidence models, analysis, and simulation. *J. Artifi. Societies Soc. Simul.* **5**(3) (2002)
15. Wehmuth, K., Ziviani, A., Fleury, E.: A unifying model for representing time-varying graphs. In: *IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, 2015. 36678 2015, pp. 1–10. IEEE (2015)
16. Kostakos, V.: Temporal graphs. *Phys. A Statist. Mech. Appl.* **388**(6), 1007–1023 (2009)

17. Masuda, N., Lambiotte, R.: A guide to temporal networks. Vol. 4. World Scientific (2016)
18. Liu, C., Li, J.: Small-world and the growing properties of the chinese railway network. *Front. Phys. China* **2**(3), 364–367 (2007)
19. Page, L., Brin, S., Motwani, R., Winograd, T.: The pagerank citation ranking: bringing order to the web. Technical Report, Stanford InfoLab (1999)



Full-Commanding a Network: The Dictator

Clara Grácio¹(✉), Sara Fernandes¹, and Luís Mário Lopes²

¹ Department of Mathematics, Universidade de Évora and CIMA-UE,
Rua Romão Ramalho, 59, 7000-671 Évora, Portugal

{mgracio,saf}@uevora.pt

² Mathematics Unit, ADM, Instituto Superior de Engenharia de Lisboa,
Rua Conselheiro Emídio Navarro, 1, 1949-014 Lisboa, Portugal
luismariolopes@gmail.com

Abstract. A network of chaotic dynamical systems may synchronize. For some networks there is the possibility that, coupling a new node to the network, the synchronization will be commanded by that new node. That possibility depends on the network and on the way the new node is coupled to the network. We consider a coupling that can provide what we call a full-commanding and we define the corresponding full-command-window. The limit situations corresponding to a completely connected network and to a completely disconnected one provide us some understanding about what makes a network more receptive or more resistant to commanding.

Keywords: Chaotic dynamical systems · Complete synchronization · Command · Full-command windows · Dictator

1 Introduction

The synchronization of coupled chaotic dynamical systems is largely studied in literature [1–5]. In Sect. 2, we consider a network of identical one-dimensional chaotic discrete dynamical systems connected to each other in a linear way and we obtain conditions for the synchronization to take place. The possibility of commanding the synchronized behavior is obviously a relevant subject. We analyse the possibility to achieve that by adding a new node in a full-commanding way, and we call this new node the dictator. In Sect. 3, we analyse the conditions that a network needs to satisfy to be full-commandable. Even if a network is full-commandable, not all the values of the coupling strength constant that couples the dictator to the other nodes of the network allow the dictator to command the network. Those which do so define the full-command-window. In Sect. 3, we also obtain that window and particularize it for a completely connected network and for a completely disconnected one, getting some conclusions about what may provide more resistance or more receptiveness to the full-commanding by a dictator.

2 Complete Synchronization in a Network

We consider a network of N one-dimensional chaotic discrete dynamical systems, x_i ($i = 1, \dots, N$), with the same dynamic defined by the map f , described by

$$x_i(t+1) = f(x_i(t)) + \sum_{\substack{j=1 \\ j \neq i}}^N a_{ij} \cdot [f(x_j(t)) - f(x_i(t))]$$

or, equivalently, by

$$x_i(t+1) = (1 - \sum_{\substack{j=1 \\ j \neq i}}^N a_{ij}) \cdot f(x_i(t)) + \sum_{\substack{j=1 \\ j \neq i}}^N a_{ij} \cdot f(x_j(t))$$

where $a_{ij} \in \mathbb{R}_0^+$ ($i, j = 1, \dots, N$) are the coupling constants.

Defining $\vec{x}(t) = [x_1(t) \ x_2(t) \ \dots \ x_N(t)]^T$ and $\vec{f}(\vec{x}(t)) = [f(x_1(t)) \ f(x_2(t)) \ \dots \ f(x_N(t))]^T$, $a_{ii} = -\sum_{\substack{j=1 \\ j \neq i}}^N a_{ij}$ and $A = [a_{ij}]$, these equations may be written as

$$\vec{x}(t+1) = (I_N + A) \cdot \vec{f}(\vec{x}(t)) \quad (1)$$

and we call A the coupling matrix.

We note that $\lambda = 0$ is an eigenvalue of A (in fact, $A \cdot \vec{1} = \vec{0} = 0 \cdot \vec{1}$, with $\vec{1} = [1 \dots 1]^T$). Further, network (1) admits a completely synchronized solution (see definition hereafter), namely $\vec{x}(t) = x_1(t) \cdot \vec{1}$ is a solution of (1), for any function $x_1(t)$ that satisfies $x_1(t+1) = f(x_1(t))$.

Definition 1. We say that the coupled system (1) admits a completely synchronized solution if there is a synchronized function $s(t)$ such that $(x_1(t), x_2(t)) = (s(t), s(t))$ is a solution of (1).

The following proposition (a variant of similar results already presented [6, 7]) provides conditions for the existence of an exponentially stable completely synchronized solution.

Proposition 1. Considering the dynamical network (1) with A a diagonalizable matrix such that $\lambda_1 = 0$ is an eigenvalue of multiplicity 1, if all the other eigenvalues λ_i ($i = 2, \dots, N$) are such that $|1 + \lambda_i| < e^{-h}$, where h is the Lyapunov exponent of the nodes, then the completely synchronized solution $\vec{x}(t) = x_1(t) \cdot \vec{1}$, with $x_1(t)$ satisfying $x_1(t+1) = f(x_1(t))$, is exponentially stable. If $|1 + \lambda_i| > e^{-h}$ for a non-zero eigenvalue λ_i , then there is not an exponentially stable completely synchronized solution.

Proof. Considering $\vec{x}(t) = x_1(t) \cdot \vec{1} + \vec{u}(t)$, the linearization of (1) around $\vec{x}(t) = x_1(t) \cdot \vec{1}$ provides

$$\begin{aligned} x_1(t+1) \cdot \vec{1} + \vec{u}(t+1) &= \\ &= (I_N + A) \cdot [f(x_1(t)) \vec{1} + f'(x_1(t)) \cdot \vec{u}(t)] \end{aligned}$$

and, since $A \cdot \vec{1} = 0 \cdot \vec{1}$ and $x_1(t+1) = f(x_1(t))$, we obtain

$$\vec{u}(t+1) = (I_N + A) \cdot f'(x_1(t)) \cdot \vec{u}(t)$$

Considering $D = \text{diag}(0, \lambda_2, \lambda_3, \dots, \lambda_N)$, a diagonal matrix similar to A , and $\vec{v}(t) = S \cdot \vec{u}(t)$, with S such that $D = SAS^{-1}$, the previous equation is equivalent to

$$\begin{aligned} \vec{v}(t+1) &= (I_N + D) \cdot f'(x_1(t)) \cdot \vec{v}(t) \\ &\Leftrightarrow \\ \begin{cases} v_1(t+1) = f'(x_1(t)) \cdot v_1(t) \\ v_i(t+1) = (1 + \lambda_i) \cdot f'(x_1(t)) \cdot v_i(t), \quad i = 2, \dots, N \end{cases} \end{aligned}$$

Then, for $i = 2, \dots, N$,

$$v_i(t+T) = (1 + \lambda_i)^T \cdot \prod_{\tau=t}^{t+T} f'(x_1(\tau)) \cdot v_i(t),$$

and since the Lyapunov exponent h of $x_1(t+1) = f(x_1(t))$ is $h = \lim_{T \rightarrow +\infty} \frac{\sum_{\tau=t}^{t+T} \ln |f'(x_1(\tau))|}{T}$, we obtain

$$\begin{aligned} \lim_{T \rightarrow +\infty} |v_i(t+T)| &= \\ &= \lim_{T \rightarrow +\infty} \left| (1 + \lambda_i)^T \right| \cdot e^{\ln \prod_{\tau=t}^{t+T} |f'(x_1(\tau))|} \cdot |v_i(t)| = \\ &= \lim_{T \rightarrow +\infty} (|1 + \lambda_i| e^h)^T \cdot |v_i(t)| \end{aligned}$$

If $|1 + \lambda_i| e^h < 1$ or, equivalently, $|1 + \lambda_i| < e^{-h}$ for $i = 2, \dots, N$, then the completely synchronized solution $\vec{x}(t) = x_1(t) \cdot \vec{1}$ is exponentially stable.

If there is a $i \in \{2, \dots, N\}$ such that $|1 + \lambda_i| e^h > 1$ or, equivalently, $|1 + \lambda_i| > e^{-h}$, then there is not an exponentially stable completely synchronized solution.

Now, we introduce another node y - the dictator - and we want to analyse how it can command the network and how the network can resist to its command. The analysis done in [8] regarding what was called there the Commanded Linear Coupled System, suggests that we make a one-way connection of the new node y to all the other nodes. We will consider a **full-commanding**: one in which the dictator is connected to all other nodes with the same commanding coupling strength constant $\epsilon \in \mathbb{R}_0^+$.

3 Full-Commanding

Using a new node y to full-command the network (1) originates the following new network:

$$\begin{cases} x_i(t+1) = f(x_i(t)) + \sum_{\substack{j=1 \\ j \neq i}}^N a_{ij} \cdot [f(x_j(t)) - f(x_i(t))] + \epsilon \cdot [f(y(t)) - f(x_i(t))] \\ y(t+1) = f(y(t)) \end{cases}$$

with $i = 1, \dots, N$, or, equivalently,

$$\begin{cases} x_i(t+1) = (1 - \sum_{\substack{j=1 \\ j \neq i}}^N a_{ij} - \epsilon) \cdot f(x_i(t)) + \sum_{\substack{j=1 \\ j \neq i}}^N a_{ij} \cdot f(x_j(t)) + \epsilon \cdot f(y(t)) \\ y(t+1) = f(y(t)) \end{cases}$$

Defining $\vec{x}_0(t) = [x_1(t) \ x_2(t) \ \dots \ x_N(t) \ y(t)]^T$ and $A_0 = \begin{bmatrix} A - \epsilon I_N & \epsilon \vec{1} \\ \vec{0}^T & 0 \end{bmatrix}$,

these equations may be written as

$$\vec{x}_0(t+1) = (I_{N+1} + A_0) \cdot \vec{f}(\vec{x}_0(t)) \quad (2)$$

Proposition 2. *If $|\epsilon - (1 + \lambda_i)| < e^{-h}$, for all the eigenvalues λ_i of the diagonalizable coupling matrix A of the network (1), then the dictator y commands the network, i.e. $\vec{x}_0(t) = y(t) \cdot \vec{1}$ is an exponentially stable solution of (2). If $|\epsilon - (1 + \lambda_i)| > e^{-h}$ for an eigenvalue λ_i , then $\vec{x}_0(t) = y(t) \cdot \vec{1}$ is not an exponentially stable solution of (2), i.e. the dictator is not able to command the network in a full-commanding way.*

Proof. The eigenvalues of A_0 are $\lambda = 0$ and $\lambda = \lambda_i - \epsilon$. Since the sum of the entries in each row of A_0 is equal to zero and the off-diagonal elements are nonnegative, lemma 2 of [9] assures that $\text{Re}(\lambda_i) \leq 0$ for all eigenvalues λ_i of A . Further $\epsilon \in \mathbb{R}_0^+$. Then A_0 has just one zero eigenvalue and Proposition 1 determines that if $|1 + \lambda_i - \epsilon| < e^{-h}$ or, equivalently, $|\epsilon - (1 + \lambda_i)| < e^{-h}$, then the synchronized solutions $\vec{x}_0(t) = x_1(t) \cdot \vec{1}$, with $x_1(t)$ satisfying $x_1(t+1) = f(x_1(t))$, are exponentially stable. Since y is a free node, i.e. since y satisfies $y(t+1) = f(y(t))$, then $\vec{x}_0(t) = y(t) \cdot \vec{1}$ is an exponentially stable solution of (2).

Proposition 1 also determines that if $|\epsilon - (1 + \lambda_i)| > e^{-h}$ for at least one of the eigenvalues λ_i of A , then $\vec{x}_0(t) = y(t) \cdot \vec{1}$ is not an exponentially stable solution of (2).

This lead us to define

Definition 2. We define full-command-window (FCW) of the network (1) as the open set of values of the commanding coupling constant ϵ for which the synchronized solution $\vec{x}_0(t) = y(t) \cdot \vec{1}$ is an exponentially stable solution of (2). If $FCW \neq \emptyset$, we say that network (1) is full-commandable.

Proposition 2 determines that

$$FCW = \bigcap_{i=1}^N \{\epsilon \in \mathbb{R}_0^+ : |\epsilon - (1 + \lambda_i)| < e^{-h}\}$$

and we note that there are networks that are not full-commandable, as is established in the following proposition.

Proposition 3. If a diagonalizable coupling matrix A has two eigenvalues that are distant from each other more than $2e^{-h}$, then the network (1) with the coupling matrix A is not full-commandable. That is also true if A has an eigenvalue λ such that $\text{Im}(\lambda) > e^{-h}$.

Proof. We name λ_1 and λ_2 the two eigenvalues of A such that $|\lambda_1 - \lambda_2| > 2e^{-h}$. Let us suppose that A is full-commandable. Then, Proposition 2 assures that $|\epsilon - (1 + \lambda_1)| \leq e^{-h}$ and $|\epsilon - (1 + \lambda_2)| \leq e^{-h}$, providing $|\lambda_1 - \lambda_2| = |[\epsilon - (1 + \lambda_2)] - [\epsilon - (1 + \lambda_1)]| \leq |\epsilon - (1 + \lambda_1)| + |\epsilon - (1 + \lambda_2)| \leq e^{-h} + e^{-h} = 2e^{-h}$. This contradicts our assumption and, so, we conclude that A is not full-commandable.

If there is a λ such that $\text{Im}(\lambda) > e^{-h}$ then

$$|\epsilon - (1 + \lambda)| = \sqrt{(\epsilon - 1 - \text{Re}(\lambda))^2 + (\text{Im}(\lambda))^2} > \text{Im}(\lambda) > e^{-h}$$

and so the network is not full-commandable.

There are situations for which we can obtain a simpler description of FCW .

Proposition 4. Considering a network (1) such that A is diagonalizable, all its eigenvalues are real and λ_N is the smallest one, then the network is full-commandable if $\lambda_N > -2e^{-h}$ and FCW reduces to $]1 - e^{-h}, 1 + \lambda_N + e^{-h}[$.

Proof. Since A is diagonalizable, we can use Proposition 2. If all the eigenvalues of A are real, then $FCW = \bigcap_{i=1}^N]1 + \lambda_i - e^{-h}, 1 + \lambda_i + e^{-h}[$. If λ_1 and λ_N are the largest and smallest eigenvalues of A , respectively, the intersection that defines the FCW will be nonempty if and only if $1 + \lambda_N + e^{-h} > 1 + \lambda_1 - e^{-h}$ or, equivalently, $\lambda_N > \lambda_1 - 2e^{-h}$ and, in that situation $FCW =]1 + \lambda_1 - e^{-h}, 1 + \lambda_N + e^{-h}[$. Since the sum of the entries in each row of A is equal to zero and the off-diagonal elements are nonnegative, lemma 2 of [9] determines that $\lambda_i \leq 0$. Further, zero is an eigenvalue of A associated to the eigenvector $\vec{1}$. So, $\lambda_1 = 0$ and we conclude that the network (1) is full-commandable if $\lambda_N > -2e^{-h}$ and $FCW =]1 - e^{-h}, 1 + \lambda_N + e^{-h}[$.

It is worthwhile to emphasize that for a full-commandable network in the conditions of the previous Proposition, the smallest commanding coupling constant is $\epsilon = 1 - e^{-h}$, a value that does not depend on the structure of the network, it only depends on the dynamic of the nodes.

Since a symmetric matrix is diagonalizable and all its eigenvalues are real, we can use Proposition 5 for a symmetric network (1), i.e for a network (1) corresponding to a symmetric coupling matrix. We consider two limit situations in what respects to connectedness:

Proposition 5. *A completely disconnected network, i.e. a network (1) with a zero matrix A , is full-commandable and its full-command-window is $]1 - e^{-h}, 1 + e^{-h}[$.*

Proof. All the eigenvalues of a zero matrix are zero. So, Proposition 5 determines that the network is full-commandable (since $0 > -2e^{-h}$) and $FCW =]1 - e^{-h}, 1 + e^{-h}[$.

Proposition 6. *A completely connected network, i.e. a network (1) with*

$$A = c \cdot \begin{bmatrix} -(N-1) & 1 & 1 & \dots & 1 \\ 1 & -(N-1) & 1 & \dots & 1 \\ 1 & 1 & -(N-1) & \dots & 1 \\ \dots & \dots & \dots & \dots & \dots \\ 1 & 1 & 1 & \dots & -(N-1) \end{bmatrix},$$

where c is the global coupling strength, is full-commandable if $Nc < 2e^{-h}$ and its full-command-window is $]1 - e^{-h}, 1 - Nc + e^{-h}[$.

Proof. Considering $\det(\frac{1}{c}A - \lambda I_N)$, if we substitute the first row (R_1) by the sum of all the rows ($\sum_{i=1}^N R_i$), and all the remaining rows (R_j , with $j = 2, \dots, N$) by $R_j + \frac{1}{\lambda} \sum_{i=1}^N R_i$, we obtain

$$\begin{aligned} \det\left(\frac{1}{c}A - \lambda I_N\right) &= \\ &= \det \begin{bmatrix} -\lambda & -\lambda & -\lambda & \dots & -\lambda \\ 0 & -N - \lambda & 0 & \dots & 0 \\ 0 & 0 & -N - \lambda & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & \dots & -N - \lambda \end{bmatrix} = \\ &= -\lambda \cdot (-N - \lambda)^{N-1}, \end{aligned}$$

i.e. $\lambda = 0$ and $\lambda = -N$ are the eigenvalues of $\frac{1}{c}A$ of multiplicity 1 and $N-1$, respectively. So, the eigenvalues of A are $\lambda = 0$ and $\lambda = -Nc$ and Proposition 5 determines that if $Nc < 2e^{-h}$ then the completely connected network is full-commandable and that its full-command-window is $FCW =]1 - e^{-h}, 1 - Nc + e^{-h}[$.

4 Conclusions

Some networks (1) may be full-commanded by a dictator. The full-commanding possibility depends on the structure of the network (namely on the eigenvalues of coupling matrix A) and on the nature of the nodes (namely on its Lyapunov exponent). If the network is symmetric, or if at least all the eigenvalues of A are real, the minimum effort (i.e. the minimum value of the commanding coupling strength constant) that needs to be used in order to the dictator to command the network just depends on the nature of the nodes (namely on its Lyapunov exponent). While a completely disconnected network is always full-commandable, a completely connected one is as more difficult to command, as larger is the network (larger N) and as stronger are the connections between nodes (larger c).

References

1. Pikovsky, A., Rosenblum, M., Kurths, J.: *Synchronization, A Universal Concept in Nonlinear Sciences*. Cambridge University Press, Cambridge (2001)
2. Hasler, M., Maistrenko, Y.L.: An introduction to the synchronization of chaotic systems: coupled Skew Tent Maps. *IEEE Trans. Circuits Syst. I* **44**(10), 856–66 (1997)
3. Sushchik Jr., M.M., Rulkov, N.F., Abarbanel, H.D.I.: Robustness ad stability of synchronized chaos: an illustrative model. *IEEE Trans. Circuits Syst. I* **44**(10), 867–73 (1997)
4. Rangarajan, G., Ding, M.Z.: Stability of synchronized chaos in coupled dynamical systems. *Phys. Lett. A* **296**, 204–9 (2002)
5. Ding, M.Z., Yang, W.M.: Stability of synchronous chaos and on-off intermittency in coupled map lattices. *Phys. Rev. E* **56**(4), 4009–16 (1997)
6. Feng, J., Jost, J., Qian, M.: *Networks: From Biology to Theory*. Springer, Heidelberg (2007)
7. Li, X., Chen, G.: Synchronization and desynchronization of complex dynamical networks: an engineering viewpoint. *IEEE Trans. Circuits Syst. I* **50**(11), 1381–90 (2003)
8. Lopes, L., Fernandes, S., Grácio, C.: Windows of synchronization non-chaotic windows. *ESAIM Proc. Surv.* **46**, 161–74 (2014)
9. Wu, C.W., Chua, L.O.: Synchronization in an array of linearly coupled dynamical systems. *IEEE Trans. Circuits Syst. I* **42**, 430–47 (1995)



Biased Dynamic Sampling for Temporal Network Streams

Shazia Tabassum^{1,2(✉)} and João Gama^{1,3}

¹ INESC TEC, University of Porto, Rua Dr. Roberto Frias, Porto, Portugal
shazia.tabassum@inesctec.pt

² Faculdade de Engenharia, University of Porto, Porto, Portugal
³ Faculdade de Economia, University of Porto, Porto, Portugal

Abstract. Considering the avalanche of evolving data and the memory constraints, streaming networks' sampling has gained much attention in the recent decade. However, samples choosing data uniformly from the beginning to the end of a temporal stream are not very relevant for temporally evolving networks where recent activities are more important than the old events. Moreover, the relationships also change overtime. Recent interactions are evident to show the current status of relationships, nevertheless some old stronger relations are also substantially significant. Considering the above issues we propose a fast memory less dynamic sampling mechanism for weighted or multi-graph high-speed streams. For this purpose, we use a forgetting function with two parameters that help introduce biases on the network based on time and relationship strengths. Our experiments on real-world data sets show that our samples not only preserve the basic properties like degree distributions but also maintain the temporal distribution correlations. We also observe that our method generates samples with increased efficiency. It also outperforms current sampling algorithms in the area.

1 Introduction

Handling and processing, high-velocity networked data generating from real-world applications is a current exigency. Dynamic sampling is an exemplary way to deal with the issues relating to massive evolving data, like answering approximate queries, running simulations, understanding and modeling true network structure, inadequate data, detecting events/changes in the network, etc. Apart from other applications one of its major appeals lies in estimating the true network properties [2] that cannot be handled in entirety. Though sampling population is a statistically established area, it is not much explored in the current scenario of real-time dynamic networked data. A comprehensive survey on sampling network streams can be found in [3]. Previous works in sampling streaming networks concentrated on preserving structural properties of a network at any time t . However, the temporal order of edges or their weights were not considered in building the samples, as it would be unfavorable to one pass and memory constraints. In [4] the authors studied the predictive value of links based on their

age. They concluded that the young links are more informative than the older ones. Therefore, we tried to address the above problems with the contributions stated below:

1. We propose one pass, memory less, and bounded size stream sampling algorithm that maintains strong and stable structural relationships over time in a multi/weighted graph stream.
2. Apart from the above traditional measures to evaluate stream sampling algorithms, we propose some interesting properties to look for in an on-line sample.
3. Our empirical results show that the samples from the proposed algorithm maintain temporal activity patterns, preserve the basic properties of network and perform better than state of the art methods.

Rest of the paper is organized as follows: In Sect. 2 we presented a brief overview of the related state of the art. Problem definitions are given in Sect. 3. Our method is outlined in Subsect. 3.1. Results in comparison with the true network are presented in Sect. 4. Additionally, a comparative assessment with other methods is carried out in Subsect. 4.3. Finally, the conclusions and future works are summarized in the Sect. 5.

2 Related Work

Most of the stream sampling algorithms are variants of the classic *Reservoir Sampling* algorithm [11] which is a probabilistic uniform random sampling method over a data stream. This algorithm runs into a limitation of old and stale items in the sample/reservoir, as the probability of new items entering into the reservoir decreases while the stream progresses. In [1] Aggarwal tried to introduce bias in the above algorithm by exponentially decreasing the probability of items in the reservoir to the probability of new items in the stream temporally. This bias was introduced to get an anytime sample with the results focusing more on the latest data while not completely ignoring the old data in the stream, as is ignored in sliding windows (which is another alternative to sample the latest data). However, the approach was limited to data streams.

In [3] Ahmed et al. presented a simple edge stream sampling which uses the same approach as reservoir sampling [11]. In this case, a new edge enters into the reservoir if its hash value is within top- m minimum hash values, where m is the size of reservoir. However, the method did not ensue as an efficient representative sample. Additionally, Ahmed et al. [3] also proposed a *Partially-induced edge sampling algorithm* called (PIES). This algorithm works by storing nodes and edges probabilistically in their reservoirs while deleting the one already present at random as in the reservoir sampling [11]. It maintained a fixed size reservoir of nodes while the reservoir size for edges varied based on nodes. CPIES, an update over PIES is given by Zhang et al. [12]. They modified the decremental module of PIES, by deleting the nodes from the reservoir with minimal degree to produce a better cluster preserving structure. PIES had a selection bias to

high degree nodes which enhances in CPIES as it tends to delete low degree nodes. Papagelis et al. [7] proposed sampling algorithms that given a user in a social network quickly obtains a near-uniform random sample of nodes in its neighborhood using random walks. Another simple variant of reservoir sampling was given by [9], where unlike the above algorithms the incoming edges are not chosen based on a probability function instead all the streaming edges enter the reservoir and randomly replace the edges already in it. We use a similar approach of [1] from data streams on a temporal stream of edges in a dynamic network. Instead of decreasing the probability of edges in the reservoir exponentially, we use the exponential function over the weights of edges in the network. The method works by sending every new edge into the reservoir/sample but the probability it stays in depends on the time step t when it occurred or reoccurred and its weight at t . A similar approach is used over ego network streams in [8].

3 Problem Definition

We model an evolving network as a stream of edges $\{e_1, e_2, e_3, e_4 \dots\} \in E$ generating from a graph stream G . Every edge $e = (u, v, t)$ is composed of a pair of vertices's from V and a time-stamp t , which indicates the time of occurrence of e . We assume that the edges are streaming in the order of time-stamps.

Definition 1 (Multi-graph). *The basic intuition of the proposed approach is for directed multi/weighted graphs. In a multi-graph stream an edge e can recur randomly in G at various time-stamps t . Another variable τ is a discrete time-step/time-interval with granularity defined by the user. Initially when $\tau = 0$ we have $G = \emptyset$. A (true) network G at $\tau = n$ is given as all the network from τ_0 to τ_n . At every τ , the weight w_e of an edge e is computed as*

$$w_\tau(e) = \# \text{ of occurrences of } e \text{ in } \tau \quad (1)$$

The weights $w_\tau(e)$ are also incremented in a streaming way sequentially. For weighted graphs w_e is the weight of e in the time-interval τ .

Definition 2 (Edge vector stream). *Every edge e in G can occur in multiple τ 's. Therefore every edge is a temporal vector \mathbf{a} of the state of presence (with a weight $w_\tau(e)$) or absence in every τ . M_τ is the number of unique edges in G_τ at any τ .*

Definition 3 (Edge Stability). *We define edge stability $s(e)$ as the property of an edge e in a multi-graph stream G at every τ is given as*

$$s_\tau(e) = \frac{\# \text{ of elapsed } \tau \text{'s where } e \text{ is present in } \mathbf{a}}{\text{total number of } \tau \text{'s elapsed}} \quad (2)$$

Stability lies in the range of [0,1]. If an edge e occurs in every time-step τ in \mathbf{a} , then the edge is considered most stable with the stability 1. For example, in a co-authorship network considering τ as one year. If two authors collaborate with a scientific paper for 2 years in an observational period of 4 years then their edge stability is equal to 0.5 at the end of 4 years. Therefore most stable edges in a graph stream characterize a strong relationship.

Definition 4 (Problem statement). Given an evolving temporal stream G_τ , our sampling algorithm aims to produce a sample \hat{G}_τ that preserves the importance of edges based on time τ , their strengths w_τ and stability $s_\tau(e)$.

3.1 Sampling with a Bias to Latest and Stable Edges (SBias)

For generating and biasing the sample \hat{G} at any τ , towards the recent behavior of network, we use a forgetting function from the class of memory less exponential functions. This function is applied on every $w(e)$ from M edge vector streams, after every τ to get $\hat{w}_\tau(e)$, which is the weight of an edge in \hat{G}_τ . The function is defined as follows:

$$\hat{w}_\tau(e) = w_\tau(e) + (1 - \alpha)\hat{w}_{\tau-1}(e) \quad (3)$$

The parameter α defines the bias rate and typically lies between 0 and 1 inclusive. In general, this parameter α is chosen in an application specific way. We use α according to the sample size we need. When α is 1 we forget all of the previous edge occurrences and only maintain information from the latest τ . When $\alpha = 0$ the function returns the true network G with original weights. If e does not recur in \mathbf{a} then $\hat{w}_\tau(e)$ is monotonically decreasing.

To bias the sample with strong and stable relations, we use a parameter θ . θ lies between 0 and the highest $\hat{w}_{\tau-1}(e)$ in the network, because we did not normalize the weights which would cost extra computation and knowing all $w_\tau(e)$ in advance (not favorable for one pass). After each τ , the edges in the sample with $\hat{w}_\tau(e) < \theta$ are deleted from it, retaining stronger and stable edges from M but we still have the weak edges from the latest τ because of the recent bias. Therefore the sample size decreases to \hat{M} at $\tau + 1$. If e appears again in G at a time-step $\tau + i$ it gets added to \hat{G}_τ again. A pictorial representation of our method is shown in Fig. 1. Weights of edges are updated with their frequency in the stream. A cross indicates the edge was absent in that τ .

A higher value of α retains most recent edges in \hat{G}_τ exponentially in the order of τ , while a lower value tends to include older edges also. Likewise, a higher value of θ would retain more strong and stable edges in \hat{G}_τ and a lower value retains less strong and stable ones. Therefore, we can tune the parameter according to the requirement.

The space complexity of SBias is $O(\hat{M}_\tau)$ where \hat{M}_τ is the edge-set for output at any τ , which is a minimum space requirement as an algorithm cannot require less space than output. We use hash map for storing edges and updating $w_\tau(e)$, therefore the time complexity is $O(1)$ and in the worst case is $O(\hat{M})$. The tightest bound is obtained when $\alpha = 1$, the sample is only the network from the latest time step τ . Therefore lower bound of the sample \hat{G}_τ is the individual network at τ not including the network from other τ' s, which is minimal compared to whole network until τ .

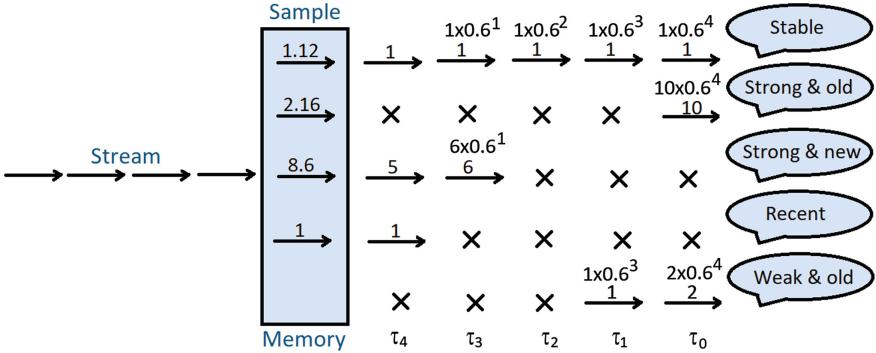


Fig. 1. Pictorial representation of SBias sample at τ_4 with $\alpha = 0.4$ and $\theta = 0.5$.

4 Experimental Evaluation

We followed two approaches for evaluating our method. Firstly we evaluated the representative properties of the samples against the original network. Then we compared our samples with the samples from Reservoir Sampling [11] and Biased Random Sampling [9].

Here we give some handy notations to help understand experiments. The true network is the network G from Definition 2. The size of the network M_τ and size of the sample \hat{M}_τ is given by the number of unique e in them. The sample% is the percentage of edges from G at the end of observed stream.

4.1 Data Sets

Call Network: We use a massive anonymized CDR (Call Detail Records) data provided by a service provider. The average speed of data is 10 to 280 calls per second around mid-night and mid-day. Calls in the data are associated with time stamps when the call was initiated (Table 1).

Facebook Wall Posts: This data set is a subset of users posts on other users wall on Facebook which is obtained from KONECT¹ networks. The detailed

Table 1. Networks' properties

Data sets	$ E $	$ V $	Unique $ E $	$ \tau $	Components	Avg. deg	Avg. wt deg	Avg. stability
Call net	389994643	12213391	95090672	30	28260	15.6	63.9	0.072
Face book	876993	46952	274086	1506	1981	37.3	71.2	0.002
College Msg	59835	1899	20296	193	4	10.7	31.4	0.009

¹ Data available at <http://konect.uni-koblenz.de/networks/facebook-wosn-wall>

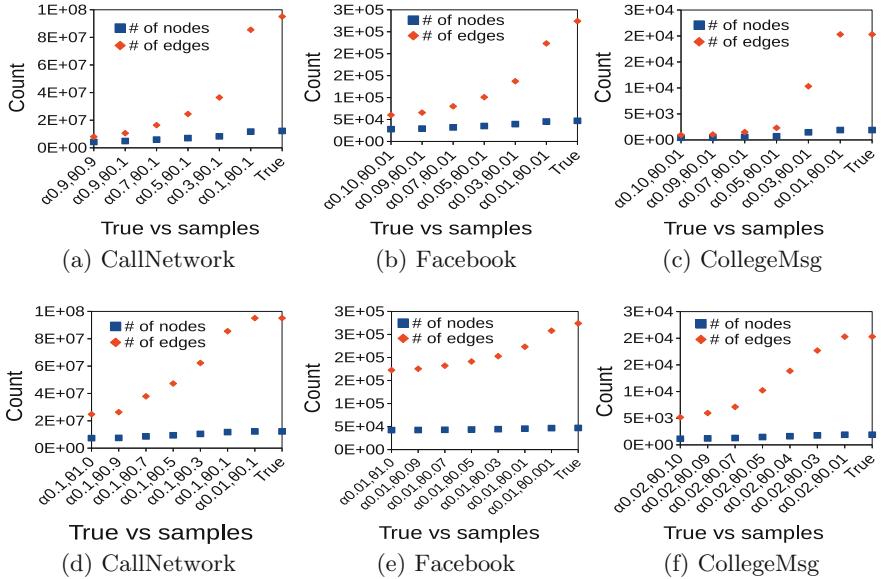


Fig. 2. Parameters α and θ influencing the size of network.

description of data is also available at [10]. The data spans from 2004 to 2009. The data is temporally very sparse (few months have just one edge).

CollegeMsg: This data set is comprised of private messages sent on an online social network at the University of California, Irvine [6], obtained from SNAP data sets².

4.2 Comparative Assessment with the True Network

Parameter Sensitivity. To evaluate the sensitivity of parameters empirically we did two experiments, firstly varying α by fixing θ and secondly varying θ by fixing α with equal intervals. We see both the outcomes in the Fig. 2 are similar but the algorithm is more sensitive to the change in α than θ . In all the experiments we see that as we increase the parameter values the number of edges is getting closer to the number of nodes which shows a structure closer to spanning tree, we will quantify this using Krackhardt efficiency in the following subsection. Note the number of nodes is also decreasing. We make use of the parameter sensitivity property to generate samples of different sizes.

Degree Distribution. For every degree d , we count the number of nodes with degree d . In all the data sets (Fig. 3) we see that the samples follow a similar distribution as in the true network. As the stream length and stability varies

² Data available at <https://snap.stanford.edu/data/CollegeMsg.html>

in three data sets we had to use different parametric values to get equal percentage samples.

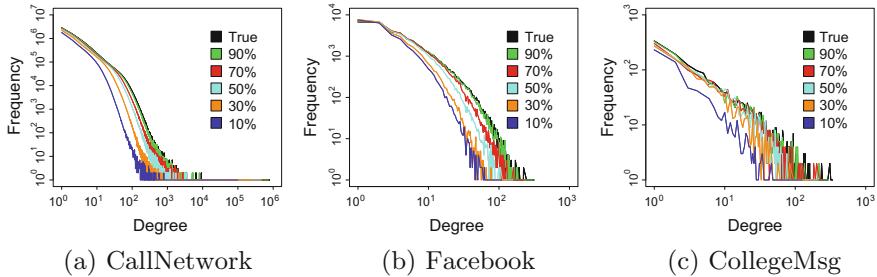


Fig. 3. Degree distributions, true network vs samples.

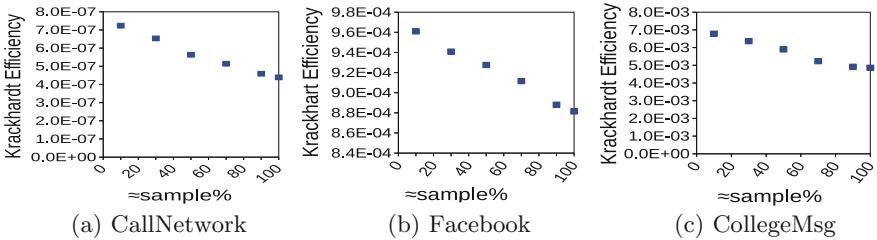


Fig. 4. Krackhardt efficiency, true network vs samples.

Krackhardt Efficiency. Krackhardt Efficiency [5] was defined by Krackhardt to measure the extent to which a graph is an out-tree or a spanning-tree (if directions in the network are not considered). It is a measure of non-redundancy in multiple components of a graph or multiple weak components of a digraph. The graph is said to be highly efficient if there are $(N_i - 1)$ number of links between N_i number of nodes in every component G_i of a graph G . The measure of efficiency is calculated with the equation below

$$1 - \frac{E(G) - \sum_{i=1}^n (N_i - 1)}{\sum_{i=1}^n (N_i(N_i - 1) - (N_i - 1))} \quad (4)$$

When there are multiple components in a graph, the above equation calculates the efficiency of components in a spanning forest. Figure 4 shows that the samples have components with an increased efficiency, that is getting close to 1 as the sample size decreases. As our method removes the weaker edges from the network we retain a sample with less redundancy and high weighted edges, approaching

a minimum spanning forest. This is a nice property as many path-finding algorithms internally build spanning trees, that increases their complexity. The true network is denoted as 100% in Fig. 4.³

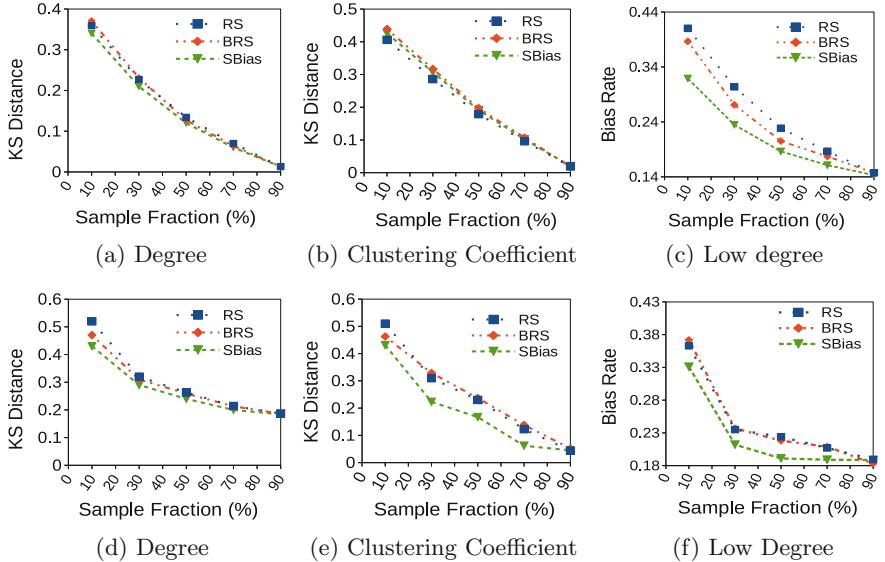


Fig. 5. KS-Distance of distributions and lowest degree bias rate in Facebook (a-c) and CollegeMsg (d-f).

4.3 Comparative Assessment with Other Methods

In the following subsection, we demonstrate two state of the art sampling algorithms that were applied on temporal multi-graph network streams in [9]. To our knowledge, these are the most related and recent approaches to our problem.

Reservoir Sampling for Edge Stream (RS). This algorithm [11] maintains a reservoir with a predefined sample size \hat{M} . Firstly the reservoir is filled with the initial edges from the stream. Every edge after that is computed for the probability \hat{M}/i of being inserted. Where i is the length of the stream exhausted till then. If the probability of the contending edge in the stream is greater than the probability of an edge in the reservoir which is $1/i$, then uniformly at random an edge is picked from the reservoir. The picked edge is replaced with the edge in the stream. In case the probability is less the streaming edge is discarded. The weights of edges are given by multi-graph in the reservoir.

³ Note: The values of y-axis in the Fig. 4 should be increased by a constant c ($y\text{-axis} = y\text{-axis} + c$). For call networks $c = 0.999999$, facebook $c = 0.999$, and CollegeMsg $c = 0.99$

Table 2. Correlation of temporal distributions with aggregated network

True vs sample%		90	70	50	30	10
Call network	SBias	0.996	0.957	0.917	0.891	0.145
Facebook	RS	0.995	0.987	0.960	0.918	0.846
	BRS	0.996	0.988	0.967	0.925	0.850
	SBias	0.997	0.993	0.984	0.975	0.967
CollegeMsg	RS	0.987	0.953	0.915	0.866	0.745
	BRS	0.986	0.953	0.895	0.792	0.442
	SBias	0.997	0.974	0.921	0.783	0.156

Biased Random Sampling for Edge Stream (BRS). This algorithm is proposed as a simple variant of RS in [9]. It is actually an unbiased random sampling technique but considering the above algorithm as a standard, this is its biased version. Unlike the above algorithm where the probability of streaming edges diminishes as the stream progresses, this algorithm ensures every edge goes into the reservoir. An edge from the reservoir is chosen for replacement at random. Therefore the edge insertion is deterministic but deletion is probabilistic. An edge staying for a long time in the reservoir has the same probability of getting out as an edge inserted recently. Consequently, the edges in the reservoir are distributed randomly over time.

Though the above two algorithms are known to be fast and simple, we were only able to run them on two data sets from three. The algorithms were not able to handle the size of call network's samples.

Table 3. Correlation of temporal distributions with edges per τ

Edges separated/ τ vs sample%		1	3	5	10	30	50	70	90
Facebook	RS	0.65	0.75	0.79	0.86	0.89	0.91	0.90	0.91
	BRS	0.64	0.75	0.80	0.87	0.90	0.90	0.90	0.91
	SBias	0.99	0.96	0.96	0.96	0.95	0.94	0.92	0.91
CollegeMsg	RS	0.08	0.10	0.06	0.008	-0.17	-0.27	-0.39	-0.51
	BRS	0.46	0.52	0.49	0.33	-0.08	-0.25	-0.38	-0.52
	SBias	0.82	0.62	0.36	0.004	-0.36	-0.48	-0.52	-0.55

Distance Measure. We used two-sided Kalmogorov-Smirnov D-statistics to measure the distance of sample distributions of different methods with the true distribution. The distance in degree distributions is given in Fig. 5. We observe

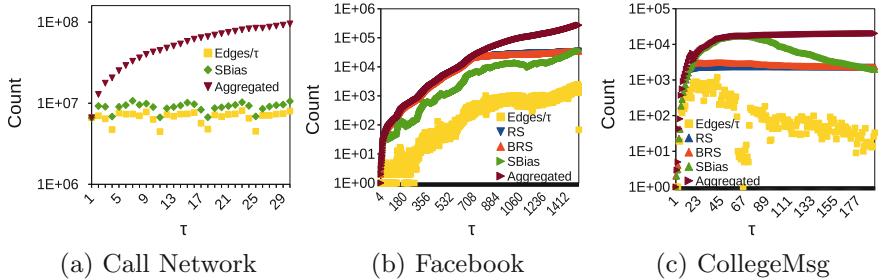


Fig. 6. Illustration of temporal distributions of dynamic samples of 10% in comparison to the # of unique edges separated per τ of the true network. Y-axis is plotted in log scale to clearly show the patterns in low scale together with high scale data.

that SBias gives better distributions. SBias gives a better distribution of cluster structures.

Low Degree Bias Rate. Usually samples from large networks are prone to low degree biases. Therefore we examined the bias to the nodes with degree one, in the samples compared to the true network which is shown Figs. 5(c) and (f). Bias rate shows the percent of one degree nodes in the network to other degree nodes. Y-axis in the Figs. 5(c) and (f) begins with the bias rate of the true network to show the comparative increase of bias in the samples. As the sample size decreases RS and BRS get more biased to low degree nodes.

Temporal Correlation. The temporal distribution of network size (# of unique edges per τ) exhibited by SBias for larger samples in the observational period are very similar to the true network distribution. The Pearson Correlation coefficients are shown in Table 2. The temporal distributions of the smaller samples are quite close to the distribution of individual edges per τ of the original network. Note, True network is the aggregated network, while edges/ τ is individual network separated per τ . A 90% sample is more correlated to true network temporal distribution while a 1% sample is more correlated to individual edges per τ temporal distribution. This property of our algorithm can help in identifying and predicting temporal trends, changes and behaviors. In Fig. 6 we see the patterns preserved by a 10% sample visually. The three data sets we choose have three different temporal distributions but the behavior of the sample is according to their respective true behavior. The Pearson Correlation between temporal distributions from the beginning to the end of the stream for different sample sizes is given in Table 3. An illustration of 10% sample is given in Fig. 6. RS and BRS do not follow the pattern of true network variations for both the data sets, as soon as the reservoir gets filled up they go flat (few deviations are seen in BRS) by maintaining the fixed sample size. When the true distribution behaves as a straight line they get a better score. They perform better on CollegeMsg network than Facebook because the size of the network is not growing exponentially over time as most other real-world networks do.

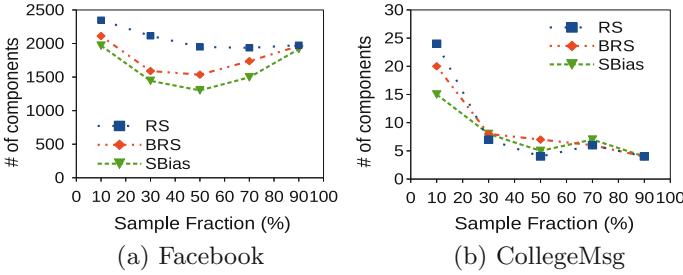


Fig. 7. Number of components in samples by three algorithms at the end of stream.

K-Components Components are the disconnected clusters in a graph. Graphs from the real world have many components (at least due to the limited availability of temporal data). Therefore, we compare the number of components from the samples of same size. Lesser the number of components is better. Bias to low degree nodes increases the number of components. We see that SBias gives better component structure compared to RS and BRS (Fig. 7).

5 Conclusions and Future Work

We proposed a one pass, memory less sampling algorithm for multi/weighted network streams to focus on the relationships and time variable. We empirically demonstrate that SBias preserves the degree distribution without biases and decreases redundancy in samples. It also maintains the patterns in temporal streams. In comparison to reservoir sampling and biased random sampling, we show that SBias preserves well the component structure. It is also able to handle the large network samples we generated from call network, which the other two algorithms were not able to process. We also proposed some measures for evaluating samples, like Krackhardt efficiency, temporal distributions, and stability of edges. For the future works, we would like to analyze the performance theoretically and mathematically and understand the limitations of SBias. We intend to compare our method to other state of the art methods for sampling streams by upgrading them to handle multi or weighted graphs, which will help us explore new properties. An interesting direction is also to estimate the properties of true network from the samples of SBias. Another appealing direction we would like to pursue is using SBias in an application specific way, for example, link prediction, as it gives bias to recent links and their strengths and, change detection, as it preserves temporal distributions.

Acknowledgement. This research was carried out in the framework of the project TEC4Growth – RL SMILES – Smart, mobile, Intelligent and Large Scale Sensing and analytics NORTE-01-0145-FEDER-000020 which is financed by the north Portugal regional operational program NORTE 2020 and partly funded by PRODEI program, Faculdade de Engenharia, Universidade do Porto.

References

1. Aggarwal, C.C.: On biased reservoir sampling in the presence of stream evolution. In: Proceedings of the 32nd International Conference on Very Large Data Bases, pp. 607–618. VLDB Endowment (2006)
2. Ahmed, N.K., Duffield, N., Willke, T.L., Rossi, R.A.: On sampling from massive graph streams. Proc. VLDB Endow. **10**(11), 1430–1441 (2017)
3. Ahmed, N.K., Neville, J., Kompella, R.: Network sampling: from static to streaming graphs. ACM Trans. Knowl. Discov. Data (TKDD) **8**(2), 7 (2014)
4. Chen, H.H., Miller, D.J., Giles, C.L.: The predictive value of young and old links in a social network. In: Proceedings of the ACM SIGMOD Workshop on Databases and Social Networks, pp. 43–48. ACM (2013)
5. Krackhardt, D.: Graph theoretical dimensions of informal organizations. In: Computational Organization Theory, pp. 107–130. Psychology Press (2014)
6. Panzarasa, P., Opsahl, T., Carley, K.M.: Patterns and dynamics of users' behavior and interaction: network analysis of an online community. J. Am. Soc. Inf. Sci. Technol. **60**(5), 911–932 (2009)
7. Papagelis, M., Das, G., Koudas, N.: Sampling online social networks. IEEE Trans. Knowl. Data Eng. **25**(3), 662–676 (2013)
8. Tabassum, S., Gama, J.: Sampling evolving ego-networks with forgetting factor (2016)
9. Tabassum, S., Gama, J.: Sampling massive streaming call graphs. In: ACM Symposium on Advanced Computing, pp. 923–928 (2016)
10. Viswanath, B., Mislove, A., Cha, M., Gummadi, K.P.: On the evolution of user interaction in facebook. In: Proceedings of the 2nd ACM Workshop on Online Social Networks, pp. 37–42. ACM (2009)
11. Vitter, J.S.: Random sampling with a reservoir. ACM Trans. Math. Softw. (TOMS) **11**(1), 37–57 (1985)
12. Zhang, J., Zhu, K., Pei, Y., Fletcher, G., Pechenizkiy, M.: Clustering-structure representative sampling from graph streams. In: International Workshop on Complex Networks and Their Applications, pp. 265–277. Springer (2017)



Using Network Reliability to Understand International Food Trade Dynamics

Madhurima Nath^{1(✉)}, Srinivasan Venkatramanan¹, Bryan Kaperick¹, Stephen Eubank², Madhav V. Marathe², Achla Marathe², and Abhijin Adiga¹

¹ Virginia Tech, Blacksburg, VA 24060, USA

{mmath,vsriniv,bryanjk,abhijin}@vt.edu

² University of Virginia, Charlottesville, VA 22904, USA

{eubank,mvm7hz,avm6bk}@virginia.edu

Abstract. Understanding the structural and dynamical properties of food networks is critical for food security and social welfare. Here, we analyze international trade networks corresponding to four solanaceous crops obtained using the Food and Agricultural Organization trade database using Moore-Shannon network reliability. We present a novel approach to identify important dynamics-induced clusters of highly-connected nodes in a directed weighted network. Our analysis shows that the structure and dynamics can greatly vary across commodities. However, a consistent pattern that we observe in these commodity-specific networks is that almost all clusters that are formed are between adjacent countries in regions where liberal bilateral trade relations exist. Our analysis of networks of different years shows that intensification of trade has led to increased size of clusters, which implies that the number of countries spared from the network effects of disruption is reducing. Finally, applying this method to the aggregate network obtained by combining the four networks reveals clusters very different from those found in the constituent networks.

Keywords: Network reliability · Food networks · Dynamics on networks · Contagion clusters

1 Introduction

The global food system is characterized by concentrated and specialized agricultural production, and an ever increasing reliance on the long-distance trade of these commodities. This makes it vulnerable to extreme events such as climate change [16], invasive species introductions [9] and contamination [5]. Therefore, understanding the structural and dynamical properties of food trade networks is critical to ensure food security, health and economic stability.

Here, we study international agricultural commodity networks, where the network consists of countries as nodes and directed weighted edges representing the volume of export from the source to the destination. In particular, we analyze a set of commodity-specific networks comprising of crops belonging to the

Solanaceae family, which includes tomato, eggplant, potato and peppers over a span of 10 years. These networks are constructed from the Food and Agricultural Organization (FAO) trade matrix database [6]. Previous works [1, 5, 17], have typically studied networks aggregated over multiple commodities. However, there are several applications where it is important to consider commodity-specific trade.

Our choice of networks of solanaceous crops is motivated by the recent global invasion of the pest, South American tomato leafminer or *Tuta absoluta* [2, 3]. Within a decade of its accidental introduction to Spain from its native habitat in South America, this moth has spread to most parts of Europe and Africa, West, Central and South Asia. There is overwhelming evidence that it has spread to extensive trade between countries in these regions. While its primary target is tomato, it can attack and survive on other solanaceous crops such as the ones mentioned above. In general, there is an emerging trend to account for trade and other human-mediated pathways while modeling invasive species spread [8]. Besides, independent of this application there is merit in analyzing flows of these crops; tomato and potato are among the top traded essential vegetables. Disruptions in their production or supply can have enormous socio-economic impact. This analysis will help identify the vulnerabilities in these networks and help risk analysts make informed decisions in mitigating contagion processes over these networks.

Our Contributions. In this paper, we develop a novel method to analyze the dynamical properties of the above-mentioned networks. Through the lens of a discrete-time SIR process coupled with network reliability principles [12, 20], we identify critical subsets of nodes in the networks, which we refer to as *contagion clusters* (formal definitions are provided later). Broadly, these are well-connected maximal subsets of nodes such that if the infection is introduced to the cluster, it spreads rapidly within the cluster. We assess their vulnerability by computing the minimum (weight) cuts and total volumes of their imports and exports with the rest of the network. Further, we compare the networks based on their contagion clusters: How do the networks and therefore the contagion clusters differ across communities? How have they evolved over time? How does an aggregated network differ from its constituent networks? Some of our results are as follows:

- We identified important contagion clusters in the food networks that are not only vulnerable to attacks, but can also spread the infection to a large number of vertices.
- We observe that the size and participating vertices of prominent contagion clusters can differ greatly across commodities. However, all of these clusters tend to be region-specific, e.g., countries localized to Europe or North America.
- Our analysis of the tomato networks over a period of 10 years reveals that intensification of trade has led to increase in both volume traded within contagion clusters and their connectivity outside. Moreover, the size of top contagion clusters has increased over the years.

- Some contagion clusters in the aggregate network span continents, unlike those in the commodity-specific networks.

Related Work. A number of works have analyzed world trade networks constructed from UN datasets. Serrano and Boguná [17] analyze the structural properties of the World Trade Web obtained by aggregated trade statistics from the ComTrade dataset. Baskaran et al. [1] study the evolution of networks induced by different product groups over time. Perhaps, the work that is closest to ours is by Ercsey et al. [5]. In this paper, an international food trade network is constructed by aggregating product codes corresponding to food. They perform structural analysis based on weighted betweenness centrality and graph density, and dynamical analysis using a diffusion model to assess the complexity and vulnerabilities of the network. There are two key differences between the above-mentioned works and ours. Unlike the FAOSTAT trade matrix, ComTrade reports the value of goods exported from one country to another in US dollars. Secondly, the network is obtained by aggregating categories corresponding to food products. Suweis et al. [18] use production and trade data from FAO and demographics information to assess the resilience and the reactivity of the coupled population food system. To this end, they develop a diffusion model referred to as food-demographic delayed logistic model. Again, they use aggregated trade data for network construction.

Agricultural commodity networks are being increasingly considered in specific application domains such as invasive species spread, bio-warfare and transportation. Nopsa et al. [8] study rail networks of grain transport in the US and Australia in the context of pathogen and mycotoxin spread. They identify key nodes in the network to monitor or quarantine so that the spread is mitigated. Venkatramanan et al. [19] model the domestic seasonal vegetable trade network in Nepal and show that spread of invasive species is correlated with trade flows of host crops. Robinson et al. [15] consider the network of US food flows, and use a linear programming approach to reconfigure the flows to minimize total miles in the network.

Community detection in unweighted, undirected networks has been studied extensively using algorithms based on modularity [13], q-state Potts model [14], extremal optimization of modularity [4]. Malliaros et al. [10] have explored clustering for directed networks. Ghosh et al. [7] have examined the dependence of a dynamic process on the structure of the network to identify important vertices participating in the dynamics and communities in the network.

2 Network Construction

The international food trade networks that we study in this paper are obtained from Food and Agriculture Organization (FAO) [6]. Especially, we obtained the Detailed Trade Matrices at the country level for each of the crops (Tomatoes, Potatoes, Eggplants and Peppers) and years (2005–2013) of interest. We also focused on the *Quantity* of trade (measured in tonnes) and not the associated monetary *Value* (measured in US dollars). The raw data is structured as follows

for each year Y and crop C : Given a pair of countries (r, p) (stands for *Reporter* and *Partner*), we have two quantities namely $E_{r,p}$ and $I_{r,p}$. $E_{r,p}$ is the quantity of the commodity being exported by r to p as reported by r . Likewise, $I_{r,p}$ is the quantity of the commodity being imported by r from p , as reported by r .

We note that there are reporting discrepancies in the raw data, i.e., $E_{r,p} \neq I_{p,r}$ for some pairs of (r, p) . So we construct a *maximal* trade matrix whose edge weights are given by $W_{i,j}$ as

$$W_{i,j} = \max(E_{i,j}, I_{j,i}).$$

In the context of food security and invasion risk, the maximal trade matrix can be considered as the worst-case scenario.

3 Network Reliability and Contagion Clusters

Network reliability was originally proposed by Moore and Shannon [11] to understand the reliability of electrical circuits with unreliable individual components. Recently [12, 20], this concept has been adapted to understand the likelihood of certain events (aka *properties*) occurring in graph-based dynamical systems. We introduce basic terminology and associated notations to describe the concepts and our algorithm to discover contagion clusters in this section. A more detailed discussion of the concepts of network reliability can be found in [12, 20].

Given a graph $\mathcal{G} = (V, E)$, and dynamics \mathcal{D} , reliability $R(\mathcal{G}, \mathcal{D}, \mathcal{P})$ is used to denote the probability of property \mathcal{P} being satisfied by the dynamic process. Here, \mathcal{D} is the independent cascade epidemic process or the discrete-time Susceptible-Infected-Removed (SIR) process where an infected node remains in that state for one unit step before transitioning to R. One possible \mathcal{P} could be the property that at least α fraction of nodes are infected. Note that for contagion-like dynamics on graphs, a typical parameter associated with it is the probability of transmission through each edge x . When the property of interest is implicitly understood, the reliability (polynomial) is sometimes expressed as $R(x; \mathcal{G})$, and is a global measure which depends on both the network structure and the dynamics.

Independent cascade process can be viewed as the random subgraph obtained by sampling edges in \mathcal{G} independently with probabilities x , thus resulting in a contagion subgraph \mathcal{G}_x . \mathcal{G}_x may or may not satisfy the property \mathcal{P} of interest. Let *struts* be the minimal subgraphs that satisfy property \mathcal{P} . Likewise, let *cuts* be minimal subgraphs which when removed from \mathcal{G} destroy the property \mathcal{P} . Together, we denote them as *cruxes*.

Remark: Network reliability calculation for homogeneous edge probabilities is mathematically and computationally more amenable than for arbitrary edge weights. Hence for the purpose of computing $R(x; \mathcal{G})$, we convert the maximal trade matrix weights into parallel edges. Given a step size Δ , an edge of weight w between nodes i and j is converted into $\lfloor \frac{w}{\Delta} \rfloor$ simple edges between them. This also allows us to (a) quantize the flow volume to separate edges representing individuals shipments, (b) evaluate the impact of reducing edge weights, instead of

completely removing them and (c) simulate unweighted version of the dynamics where each simple edge has an associated probability of x . The resulting multi-graph \mathcal{G} is then the quantized representation of the maximal trade matrix W .

Contagion Clusters Extraction Algorithm. In \mathcal{G} , we are interested in collections of nodes $\{CC_l\}$ which have a shared vulnerability to the contagion, i.e., infection of a node $i \in CC_l$ significantly increases the likelihood of other nodes $j \in CC_l$ to be infected. Though similar to the concept of communities, note that there is an added emphasis on the dynamics. We refer to these as *contagion clusters*. The necessary characteristic of a contagion cluster is that it must induce a strongly connected component (SCC) in the network. In addition, the connectivity within the cluster should be strong enough for each vertex to have enough influence on the rest of the nodes in the cluster. We develop an algorithm called **COCLEA** (**CO**ntagion **CL**usters **E**xtraction **A**lgorithm) to identify these clusters using network reliability.

Definition: Given a network \mathcal{G} , the *contagion score* of a set $X \subseteq V$ is given as:

$$\psi(X) = \frac{1}{|X|} \sum_{i \in X} \sigma_X^{\mathcal{D}}(i)$$

where $\sigma_X^{\mathcal{D}}(i)$ is the expected number of nodes in X that get infected by dynamics \mathcal{D} , when we start by initially infecting i with transmission probability, x . A set X is a *contagion cluster* of size n , if it maximizes $\psi(X)$ for all $X \subseteq V$ with $|X| = n$.

The algorithm for extracting mutually exclusive contagion clusters (or potential candidates) of size $\leq n$ is described in Algorithm 1. The property \mathcal{P} of interest is that there exists no strongly connected component (SCC) with $|SCC| > n$, where n is a parameter that governs maximum contagion cluster size. The edge ranking algorithm scores each edge by its marginal contribution to the overall reliability. The contribution of an edge e is estimated as follows: Draw a random sample of cruxes $K = \{k\}$, and identify the subset of cruxes that do not contain e , i.e., $K'_e = \{k : e \notin k\}$. The inverse of the probability that at least one of these cruxes occur in a random subgraph of \mathcal{G} is the score $\eta_x(e)$. Let $\max |SCC|$ be the size of the largest strongly connected component in \mathcal{G} . When $\max |SCC| > n$, edges are removed sequentially by importance until there exists no SCC with $|SCC| > n$. If $\max |SCC| \leq n$, the algorithm terminates and returns all contagion clusters at that stage. Note that in the process of detecting such clusters, we also rank the edges in \mathcal{G} in terms of their importance for the contagion process.

Remark on edge score: The more likely it is that a random edge would show up in a certain set of cruxes, the less likely it is that an edge that shows up in that set is important. Hence an edge present in a single small crux is more important than an edge that is present in a single larger crux. Similarly, an edge that is present in many cruxes is more important than one that is present in only a few (if their sizes are all the same).

Algorithm 1: Contagion Clusters Extraction Algorithm (COCLEA)

```

Result: edge rankings  $ER$ ; contagion clusters  $\{CC_i\}$ 
1  $\mathcal{G} = (V, E)$ ;  $ER = []$ ;
2 while  $\mathcal{G}$  does not satisfy  $\mathcal{P}$  do
3   Randomly sample cruxes  $K = \{k\}$ ;
4   For every  $e \in E(\mathcal{G})$ ,  $\eta_x(e) = \mathbb{P}[\exists k' \in K'_e : k' \subseteq \mathcal{G}_x]^{-1}$ ;
5    $e^* = \arg \max_e \eta_x(e)$ ;
6    $ER = ER \cup \{e^*\}$ ;
7    $E(\mathcal{G}) = E(\mathcal{G}) \setminus e^*$ ;
8 end
9  $\{CC_i\}$  = strongly connected components in  $\mathcal{G}$ 

```

4 Experimental Results

The clustering algorithm was applied to study annual trade networks corresponding to four major crops of the *Solanaceae* family: tomato, potato, eggplant and pepper between 2005 and 2013 (Table 1). We discovered the contagion clusters for each of these networks, the prominent ones are listed in Table 2. In this section, we study the properties of these clusters and compare them across commodities and years. For the three input parameters of COCLEA, we used the following values: cluster size ($n = 2, 5, 10, 20, 30, |V|$), step size ($\Delta = 1000$) and probability ($p = 0.01, 0.1, 0.5$). Also, throughout this section, for brevity, we refer to countries by their ISO-3 code.

Table 1. Networks and their properties.

Network	Nodes	Edges	Volume (Kilo Ton.)	Network	Nodes	Edges	Volume (Kilo Ton.)
Tom05	186	1198	5355	Egg05	142	656	346
Tom07	189	1310	7360	Egg07	149	715	440
Tom09	188	1200	7490	Egg09	146	681	458
Tom11	192	1249	7822	Egg11	146	739	490
Tom13	189	1305	8247	Egg13	160	776	487
Pot05	206	1561	10156	Pep05	184	1809	330
Pot07	204	1780	12417	Pep07	188	1973	350
Pot09	204	1657	11659	Pep09	183	1804	348
Pot11	205	1778	14239	Pep11	183	1912	367
Pot13	205	1834	14575	Pep13	192	2063	412

To quantify the structural properties of the clusters and compare one another we computed different metrics for each mined cluster. This is summarized in Table 2 for some of the chosen clusters. To assess the strength of connectivity

Table 2. Prominent/interesting contagion clusters. All volumes are in Kilo Tonnes. Regarding network names, “NA” stands for North America.

Cluster name	Countries	Network	Num. edges	Vol.	Min. wt. cut	Min. cut	Inc. nbrs	Inc. vol.	Outg. nbrs	Outg. vol.
EU-T-13	PRT, NLD, BEL, FRA, ESP	Tom13	20	691	32	3	47	421	78	2142
EU-T-11	”	Tom11	20	705	44	3	46	400	68	1949
EU-T-09	”	Tom09	21	658	31	3	41	447	69	1755
EU-T-07	”	Tom07	20	646	28	2	48	371	70	1650
EU-T-05	”	Tom05	20	580	23	2	52	309	90	1567
NA-T-13	USA, MEX, CAN	Tom13	6	1955	15	1	23	21	43	12
NA-T-11	”	Tom11	6	1858	23	1	28	25	36	9
NA-T-09	”	Tom09	6	1604	57	1	25	17	35	4
NA-T-07	”	Tom07	6	1522	48	1	22	13	43	8
NA-T-05	”	Tom05	6	1265	14	1	26	13	33	7
C1-P-13	FRA, BEL, NLD, DEU, ISR	Pot13	20	4913	53	4	43	385	145	3773
C2-P-13	FRA, EGY, BEL, ITA, DEU, DNK, GBR, ESP, NLD, ISR	Pot13	87	7602	53	6	52	249	147	248
SE-P-13	MYS, SGP, IDN	Pot13	6	14	2	2	28	290	13	2
SA-P-13	LKA, IND, VNM	Pep13	5	17	1	1	35	19	122	166

among cluster vertices, we computed minimum weighted cut and the minimum cut of the cluster by applying the max-flow min-cut algorithm on the weighted and unweighted network induced by the cluster. The minimum weighted cut gives the minimum total volume of trade that needs to be reduced in order to disconnect the cluster, i.e., it will no longer be a strongly connected component. The minimum cut gives the minimum number of edges that need to be removed to disconnect the cluster. To assess a cluster’s role in the bigger network that it is part of, we also looked at the total volume of import into the cluster (*incoming volume*) and total volume of export from the cluster (*outgoing volume*). The former is an indicator of vulnerability of the cluster to attacks, while the latter indicates the influence the cluster has on the rest of the network. We also computed the number of distinct *in-neighbors* and *out-neighbors* of the cluster. An in-neighbor (out-neighbor) is a node which does not belong to the cluster and has an outgoing (incoming) edge to (from) the cluster.

Contagion Clusters in Commodity Specific Networks. Figure 1 show the contagion clusters for tomato and potato trade networks when $\Delta = 1000$ and n is restricted to 20. The colored clusters denote the contagion clusters obtained using COCLEA. The remaining nodes, which do not fall into any of these clusters are to the left, not shown due to lack of space. The colored edges connect countries belonging to corresponding colored cluster.

We observe that for tomato networks, the European countries form a strong cluster and is separated from others. However, in the case of potatoes, the European countries have strong trading relations with both African and West Asian

countries (C1-P-13 and C2-P-13 for example). The North American countries (NA-T-13) are fairly isolated from the rest of the world. Among the European countries, the core countries correspond to EU-T-13 (Table 2). The European clusters, unlike the North American cluster have lot more incoming and outgoing neighbors as well as volume. This indicates that not only is this cluster more vulnerable to attacks, it is also a big hub and can lead to rapid spread of contagion to other several countries. Another interesting difference between tomato and potato networks is the existence of several smaller clusters compared to tomato networks (e.g., SE-P-13).

In general, the eggplant network is very similar to that of the tomato network. However, the volumes of trade are much small. While the EU-T-13 cluster appears here, among the North American countries, only the cluster (USA, CAN) appears. The most distinct of all networks is that of peppers (Pep13). It has only one cluster (SA-P-13).

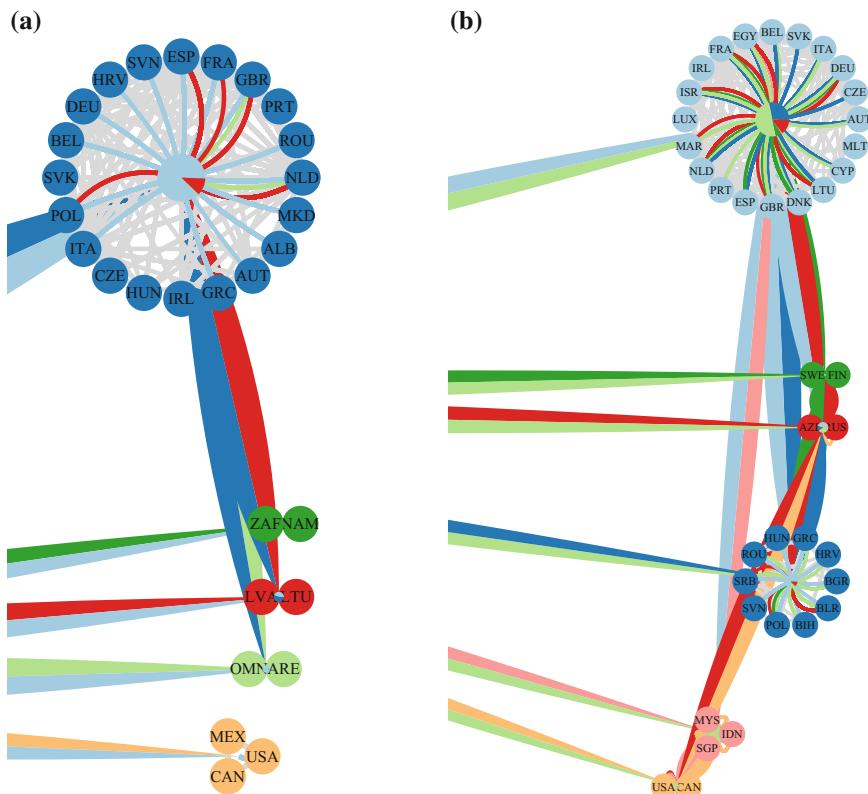


Fig. 1. Clusters for (a) Tomato trade network (b) Potato trade network, where $\Delta = 1000$ and $n = 20$. The colored clusters on the right of both the figures denote the contagion clusters. The rest of the graph is to the left and cannot be seen in this figure in the interest of zooming in on the clusters. The colored edges connect countries belonging to corresponding colored cluster.

Effect of Cluster Size. Figure 2 shows how the largest SCC in a network decomposes into smaller clusters as we decrease the parameter n . Consider network Tom13 quantized with $\Delta = 1000$. The last row represents the largest strongly connected component obtained for all $n > 21$, and consists solely of European countries. As we decrease n , note that smaller clusters such as (SVN, HRV), (CZE, SVK), (HUN, ROU) separate out at fairly early stages. Also countries like GRC and BGR drop out of any strongly connected components (shown in grey). For smaller n , we see the emergence of sub-clusters among Western European nations such as (GBR, IRL) and the (ESP, PRT), etc. Alternatively for potato network with $\Delta = 10000$, we see emergence of inter-continent clusters such as (EGY, GBR), and the presence of ISR in the largest SCC for most values of n .

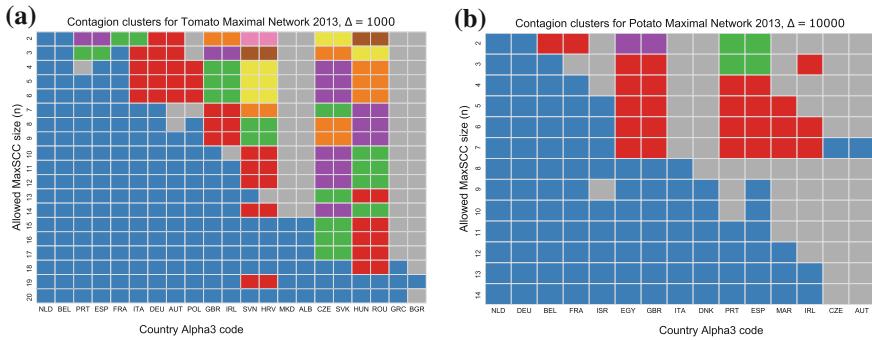


Fig. 2. Clusters for (a) Tomato trade network with $\Delta = 1000$ and (b) Potato trade network with $\Delta = 10000$ and $n = 20$.

Network Evolution Over Time. We studied the evolution of the cluster structure over time. Due to lack of space, we only present the results for the tomato networks. Overall, some clusters have remained stable over time. Two of them that stand out are the core European cluster (EU-T-*) and the North American triple (NA-T-*) (see Table 2). However, trade within these clusters has intensified over the years. Also, both the incoming and outgoing volume have increased considerably. While the number of countries to which these countries export has increased in the case of NA-T-*, the number of countries from which this cluster imports has generally remained same. Other clusters which appear for all years are (NZL, AUS) and (ARE, OMN). In the recent years, new clusters have formed (THA, MYS) in Southeast Asia and (ZAF, NAM) in Southern Africa. In fact, both these clusters export to a large number of countries (43 and 23 respectively). There are some clusters (SYR, JOR, LBN, SAU) which appear only in 2007 and earlier networks. Perhaps, political conditions have led to a decrease or pause in trade between them.

The Effect of Aggregation. While in the commodity-specific networks, all major clusters that were discovered were mostly region-specific, in the aggre-

gated network (All13), we observed the formation of clusters spanning multiple continents. When we allowed the clusters to be of any size, we observed one big cluster of size 69. When we limited it to 20 and below, interesting patterns emerged. Perhaps the most extreme of them all is (GTM, CAN, MEX, IND, VNM, ZAF, IDN, USA, NAM) spanning Asia, North America and Africa. A more regional phenomenon that we observe is the breaking up of European countries into two clusters. One cluster consists of EU-T-13 and other European countries merged with some of the North African and West Asian countries such as Israel. The other cluster consists of East European countries and some Arab countries. Clearly, this indicates that if a pest or pathogen is capable of spreading through trade of multiple commodities, then, the expansion can be very rapid and unpredictable. Fortunately, there is no evidence of *T. absoluta* having spread through trade of any crop other than tomato. This study also suggests that one should be cautious when aggregating commodities and interpreting the resulting networks.

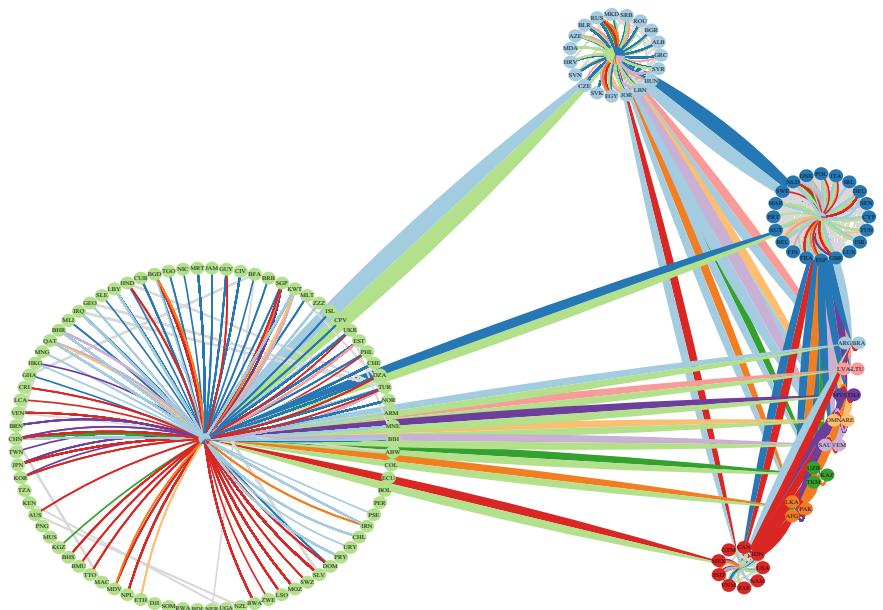


Fig. 3. Clusters for All13 network with $\Delta = 1000$ and $n = 20$. The colored clusters on the right of both the figures denote the contagion clusters. The big cluster on the left represent countries which do not belong to any such SCC. There are 2 big clusters on the right, containing 19 and 20, containing European, African and Middle East Asian countries. The colored edges connect countries belonging to corresponding colored cluster.

Cluster Size, Step size, Probability. Cluster size plays a critical role in discovering important contagion clusters. We have repeated our analysis for step size, $\Delta = 500, 1000$ and 10000 . We observed that as Δ increases, stronger communities remain intact, even though lesser countries are present in the network. When we ran our algorithm with transmission probability, $x = 0.01, 0.1, 0.5$ and restricted the cluster size to be 20 for Tom13, we obtained the same results for the ranked edges and the clusters. This suggests that the clusters are robust to change in probability of transmission.

5 Conclusion and Future Directions

We analyzed international agricultural commodity networks using the Moore-Shannon network reliability. Under the assumption of a discrete-time SIR diffusion process, we proposed a method to discover highly-connected subsets of vertices, which we refer to as contagion clusters. The method was applied to international trade networks of four crops. We studied the difference in structure of the networks across commodities and across years. We identified important subsets of vertices which can play a key role in the contagion process. There are several obvious directions to extend this work. One important line of work is to study how the cluster structure varies with the underlying diffusion process, such as an SI, SIS model or any other model of the phenomenon being studied. Our multi-graph representation implicitly models the relationship between flow volumes (edge weights) and the infection probability. It would be interesting to know how different functions of edge weights affect mined clusters. In general, a direct implementation of computing edge rankings for weighted graphs will be a significant methodological contribution. Summarizing, the network reliability based methods provide a powerful tool to discover important dynamics-related properties of networks.

Acknowledgments. This work was supported in part by the United States Agency for International Development under the Cooperative Agreement NO. AID-OAA-L-15-00001 Feed the Future Innovation Lab for Integrated Pest Management, DTRA CNIMS Contract HDTRA1-11-D-0016-0001, NSF BIG DATA Grant IIS-1633028, NSF DIBBS Grant ACI-1443054, NIH Grant 1R01GM109718 and NSF NRT-DESE Grant DGE-154362.

References

1. Baskaran, T., Blöchl, F., Brück, T., Theis, F.J.: The Heckscher-Ohlin model and the network structure of international trade. *Int. Rev. Econ. Financ.* **20**(2), 135–145 (2011)
2. Biondi, A., Guedes, R.N.C., Wan, F.H., Desneux, N.: Ecology, worldwide spread, and management of the invasive south american tomato pinworm, *Tuta absoluta*: past, present, and future. *Annu. Rev. Entomol.* **63**, 239–258 (2018)
3. Campos, M.R., Biondi, A., Adiga, A., Guedes, R.N., Desneux, N.: From the western palaearctic region to beyond: *Tuta absoluta* 10 years after invading europe. *J. Pest Sci.* **90**(3), 787–796 (2017)

4. Duch, J., Arenas, A.: Community detection in complex networks using extremal optimization. *Phys. Rev. E* **72**(2), 027,104 (2005)
5. Ercsey-Ravasz, M., Toroczkai, Z., Lakner, Z., Baranyi, J.: Complexity of the international agro-food trade network and its impact on food safety. *PloS One* **7**(5), e37,810 (2012)
6. FAO: Production and trade. <http://www.fao.org/faostat/en/#data> (2016)
7. Ghosh, R., Teng, S.H., Lerman, K., Yan, X.: The interplay between dynamics and networks: centrality, communities, and cheeger inequality. In: Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 1406–1415. ACM (2014)
8. Hernandez Nopsa, J.F., et al.: Ecological networks in stored grain: key postharvest nodes for emerging pests, pathogens, and mycotoxins. *BioScience* **65**(10), 985–1002 (2015)
9. Hulme, P.E.: Trade, transport and trouble: managing invasive species pathways in an era of globalization. *J. Appl. Ecol.* **46**(1), 10–18 (2009)
10. Malliaros, F.D., Vazirgiannis, M.: Clustering and community detection in directed networks: a survey. *Phys. Rep.* **533**(4), 95–142 (2013)
11. Moore, E.F., Shannon, C.E.: Reliable circuits using less reliable relays. *J. Frankl. Inst.* **262**(3), 191–208 (1956)
12. Nath, M., Ren, Y., Khorramzadeh, Y., Eubank, S.: Determining whether a class of random graphs is consistent with an observed contact network. *J. Theor. Biol.* **440**, 121–132 (2018). <https://doi.org/10.1016/j.jtbi.2017.12.021>, <http://www.sciencedirect.com/science/article/pii/S002251931730560X>
13. Newman, M.E.: Modularity and community structure in networks. *Proc. Natl. Acad. Sci.* **103**(23), 8577–8582 (2006)
14. Reichardt, J., Bornholdt, S.: Detecting fuzzy community structures in complex networks with a Potts model. *Phys. Rev. Lett.* **93**(21), 218,701 (2004)
15. Robinson, C., Shirazi, A., Liu, M., Dilkina, B.: Network optimization of food flows in the US. In: 2016 IEEE International Conference on Big Data (Big Data), pp. 2190–2198. IEEE (2016)
16. Rosenzweig, C., Parry, M.L., et al.: Potential impact of climate change on world food supply. *Nature* **367**(6459), 133–138 (1994)
17. Serrano, M.A., Boguná, M.: Topology of the world trade web. *Phys. Rev. E* **68**(1), 015,101 (2003)
18. Suweis, S., Carr, J.A., Maritan, A., Rinaldo, A., D’Odorico, P.: Resilience and reactivity of global food security, p. 201507366 (2015)
19. Venkatramanan, S., et al.: Towards robust models of food flows and their role in invasive species spread. In: 2017 IEEE International Conference on Big Data (Big Data), pp. 435–444. IEEE (2017)
20. Youssef, M., Khorramzadeh, Y., Eubank, S.: Network reliability: the effect of local network structure on diffusive processes. *Phys. Rev. E* **88**(5), 052,810 (2013)



An Approach to Structural Analysis Using Moore-Shannon Network Reliability

Madhurima Nath^{1(✉)}, Yihui Ren², and Stephen Eubank³

¹ Department of Physics, Virginia Tech, Blacksburg, VA, USA
mnath@vt.edu

² Brookhaven National Lab, Upton, NY, USA
yren@bnl.gov

³ University of Virginia, Charlottesville, VA, USA
eubank@virginia.edu

Abstract. We develop a formalism for addressing many common questions about network structure. The formalism is based on Shannon-Moore network reliability and on the Birnbaum definition of importance in terms of reliability. The computational methods we suggest based on this formalism are designed to answer specific questions about how network structure affects particular aspects of the dynamics, and can be applied to a wide variety of dynamical systems on interaction networks. The formalism and methods bring well-understood analytical techniques from statistical physics and computer science to bear on current issues in network science. We introduce the concepts that underpin the formalism, explore methods of applying these concepts, and demonstrate applications to problems such as edge ranking and graph reduction.

Keywords: Network reliability · Birnbaum importance · Network structure · Dynamics on networks

1 Introduction

Suppose we are given a communication network with individual links that fail at known rates and asked which links are most important. Concretely, suppose we can replace a fixed number, b , of the links with more reliable ones. Which ones should we replace? It depends. Obviously, it depends on the structure of the network, the dynamics of communication and our goal – e.g., whether we are trying to broadcast a message or send it to a specific node; less obviously, it depends on the failure rate; and somewhat surprisingly, it can depend on b . We develop the formalism that includes these dependencies and propose methods for approximating solutions to this and related problems that apply to many kinds of dynamics, networks, and goals, with boundable error that is a function of their (controllable) time complexity.

We are interested in systems of discrete nodes whose state changes according to their interactions with neighbors on a network. The network may be directed, weighted, multiplexed, time-dependent, and/or composed of inhomogeneous nodes. The collection of states of all the nodes and weights on all the edges is called the system's *configuration*. The probability of visiting a particular set of system configurations \mathcal{C} in a time window τ for the network G is a natural generalization of *Moore-Shannon network reliability* $R(G)$ [7]. Network reliability is useful for summarizing the influence of network structure on dynamics, and can be used as a statistic to characterize families of networks [8]. The difference in reliability between G and a subnetwork $G \setminus g$, where g is a subset of edges and/or nodes, is known as g 's *Birnbaum importance* [2], $B(G, g)$. Birnbaum importance can replace partial derivatives in a stability analysis of the network dynamical system with respect to (inherently discrete) changes to network structure. Indeed, in the limit of infinitesimal changes in edge weights, Birnbaum importance *is* a partial derivative with respect to the network. The network reliability and Birnbaum importance depend on the choice of system configurations, the time window, and the interaction rules and their parameter values. Optimizing Birnbaum importance thus has a clear interpretation, viz., removing the k most important edges reduces the probability that a system operating under the given dynamics would enter any of a set of configurations \mathcal{C} more than removing any other combination of k edges.

Although evaluating $R(G)$ is known to be #P-complete in most cases [3], for some networks and some choices of configurations \mathcal{C} , it can be expressed in closed form; otherwise, simulations provide practically useful estimates [5, 9]. Each simulation run provides one sample of a Bernoulli process, like tossing a coin with bias $R(G)$. Unfortunately, the difference in importance between two edges in a large network is typically so small that Monte Carlo simulation is infeasible because too many samples are needed to obtain sufficient precision. However, as explained in Sect. 2, for a large class of problems, we can construct $R(G)$ from the probabilities of certain subnetworks that we call *cruxes*. Identifying a crux depends on determining whether any possible outcome of the dynamics is in \mathcal{C} , not making fine distinctions between the total probability of all those outcomes.

The Birnbaum importance of an edge takes into account the contributions of all other edges and thus depends on the network's global structure. Many problems require finding a set of k edges with the largest or smallest collective importance. In principle, exact optimization requires exhaustive comparison of all $\binom{E}{k}$ possible sets. A greedy approximation replaces exhaustive comparison with a step-by-step optimization, adding one edge to the set on each step. However, the approximation error of a greedy algorithm cannot be controlled well unless the optimization function is submodular. Birnbaum importance is not a submodular function of sets of edges. Fortunately, for important classes of reliability, it *is* a submodular function of cruxes.

2 Formalism

Although the formalism we present is general, we discuss it with reference to a particular dynamics, the independent cascade diffusion process. This process is also known as discrete-time Susceptible-Infectious-Removed (SIR) dynamics.

SIR dynamics. Under SIR dynamics, each node can be in one of three states: susceptible (S), infectious (I), or removed (R). A node in state I at time t will always be in state R at time $t + 1$; a node in state R remains there; a node in state I transmits infection to a node in state S with probability ρ . In our language, only a transmission counts as an “interaction”. If a susceptible node has several infectious neighbors, they transmit infection independently. That is, if there are n infectious neighbors, the total probability of transmission is $1 - (1 - \rho)^n$. Under SIR dynamics, any initial configuration will reach a fixed point consisting of nodes in states S or R only within V time steps, where V is the number of nodes in the network.

Network reliability. Network reliability is a function of dynamics D , a set of configurations \mathcal{C} , and a set of allowed initial configurations \mathcal{C}_0 , as well as the interaction network G . For ease of notation, we will usually leave D , \mathcal{C} , and \mathcal{C}_0 implicit. However, we will use two common choices of \mathcal{C} and \mathcal{C}_0 as illustrations:

1. For “Two-Terminal” reliability (R_{OD}), \mathcal{C} is the set of all configurations in which an origin node O and a destination node D are connected by a path of interactions. \mathcal{C}_0 includes only a single initial configuration, in which node O is perturbed away from its state in a configuration that is a fixed point of the dynamics. In statistical physics, R_{OD} is known as a *propagator*: the probability that a disturbance at O will propagate to D . Under SIR dynamics, R_{OD} is the probability that D will eventually become infected if node O is infected in a susceptible population.
2. For “N-terminal” reliability (R_N), \mathcal{C} is the set of all configurations in which at least N nodes are connected to a single origin by at least one path. \mathcal{C}_0 contains the same configurations as for R_{OD} , but for any choice of origin. Under SIR dynamics, R_N gives the probability that infecting a single, randomly chosen person in a susceptible population leads to an outbreak involving at least N people.

Trajectories, interactions, and active sets. A dynamical system assigns a probability to every *trajectory* – i.e., sequence of system configurations $\{C_0, C_1, \dots\}$ – by defining the conditional probability of every possible successor to any given configuration, $p(C_{t+1}|C_t)$. In a network dynamical system, the state of a node and its neighbors in C_t is a set of sufficient statistics for its state in C_{t+1} .

We define an *interaction* as a mapping from a set of at least two node states to a single node state. An interaction describes the change of state in one node between C_t and C_{t+1} , together with the states in C_t of the node(s) that caused the change. Under SIR dynamics, there is only one type of interaction, $\{S, I\} \rightarrow I$. Because simultaneous transmission from different sources happens independently, it can be decomposed into a set of pairwise interactions.

Under other dynamics, such as threshold dynamics, interactions among larger sets must be considered. Any interaction can be represented by a weighted, directed (hyper-)edge containing the set of interacting nodes, directed from the nodes causing the change to the node whose state changes. Its weight is the probability that the interaction occurs.

Because we observe only changes in configurations – i.e., nodes’ states – and not the interactions that caused them, the causal chain of interactions is ambiguous. Specifically, if several different sets of interactions can cause the same change in a single node’s state, it is impossible to infer which set led to the change. For example, consider a node, T , that, at time t , is susceptible with two infectious neighbors, A and B , and that becomes infectious at $t + 1$. Under SIR dynamics, there are three sets of interactions that can lead to this transition: T interacts only with A ; only with B ; or with both A and B . To understand the contribution any single interaction makes to the probability of a trajectory, it is crucial to take this ambiguity into account.

For any trajectory, the *active* interactions at time t are those in the union of all possible sets of interactions that could have led from C_t to C_{t+1} . Because any interaction can be represented as a (hyper-)edge, there is a 1-1 mapping between active interactions and subgraphs of the interaction network. Hence we will use the terms active interactions and subgraphs or subnetworks interchangeably.

Cruxes: minimal struts and cuts. For *monotonic* reliability criteria – those for which $g \subseteq g' \Leftrightarrow R(g) \leq R(g')$ – the reliability is completely determined by the probabilities of either of two sets of interactions that we call minimal cuts or struts. A *strut* is a subgraph g of G for which $R(g) > 0$; a *cut*, one for which $R(G \setminus g) = 0$. A *minimal strut* (resp., *cut*) is a strut (resp., cut) that has no subgraph that is a strut (resp., cut). Collectively, we refer to minimal struts and cuts as *cruxes*.

Birnbaum importance. The contribution of a crux to the reliability *in the context of all other cruxes* is its Birnbaum importance, i.e., the difference in reliability when the probabilities of all trajectories whose active interactions include the crux are adjusted to account for its absence. Because of the ambiguity noted above, the contribution of a set of cruxes is not necessarily the sum of their individual contributions, but it is bounded above by that sum. In particular, if A , B , and C are the only cruxes for reliability R in G and we let x represent the random event that crux X is in the active set of interactions in any instance of the dynamics, then

$$R(G) = p(a \vee b \vee c) \tag{1}$$

$$= p(a) + p(b) + p(c) - p(a \wedge b) - p(a \wedge c) - p(b \wedge c) + p(a \wedge b \wedge c). \tag{2}$$

Equation 2 is a consequence of unitarity as encoded in the Inclusion-Exclusion expansion. We can evaluate the Birnbaum importance of each crux or union of cruxes, e.g.:

$$B(G, A) = p(a) - p(a \wedge b) - p(a \wedge c) + p(a \wedge b \wedge c) \tag{3}$$

$$B(G, A \vee B) = p(c), \tag{4}$$

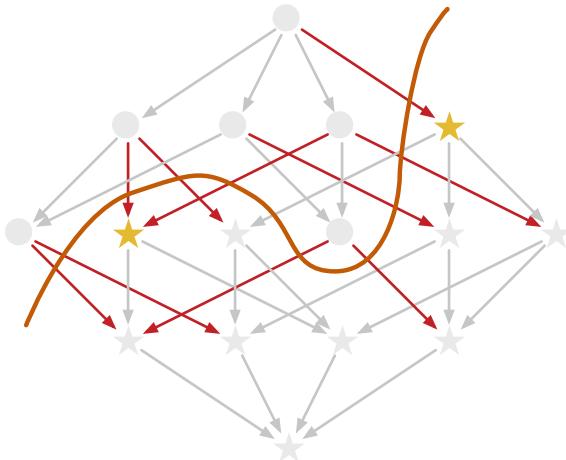


Fig. 1. Sketch of a partial order lattice for an interaction network with four edges under the subnetwork inclusion ordering. Stars represent struts, the sets of interactions that define trajectories satisfying the reliability constraints, i.e., leading to a configuration in \mathcal{C} ; circles represent cuts, sets of interactions that do not satisfy the constraints. Under a monotonic reliability rule there is a separatrix, shown here in red, between the stars and circles that is crossed by every directed path from top to bottom, e.g, by the red edges. Not every crossing edge ends in a crux, though, because the end node may not be minimal. Only the yellow stars are minimal struts.

and hence the difference in Birnbaum importance for any two cruxes:

$$B(G, A) - B(G, B) = p(a) - p(b) - p(a \wedge c) + p(b \wedge c). \quad (5)$$

3 Computational Methods

Finding cruxes. Cruxes can be identified using a binary search algorithm. First construct a (directed) partial order lattice of subgraphs of the interaction network, as illustrated in Fig. 1. The lattice can be arranged in layers labeled according to the number of edges in the subgraph. For a monotonic reliability, any path on the lattice from the empty graph at the top to G itself at the bottom is guaranteed to intersect exactly once the separatrix between struts and cuts. The strut below this intersection is not necessarily minimal, but if not, it must have an ancestor that is also a strut. Restart the search after replacing G with that ancestor. This algorithm must converge on a crux in at most E iterations of a binary search that typically has $\log E$ steps, where E is the number of edges in the interaction network. For certain forms of reliability and certain dynamics, there may be *ad-hoc* methods for finding cruxes that are more efficient, but this approach is general and embarrassingly parallel.

Greedy optimization. We use a greedy algorithm to find sets of cruxes with approximately maximum Birnbaum importance. Each step of the greedy approach adds the most important single crux to the set, removes it from the list

of all cruxes, and recomputes the importance for each of the remaining cruxes on the list. In practice, removing the most important crux from the list of all cruxes is accomplished by removing one of its interactions from the interaction network. However, removing a single interaction, e.g. edge (O, A) in Fig. 2, can remove more than one crux. The total decrease in reliability due to removing an edge is not simply the sum of the contributions of each crux it appears in. Indeed, it may be the case that removing an edge that is not in the top-ranked crux decreases the reliability more than removing one in the top-ranked crux, as illustrated in Sect. 4.

We illustrate the greedy algorithm by applying it to the network in Fig. 2. The Two-Terminal reliability of this network between nodes O and D under SIR dynamics is $R_{OD}(G, \rho) = \rho^2 + 2\rho^3 - 3\rho^5 + \rho^7$. If the shortcut, the top crux, is removed from the list of cruxes, the reliability decreases to $R_{OD}(G \setminus \chi_a, \rho) = 2\rho^3 - \rho^5$, so its Birnbaum importance is $B_{OD}(G, \chi_a, \rho) = \rho^2 - 2\rho^5 + \rho^7$. Repeating this computation for the configurations, cruxes, and edges listed in Fig. 2 gives the results in Table 1. Removing crux a is always a better choice than removing crux b or c , but removing the edge (O, A) removes both cruxes b and c at once. Notice that the relative importance of the most important edges depends on ρ . For this interaction network, the sign of the relative importance changes at the zero of $1 - 2\rho + \rho^3$, roughly $\rho_c \approx 0.618$.

Table 1. Birnbaum importance $R_{OD}(G, \dots, \rho)$ for the structures in Fig. 2, in the order they appear there.

Configuration	contribution	Crux	contribution
SSSR	$\rho^2(1 - \rho)$	$\{(O, B), (B, D)\}$	$2\rho^3 - 3\rho^5 + \rho^7$
RSSR	$\rho^3(1 - \rho)^2$	$\{(O, A), (A, A_1)(A_1, D)\}$	$\rho^3 - 2\rho^5 + \rho^7$
RRSR	$\rho^4(1 - \rho)(2 - \rho)$	$\{(O, A), (A, A_2), (A_2, D)\}$	$\rho^3 - 2\rho^5 + \rho^7$
RSRR	$\rho^4(1 - \rho)(2 - \rho)$		
RRRR	$\rho^5(3 - 3\rho + \rho^2)$	Edge	contribution
RRSS	$\rho^3(1 - \rho)^2$	(O, B)	$2\rho^3 - 3\rho^5 + \rho^7$
RSRS	$\rho^3(1 - \rho)^2$	(O, A)	$\rho^2 - 2\rho^5 + \rho^7$
RRRS	$\rho^4(1 - \rho)(2 - \rho)$	(A, A_1)	$\rho^3 - 2\rho^5 + \rho^7$

Cut/Strut Duality. There is a set covering relationship between minimal cuts and minimal struts. Specifically, every minimal cut is the union of at least one element from each minimal strut and vice versa. Thus, given an exhaustive list of minimal struts, we can construct all the minimal cuts and vice versa. To ensure minimality as we construct a cut, we put the list of minimal struts through a sieve, removing all those that contain each edge selected for the cut as it is selected.

The nature of minimal struts and cuts depends on the form of reliability, specifically on \mathcal{C}_0 and \mathcal{C} . For example, for Two-Terminal reliability, minimal

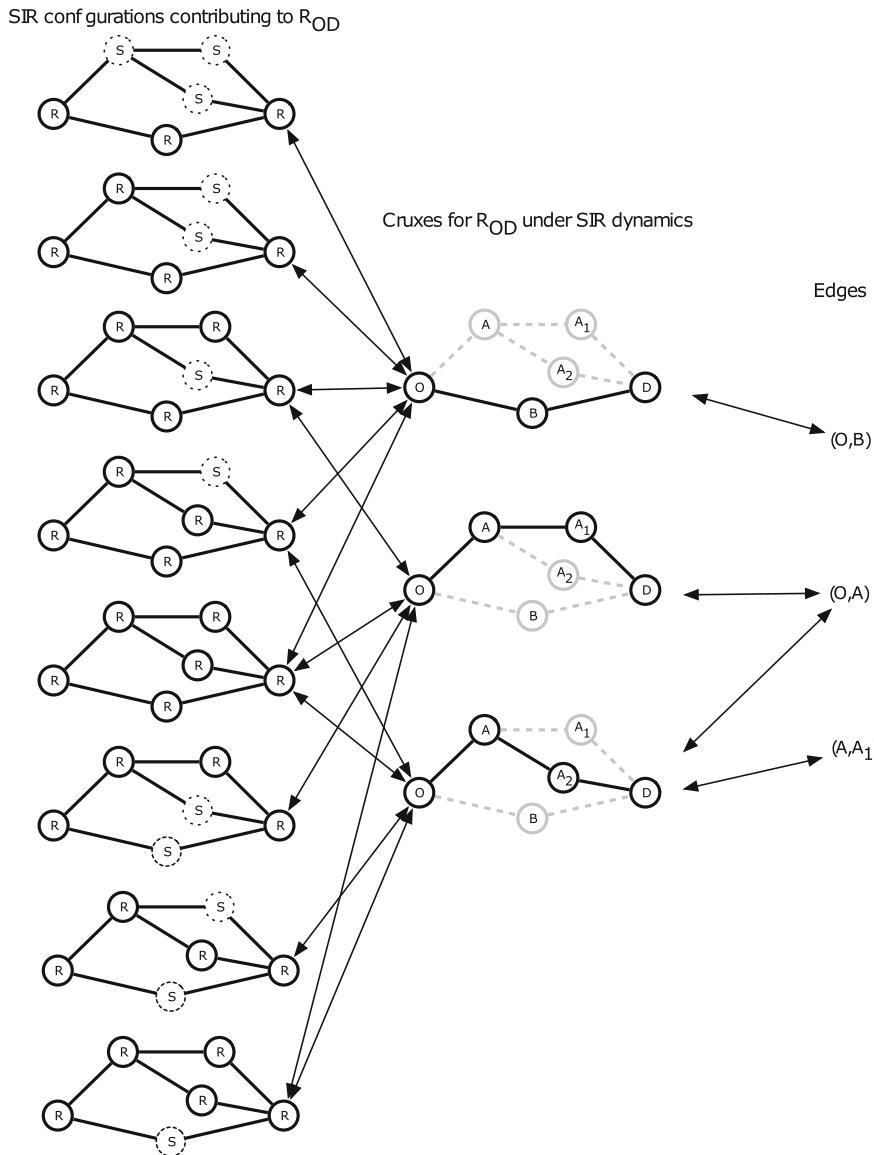


Fig. 2. Relationships among: (left) configurations that contribute to a Two-Terminal reliability under SIR dynamics in a toy network; (center) cruxes (minimal struts); and (right) individual interactions (edges). In each column, items are shown in decreasing order of their Birnbaum importance, which measures their contribution to the overall reliability in the context of everything else in the column. Ambiguity in inferring the sequence of interactions that produce a given configuration, indicated by configurations connected to multiple cruxes, creates an interaction among cruxes that, in turn, makes the reliability a nonlinear, but submodular, function of Birnbaum importance. Removing an interaction from the network often removes more than one crux, and the overall impact on the reliability is not trivial to compute.

struts are directed, acyclic paths from origin to destination and cuts are the usual cut sets. Note that in this case, minimal struts need not be *shortest* paths; only the smallest minimal struts are shortest paths. For N -Terminal reliability, minimal struts are trees with $N - 1$ edges, i.e., spanning trees for subsets of N nodes; all minimal struts have the same size. Minimal cuts in this case are sets of edges whose removal shatters the interaction network into connected components with size no greater than N .

Crux-based reliability estimation. Given a set of S cruxes, we can approximate network reliability by truncating the exact Inclusion-Exclusion expansion from 2^S terms to $O(\binom{S}{D})$ terms, including only combinations of up to D cruxes. The number of possible cruxes scales combinatorially with the size of a network, though, so we introduce a further approximation by sampling S cruxes. The two parameters S and D control both the accuracy and computational complexity of the estimate.

The greedy algorithm only compares the Birnbaum importance of edges pairwise. For edges e_i and e_j the difference $\Delta_{i,j} \equiv B(G, e_i) - B(G, e_j)$ reduces to $R(G \setminus e_j) - R(G \setminus e_i)$. If the set of cruxes in $G \setminus e_i$ is denoted χ_i , then $\Delta_{i,j}$ only depends on interactions among cruxes in $(\chi_i \cup \chi_j) \setminus (\chi_i \cap \chi_j)$. This drastic simplification, together with importance sampling of the cruxes and their submodularity, renders approximate solutions to optimization problems with bounded error computationally feasible, even for networks with hundreds of millions of edges.

Constructing networks from crux sets. The network reliability is entirely determined by the set of cruxes it contains and their probabilities. That is, if there is a mapping from $\{\mathcal{C}, G, D\}$ to $\{\mathcal{C}', G', D'\}$ that holds invariant the cruxes and their probabilities, then the reliabilities of the two systems will be identical. By choosing a particular \mathcal{C}' and D' , this mapping defines an equivalence relation between networks. For example, choosing Two-Terminal reliability under SIR dynamics we would construct a network equivalent to G as follows:

1. Create two nodes O and D in the new network.
2. Choose a minimal strut $u \subseteq G$.
3. Construct a linear chain in G' from O to D with $|u|$ edges.
4. Establish a 1-1 mapping from edges in u to edges in the chain.
5. Repeat until the list of minimal struts is empty:
 - (a) Remove u from the list of minimal struts and choose another one.
 - (b) Construct $|u|$ edges in a path from O to D as above, adding nodes as needed, except that any edge in u that has already appeared must be re-used.

Clearly, when $\mathcal{C} = \mathcal{C}'$ and $D = D'$, this procedure constructs a graph $G' \subseteq G$, where the missing parts do not contribute to the reliability. It remains to be seen how to ensure the probabilities of cruxes in the new dynamical system are consistent with those in the original system in the general case.

Replacing \mathcal{C} by its complement is equivalent to replacing R by $1 - R$ or, equivalently, treating the cuts as struts and vice versa. For example, the set of minimal cuts contained in a network G for Two-Terminal reliability under SIR

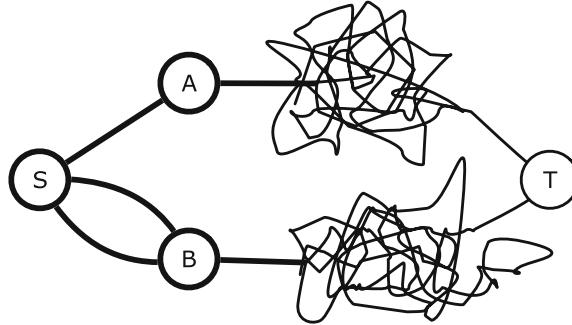


Fig. 3. Here, depending on the relative reliabilities of the network blobs and the individual edges, it may be the case that the most important single edge is (S, A) , but it is not present in the most important pair of edges, the two connecting node S to B .

dynamics can be used as minimal *struts* to construct a dual network \tilde{G}' with complementary reliability under SIR dynamics: $R_{OD}(\tilde{G}, \rho) = 1 - R_{OD}(G, 1 - \rho)$, as illustrated in the right panel of Fig. 4.

Deterministic processes and duality of weak- and strong-coupling. When all the state transition probabilities are either 1 or 0, the dynamical system is deterministic. The probability that a set of interactions leads to a configuration in \mathcal{C} is likewise either 1 or 0, as is the Birnbaum importance of every edge. In the face of this massive degeneracy, Birnbaum importance cannot be used to rank edges, but it still distinguishes relevant from irrelevant edges. Hence, it can still be used to find cruxes. In regimes near deterministic dynamics, we can write down perturbative expansions for reliability as Taylor series. For example, when the probability of transmission $\rho \ll 1$ under SIR dynamics, the interactions can be characterized as “weak”, because they rarely cause a state change; similarly, when $1 - \rho \ll 1$, the interactions are “strong”. Taylor series in ρ about $\rho = 0$ or in $1 - \rho$ about $\rho = 1$ are the well-known weak- and strong-coupling expansions of statistical physics. The coefficients of these two series can be derived from terms in the Inclusion-Exclusion expansion for minimal struts and cuts, respectively. The set covering relationship between cuts and struts produces constraints on terms in these series such as the Min Cut / Max Flow relationship.

4 Applications

We show here results for several applications of crux-based analysis under SIR dynamics: estimating the coefficients of weak- and strong-coupling expansions; edge ranking, specifically the problem of detecting a crossover in rank; finding cuts that recover the built-in structure in a stochastic blockmodel; and graph reduction or renormalization. First, though, we show that the theory helps us understand counter-intuitive properties such as the inconsistency of edge rankings as the budget changes, as mentioned in the Introduction.

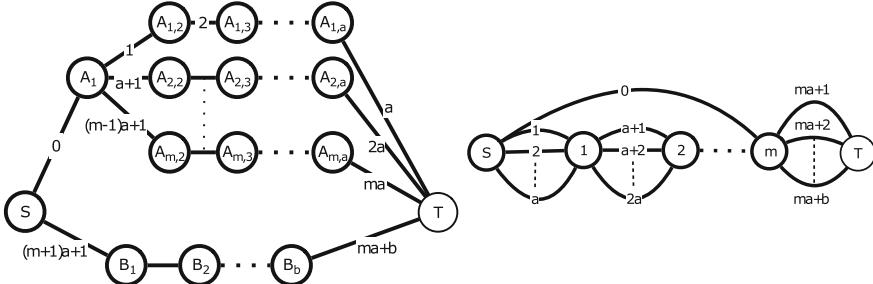


Fig. 4. (Left) A family of toy networks with 3 parameters (m, a, b). (Right) Their duals as defined here, for Two-Terminal reliability under SIR dynamics with transmissibility ρ . By construction, the network reliability of any member of the family, $G(m, a, b)$ as a function of transmissibility ρ is related to that of its dual $\tilde{G}(m, a, b)$ by $R_{OD}(\rho, G) = 1 - R_{OD}(1 - \rho, \tilde{G})$.

Generating interesting example networks. Suppose that in the network shown in Fig. 3, the probability of transmitting from node A to T is r_A and from B to T is r_B . Then the Birnbaum importance of combinations of the edges connected to S is:

$$B(\{(S, A)\}) = r_A - \rho(2 - \rho)r_A r_B \quad (6)$$

$$B(\{(S, B)\}) = (1 - \rho)r_B(1 - \rho r_A) \quad (7)$$

$$B(\{(S, A), (S, B)\}) = r_A + \rho(1 - \rho)r_B - \rho(2 - \rho)r_A r_B \quad (8)$$

$$B(\{(S, B), (S, B)\}) = \rho(2 - \rho)r_B(1 - r_A). \quad (9)$$

There is a feasible regime for which $B(\{(S, A)\}) > B(\{(S, B)\})$ but

$B(\{(S, B), (S, B)\}) > B(\{(S, A), (S, B)\})$. In this regime, not only the ranking but the edges ranked depend on the budget, b . When $b = 1$, the top-ranked edge is (S, A) , but when $b = 2$, it does not appear in the list of top-ranked edges.

Strong and weak coupling limits. For SIR dynamics, the probability of a strut s_i (not necessarily minimal) appearing in a set of active interactions is $\rho^{|s_i|}$, where $|s_i|$ is the number of edges in the strut. The probability that a pair of minimal struts χ_i and χ_j appear simultaneously in a set of active interactions is $\rho^{|\chi_i \cup \chi_j|}$. Hence the network reliability is a finite-degree polynomial in ρ . The coefficient of ρ^k depends only on the sets of cruxes whose union contains k or fewer edges. Evaluating these coefficients on a sample of minimal struts yields an approximate weak-coupling expansion; evaluating them on a sample of minimal cuts yields an approximate strong-coupling expansion. Although, as in other physical systems, these expansions often diverge due to poles in the reliability near the real axis, it is possible to combine them and bound the approximation error using the Bernstein basis for polynomials, i.e., Bezier curves. The left panel of Fig. 5 exhibits the two Taylor expansions, $R_{OD} = \rho + O(\rho^2)$ and $R_{OD} = 1 - 2(1 - \rho)^2 + O((1 - \rho)^3)$, along with the exact polynomial and an interpolating Bezier polynomial.

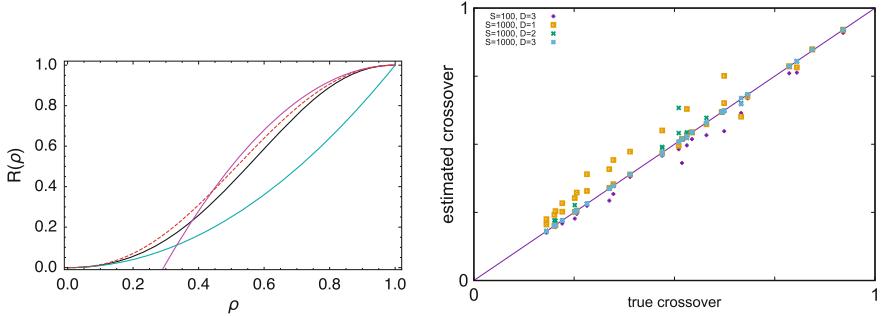


Fig. 5. (Left) The weak- and strong-coupling expansions (top and bottom curves at $\rho = 0.5$, respectively) and a Bezier interpolation (dashed curve) interpolating between the two, compared to the exact curve for Two Terminal reliability under SIR dynamics as a function of transmissibility ρ , for the network in Fig. 2. (Right) Estimated vs. exact crossover in relative importance of two edges for the family of toy networks shown in Fig. 4. As explained in the text, increasing the parameters S and D increases the accuracy and running time of the estimator.

Ranking edges. We compared the Birnbaum importance under Two Terminal reliability of two edges in the 3-parameter family of toy networks shown in the left panel of Fig. 4. The relative Birnbaum importance of the edges (S, A) and (S, B) is exactly: $\Delta_{A,B}(\rho; a, b, m) = \rho[1 - (1 - \rho^a)^m] - \rho^b$. Notice that the relative rank of the two edges changes at zeroes of Δ . Estimating the location of this crossover point is a stringent test of edge ranking. The right panel of Fig. 5 compares our estimates to the exact values for networks with $2 \leq m \leq 4$, $4 \leq a \leq 7$, and $3 \leq b \leq 6$.

Shattering networks. The minimal cuts for N-Terminal reliability are edge sets whose removal shatters a network into connected components with size at most N . Although the crux-finding algorithm described above identifies minimal cuts, it does not necessarily find the smallest ones. However, the set-covering relationship between minimal cuts and minimal struts can be used to construct approximately smallest cuts. A greedy algorithm removes one edge contained in the largest number of the most probable (smallest) cruxes in the Monte Carlo sample at each step. The order of removal is unimportant, because the goal is only to minimize the size of the final cut. We found cuts in two 128-node networks that are instances of planted 4-partition [4] or, equivalently, stochastic 4-block networks [6] under N-terminal reliability with $N = 32$: “Z3” has 3 inter-cluster edges and 13 within-cluster edges per node; “Z7” has 7 inter-cluster and 9 within-cluster edges. The cut size and modularity for Z3 are 188 and 0.561; for Z7, 409 and 0.334. The algorithm identifies the planted clusters for Z3, but not for Z7. Table 2 shows how the cut size (and modularity, for the incorrect Z7 clusters) depends on the crux sample size.

Graph reduction. The least important crux is one whose absence reduces reliability the least. Removing it from the set of all cruxes leaves a set that provides

Table 2. Shattering results for networks described in the text: (Left) Z3, (Right) Z7.

Sample size	Cut size	Sample size	Cut size	Modularity
50	324	200	624	0.113
200	270	1000	605	0.156
800	221	2000	596	0.151

a good approximation to the original network's reliability. We can generate a new, smaller network from the remaining cruxes that approximates the original network's reliability well. Because the reliability is held approximately invariant as the network is changed, and the reliability is essentially the system's partition function, this process is analogous to the real-space renormalization group approach of statistical physics.

Application to analyze real data. We used this reliability based algorithm to analyze international food and agricultural trade networks [1].

5 Future Directions

Algorithms exist for the applications above, but the framework developed here is valuable for many reasons. It clarifies how results depend on both the dynamics and the system configurations of interest and provides well-specified optimization functions with well-defined semantics for the solutions. The concepts introduced bridge gaps to methods proven useful in statistics, physics, and computer science. They provide a mechanism for reducing common questions to well-known problems such as set cover and submodular optimization. The algorithms and approximations the framework suggests are not heuristics, but are provably correct with bounds on computational complexity and approximation error and controllable tradeoffs between the two. Finally, because it is not based on spectral analysis, the framework generalizes easily to stochastic dynamical systems on directed hyper-networks.

The applications we have described here barely scratch the surface of possibilities. What are the net effects of approximations such as truncating the Inclusion-Exclusion expansion, sampling cruxes, and using a greedy algorithm? How large is the universality class of Two Terminal reliability under SIR dynamics? How well does graph renormalization using sampled cruxes work? Can the reduced networks be used to reason about dynamical stability on the original network? Can we create a random graph model based on network reliability? All these questions are amenable to investigation with formal methods within this framework.

References

1. Aiello, L.M., Cherifi, C., Cherifi, H., Lambiotte, R., Lio, P., Rocha, L.M. (eds.): Accepted to appear in Complex Networks & Their Applications VII. In: Proceedings

- of the 7th International Conference on Complex Networks and their Applications, COMPLEX NETWORKS 2018, Springer (2018)
- 2. Birnbaum, Z.W.: On the importance of different components in a multicomponent system. In: Krishnaiah, P.R. (ed.) Multivariate Analysis II, Proceedings of the 2nd International Symposium on Multivariate Analysis, pp. 581–592. Academic Press, New York (1969)
 - 3. Colbourn, C.J.: The Combinatorics of Network Reliability. Oxford University Press, New York (1987)
 - 4. Condon, A., Karp, R.M.: Algorithms for graph partitioning on the planted partition model. *Random Struct. Algorithms* **18**(2), 116–140 (2001)
 - 5. Gertsbakh, I.B., Shpungin, Y.: Models of Network Reliability: Analysis, Combinatorics, and Monte Carlo. CRC Press (2016)
 - 6. Karrer, B., Newman, M.E.: Stochastic blockmodels and community structure in networks. *Phys. Rev. E* **83**(1), 016107 (2011)
 - 7. Moore, E.F., Shannon, C.E.: Reliable circuits using less reliable relays. *J. Franklin Inst.* **262**(3), 191–208 (1956). [https://doi.org/10.1016/0016-0032\(56\)90559-2](https://doi.org/10.1016/0016-0032(56)90559-2), <http://www.sciencedirect.com/science/article/pii/0016003256905592>
 - 8. Nath, M., Ren, Y., Eubank, S.G.: Determining whether a particular contact network is consistent with a network model. *J. Theor. Biol.* **400C**, 121–132 (2017)
 - 9. Ren, Y., Eubank, S., Nath, M.: From network reliability to the Ising model: A parallel scheme for estimating the joint density of states. *Phys. Rev. E* **94**(4), 042125 (2016). <https://doi.org/10.1103/PhysRevE.94.042125>

Motif Discovery



New Deterministic Model of Evolving Trinomial Networks

Alexander Goryashko^{1(✉)}, Leonid Samokhine², and Pavel Bocharov³

¹ Moscow Technological Institute, Moscow, Russia

petrovich4you@gmail.com

² Qcue Inc., Austin, USA

lsamokhin@gmail.com

³ Wheely, Moscow, Russia

pavel@wheely.com

Abstract. We provide a new model of attributed networks where label of each vertex is a partition of integer n into at most m integer parts and all labels are different. The metric in the space of (n, m) -partitions is introduced. First, we investigate special class of the trinomial (m^2, m) -partitions as a base for synthesis of networks $G(m)$. It turns out that algorithmic complexity (the shortest computer program that produces $G(m)$ upon halting) of these networks grows with m as $\log m$ only. Numerical simulations of simple graphs for trinomial (m^2, m) -partition families ($m = 3, 4, \dots, 9$) allows to estimate topological parameters of the graphs—clustering coefficients, cliques distribution, vertex degree distribution—and to show existence of such effects as scale-free and self-similarity for evolving networks. Since the model under consideration is completely deterministic, these results put forward new mode of thought about mechanisms of similarity, preferential attachment and popularity of complex networks. In addition, we obtained some numerical results relating robust behavior of the networks to disturbances like deleting nodes or cliques.

Keywords: Partition · Simple graph · Perfect graph
Algorithmic complexity · Trinomial coefficients · Predictability
Topological parameters · Clustering coefficient · Connectivity
Emergency · Robust

1 Introduction

Evolving complex networks became the field of active research in the last twenty years. Most of the works are based on two preliminary assumptions: the networks are random (for example, the probability that two arbitrary nodes are connected equals $p > 0$) and the new nodes were added to the network in accordance to some rule of preferential attachment (for example, type of “rich gets richer”), i. e. growth depends on some heuristic rules. There were some attempts to introduce such special labels to nodes as ability to connect with other networks nodes [18].

In this case, the probability of establishing connections in the network depends on the characteristics of the nodes such as fitness and degree. However, the genesis of the “fitness” remains a kind of “Deus ex machine”.

Although stochastic models are predominating in studying of complex networks, the deterministic approach has certain advantages. First of all the deterministic models thanks to their nature often can be solved exactly without using asymptotic approximations. Besides some facts about behavior of complex networks can be obtained more simply especially when laws of the statistical physic fail. For example, in [3] the simple deterministic model that generates scale-free networks was introduced and authors show that the tail of the degree distribution follows a power law. First attempt to investigate changes in topological characteristics of networks in the case when simple “regular” networks (more exactly, chordal graphs) are “rewired” to introduce “disorder” was taken in [22]. The new approach to generate deterministic synthetic network models was proposed in a cycle of works (see [13] and references therein). Here each node can be attributed to some features. The probability of linking two nodes depends on individual attribute similarities what can model *homophily* (nodes with similar attribute values are more likely to be linked) or *heterophily*. Distinctions between random and deterministic models of complex networks were investigated in details with the aid of algorithmic complexity theory [8, 12] as evidenced by the body of publications in last years (see for example [15, 17] and references herein). Specifically, authors [23] had shown that estimate of K -complexity approximations captures important topological properties of networks such as symmetry, density and connectedness.

Our investigation is based on the invention of the special class of evolving deterministic graphs and networks possessing optimal K -complexity and high robustness simultaneously. We have shown that there is a simple method (short program) to design these networks. The basic peculiarity of this method is a new way of labeling the network nodes. Each node has unique label representing a partition of integer m^2 into at most m integer parts. The metric on the set of (m^2, m) -partitions is defined and connections of nodes in networks depend on distance between nodes in this metric. Therefore, one can get a family of self-similar complex networks where the number of nodes grows exponentially with m although K -complexity of such networks is optimal (up to in additive constant).

Our real world is corrupted by different noises, defects and disturbances. So synthesis of robust networks is one of the most important topics (both theoretically and in practice). Our computer modelling demonstrated that the class of networks under consideration is stable with respect to disconnection of the network.

The organization of the paper is as follows. In Sect. 3 we introduce basic definitions related to the model of labeling network elements and metric space of partitions. Section 4 presents a recurrent method for the design of partition network families and the principle of hereditary for such families. Here we show also how Kolmogorov complexity can be applied to the design of partition networks. The values of the topological parameters of families of graphs with

$m = 3, 4, \dots, 9$ derived from numerical experiments are presented in Sect. 5. The robustness of the partition networks (quantified by the proportion of nodes and cliques that can be removed before the network loses connectivity) is discussed in Sect. 6. We conclude the paper in Sect. 7 with a brief summary and the agenda for future work.

2 Preliminary Notations

Definition 1. *(n, m) -partition of n into at most m parts is defined as a sequence of non-negative integers $a_1 \geq a_2 \geq \dots \geq a_k \geq 0$, such that $n = a_1 + a_2 + \dots + a_m$. The set of all feasible (n, m) -partitions is denoted by $P(n, m)$ ¹.*

Definition 2. *For (n, m) -partition $\alpha = (a_1, \dots, a_m)$ the gaps between parts are the values $k_i = a_i - a_{i+1}$ ($1 \leq i \leq m - 1$). A partition which gaps k_i are the same and equal to some specific $k \geq 2$ is called partition with uniform gaps, and the set of all these partitions is denoted by $UGP_k(m)$. Note that any partition set $P(m^2, m)$ where $m \geq 3$ contains $UGP_2(m)$, which is the sequence of odd numbers $(2m - 1, 2m - 3, \dots, 3, 1)$.*

Definition 3. *The distance between partitions $\alpha = (a_1, \dots, a_m)$ and $\beta = (b_1, \dots, b_m)$ ($\alpha, \beta \in P(n, m)$) be defined as*

$$\rho(\alpha, \beta) = \max_{i=1, \dots, m} |a_i - b_i|. \quad (1)$$

For all $\alpha, \beta \in P(n, m)$ ($\alpha \neq \beta$) it holds $2 \leq \rho(\alpha, \beta) \leq 2n(1 - 1/m)$. The least upper bound is achieved for $\alpha = (n, 0, \dots, 0)$ and $\beta = (n/m, n/m, \dots, n/m)$.

Definition 4. *Let a graph $G(V_P, E)$ consist of a set of vertices (nodes) V where each node has unique label $\alpha \in \mathcal{P} \subseteq P(n, m)$ and every two nodes $v_\alpha, v_\beta \in V_P$ are connected by an edge $e \in E$ iff $\rho(\alpha, \beta) = 1$.*

3 Partition Graph Model

Let $G(V_P, E)$ from Definition 4 be a simple graph (an undirected graph without multiple edges or loops), $|V| = N$ and $A(G)$ be the binary adjacency $N \times N$ matrix where $a_{ij} = 1$ iff node v_i is connected with v_j . Adjacency matrices are indispensable for numerical calculation, but pictures are more useful visually. Below we display partition graphs by pictures. Suppose, we have the set $P(9, 3)$ containing 12 partitions: $P(9, 3) = \{(9, 0, 0), (8, 1, 0), (7, 2, 0), (7, 1, 1), (6, 3, 0), (6, 2, 1), (5, 4, 0), (5, 3, 1), (5, 2, 2), (4, 4, 1), (4, 3, 2), (3, 3, 3)\}$.

The graph $G_1(V_{P(9,3)}, E_1)$ (see Fig. 1) has 12 nodes labeled by partitions from $P(9, 3)$. Its clustering coefficient (C_c) is 0.465 and diameter (D) is 6. Let subset $\vartheta(9, 3) \subset P(9, 3)$ contains 7 partitions of 12: $(6, 3, 0), (6, 2, 1), (5, 4, 0)$,

¹ There exist optimal algorithm and recurrent rules for exact computation of number of partitions in the classes $P(n, m)$ [11].

$(5, 3, 1), (5, 2, 2), (4, 4, 1), (4, 3, 2)$. Evidently $G_1(V_{P(9,3)}, E_1)$ and $G_2(V_{\vartheta(9,3)}, E_2)$ are quite different topologically. Subgraph G_2 is the hexagon (symmetrical group $6n$) with central node is labeled by $\text{UGP}_2(3) = (5, 3, 1)$. For G_2 $Cc = 0.545$ and $D = 2$.

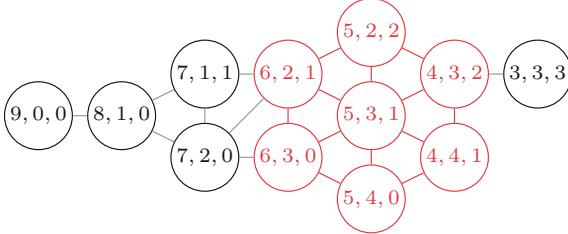


Fig. 1. The graph $G(V_{P(9,3)}, E_1)$ with subgraph $G(V_{\vartheta(9,3)}, E_2)$.

These simple examples are intended to illustrate the intuitively obvious fact – graph topology depends on labels of the nodes and on the metric on the set of partitions. Fortunately, there is deterministic method to design partition set with very intriguing and unexpected properties.

4 Trinomial Partition Family

Our goal is to propose the algorithm for synthesis of partition networks possessing the properties as follows:

- low algorithmic complexity;
- ability to create evolving topological structures;
- low cost of numerical simulation of the topological characteristics.

The choice of the metric for set partition (1) seems to be the best way to satisfy all properties mentioned above. Any (n, m) -partitions α, β will be adjacent iff $\alpha - \beta = (1 - 1, \dots, m - m)$ is sequence $\mathfrak{R}_m = (\delta_1, \delta_2, \dots, \delta_m)$, where $\delta_i = (1, 0, -1)$ for $i = 1, \dots, m$ and $\sum \delta_i = 0$. It is known [1, 2] that the number of permutations \mathfrak{R}_m exactly equals the values of the trinomial triangle (generalization of Pascal's Triangle) depicted on Fig. 2.

$$\begin{array}{ccccccc}
 & & & 1 & & & \\
 & & & 1 & 1 & 1 & \\
 & & & 1 & 2 & 3 & 2 & 1 \\
 & & & 1 & 3 & 6 & 7 & 6 & 3 & 1 \\
 & & & 1 & 4 & 10 & 16 & 19 & 16 & 10 & 4 & 1
 \end{array}$$

Fig. 2. First 5 rows of trinomial triangle.

Following the notation [1], the trinomial coefficient $\binom{m}{k}$ with $m \geq 0$ and $-m \leq k \leq m$, is given by the coefficient of x^{m+k} in the expansion of $(1+x+x^2)^m$.

The central trinomial coefficient ($k = 0$) for $m = 3, 4, 5, 6, 7, 8, 9, 10, \dots$ is given by the sequence 7, 19, 51, 141, 393, 1107, 3139, 8953, ... (sequence A002426 in the OEIS [19]). If $k = 1$, $k = -1$ trinomial coefficients is given by the sequence 6, 16, 45, 126, 357, 1016, 2907, 8350, ... (sequence A005717 in the OEIS). Asymptotic for central trinomial coefficient is $a(m) \approx d \cdot 3^m \sqrt{m}$, where $d = \sqrt{3/\pi}/2$ [19]. Thus, we have a possibility not only to estimate a priori the number of adjacent partitions for any $m \geq 3$, but also to create of a simple recurrent procedure to design graph $G(V_{\vartheta(m^2, m)}, E_m)$ from graph $G(V_{\vartheta((m-1)^2, m-1)}, E_{m-1})$ where $m > 3$. In what follows graph $G(V_{\vartheta(9, 3)}, E_2)$ (see Fig. 1) where $|V_{\vartheta(9, 3)}| = 7$ represent of initial TPF and be named *T-cell*².

4.1 Recurrent Procedure of Design Graph of TPF(m) Where $m > 3$

Step 1. Take partition $\alpha_0(m-1) \equiv \text{UGP}_{\vartheta}(m-1)$ in the set $\vartheta((m-1)^2, m-1)$ (there is only one such partition) where $a_1 = 2(m-1) + 1$ and prepend the odd number $a_1 = 2m+1$. Leave other parts without changes. As result, we will get a partition $\alpha_0(m) \equiv \text{UGP}_2(m)$. In the following $\alpha_0(m)$ will be named “head of family TPF(m)”.

Step 2. Prepend the odd number $2m+1$ to all partitions from the set $\vartheta((m-1)^2, m-1)$. In this way we will have a central subset of partitions $\vartheta_0(m^2, m)$.

Step 3. Find two side subsets: one $\vartheta_{-1}(m^2, m)$ with first part $a_1 = 2m$ and other $\vartheta_{+1}(m^2, m)$ with first part $a_1 = 2m+2$. Let $\vartheta(m^2, m) = \vartheta_0(m^2, m) \cup \vartheta_{-1}(m^2, m) \cup \vartheta_{+1}(m^2, m)$ is TPF(m). The distance between $\alpha_0(m)$ and any partition from $\vartheta(m^2, m)$ equals 1.

Step 4. Create graph $G(V_{\vartheta(m^2, m)}, E_m)$ where edge $e(n(i), n(j)) \in E_m$ iff $\rho(\alpha_{n(i)}, \beta_{n(j)}) = 1$ for labels $\alpha_{n(i)} = (a_1, \dots, a_m)$, $\beta_{n(j)} = (b_1, \dots, b_m)$. \square

Let us now estimate the complexity of procedure above by the K -complexity [12]. K -complexity (s) is defined for strings s of characters as

$$(s) = \min\{|P|, T(P) = s\}, \quad (2)$$

where P is a program which produces the string s when executed on the universal Turing machine and $|P|$ is the length of the program P —the number of bits required to represent P . K -complexity is upper semicomputable, i. e. only the upper bound of the value of K -complexity can be computed for a given string s . As is easy to see from description of this procedure the upper bound on the number of bits required to represent adjacency matrix of $G(V_{\vartheta(m^2, m)}, E_m)$ is $\log m + \text{const}$ because only value of m is necessary to create the appropriate program $P(m)$ ³. The running time for the procedure will grow exponentially

² There is *totipotency cell* in living organisms, i.e. a single cell with the ability to divide and produce all of the differentiated cells in an organism.

³ The closest analogy to low value for K -complexity is the problem of generating all partitions of n into at most m parts. An upper bound on K -complexity of the problem is $\log n$ because the recurrence holds for all integers m and n but running time to generate all partitions grows exponentially with n [11].

with m for universal Turing machine with one tape, but could be much lower (till $O(m)$) for case of parallel computing.

Example 1. Design graph $G(V_{\vartheta(16,4)}, E_4)$ from T-cell.

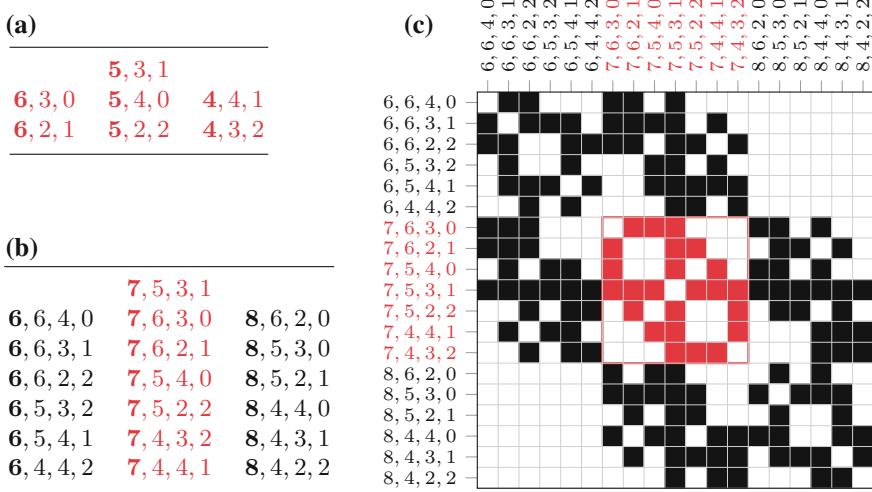


Fig. 3. Set of $(16, 4)$ -partitions $\vartheta(16, 4)$ (b), its T-cell (a) and adjacency matrix for $G(V_{\vartheta(16,4)}, E_4)$ (c). The red squares corresponds to adjacency matrix for T-cell.

As is clear from the description of the procedure and Fig. 3 at each step (increase m by one) adjacency matrix from the previous step becomes the central part of new matrix. This principle of iterative nesting provides for the effect of self-similarity (it is similar to Russian matryoshka doll).

5 Topological Properties of Graphs of $\text{TPF}(m)$ Graphs

Most of the problems of numerical modelling of graphs' properties are hard from the viewpoint of required computational resources (often these problems are NP-complete or NP-hard). However, it is known that for some graph classes—above all for the perfect graphs⁴—topological graph analysis is computationally accessible because its computational complexity grows as some polynomial of graph size. For example, the graph coloring problem, maximum clique problem, and maximum independent set problem can all be solved in polynomial time [9]. Fortunately, any simple graph $G(V_{\text{TPF}(m)}, E) \equiv G^v(m)$ where all nodes are labeled by partitions from $\text{TPF}(m)$ and simple graph $G(V_{\text{TPF}(m)/\alpha_0(m)}, E) \equiv G^v(m) \equiv G_0^v(m)$, i. e. $\text{TPF}(m)$ without “head” are the perfect graphs.

⁴ Perfect graph is a graph in which the chromatic number of every induced subgraph equals the size of the largest clique of that subgraph [16].

5.1 General Topological Parameters of TPF(m) Graphs ($m = 4, \dots, 9$)

The programming tools for computer modelling of the topological parameters can be found in [4, 20]. Topological parameters of trinomial families—graph $G^v(m)$, $m = 3, 4, \dots, 9$ are depicted in Tables 1 and 2 below.

Table 1. The results of numerical calculation the cliques parameters (size and quantity)

m	$ V $	Cliques in $G^v(m)$ (c_q , where q is quantity of cliques of the size c)								
		$(c_q$, where q is quantity of cliques of the size c)								
		$\binom{m}{1}^*$	$\binom{m}{2}$	$\binom{m}{3}$	$\binom{m}{4}$	$\binom{m}{5}$	$\binom{m}{6}$	$\binom{m}{7}$	$\binom{m}{8}$	Total
3	7	3 ₃	3 ₃	—	—	—	—	—	—	6
4	19	4 ₄	6 ₆	4 ₄	—	—	—	—	—	14
5	51	5 ₅	10 ₁₀	10 ₁₀	5 ₅	—	—	—	—	30
6	141	6 ₆	15 ₁₅	20 ₂₀	15 ₁₅	6 ₆	—	—	—	62
7	393	7 ₇	21 ₂₁	35 ₃₅	35 ₃₅	21 ₂₁	7 ₇	—	—	156
8	1107	8 ₈	28 ₂₈	56 ₅₆	70 ₇₀	56 ₅₆	28 ₂₈	8 ₈	—	254
9	3139	9 ₉	36 ₃₆	84 ₈₄	126 ₁₂₆	126 ₁₂₆	84 ₈₄	36 ₃₆	9 ₉	510

* $\binom{m}{i}$ —binomial coefficient ($i = 1, \dots, 8$).

The main (and surprising at the first glance) peculiarity of data from Table 1 is their predictability. It turns out that for $m = 3, 4, \dots, 9$ cliques' sizes are given by the first $\lceil m/2 \rceil$ binomial coefficients of Pascal triangle and total number of the cliques equals $2^m - 2$.

5.2 Graphs of Nodes $G_0^v(m)$ and Graphs of Cliques $G_0^{cl}(m)$

As one can see from Table 2 distribution of the degrees of vertices resembles a binomial distribution (distinct from power law for random graphs). Other topological properties of the graphs $G^v(m)$ —the diameter and the density index are different from randomly evolving network as well. For example, asymptotic estimate of the diameter D for Balabos-Riordan network model is $D(|V|) \approx \ln|V|/\ln\ln|V|$ [7]. It can be proved that in our deterministic model for any m $D(G^v(m)) = 2$ and $D(G_0^v(m)) = 3$. The value of the average index density of graph $G_v(m)$ decreases with m fairly slow (approximately as $1/\ln|V(G^v(m))|$ for $m = 4, \dots, 9$).

As is shown in Table 1 the sizes and the numbers of cliques in any graph $G^v(m)$ is known exactly. It is apparent that topological structure of TPF graphs (primarily, the numbers and the sizes of cliques) differs essentially from random graphs. For example, if every two nodes of random graph with 3139 nodes (as for $G^v(9)$) are connected with probability 0,94, the maximal size of clique is at most 100 [6]. The total number of nodes in all such cliques is *more than ten*

Table 2. The results of numerical calculation of graph's connectivity (degrees of nodes, density indexes and clustering coefficients).

<i>m</i>	V	Node	<i>I</i> (1)	<i>I</i> (2)	<i>I</i> (3)	<i>I</i> (4)	<i>I</i> (5)	<i>Md</i>	<i>MFd</i>	<i>MIdCl</i>	<i>MIdv</i>	<i>Cc</i>
3	7	Deg.	3	6	—	—	—	3, 45	2, 57	—	0, 87	0, 545
		Quant.	6	1	—	—	—					
4	19	Deg.	5	9	18	—	—	8, 12	3, 57	0, 527	0, 39	0, 595
		Quant.	6	12	1	—	—					
5	51	Deg.	13	25	50	—	—	18, 19	4, 9	0, 62	0, 34	0, 564
		Quant.	30	20	1	—	—					
6	141	Deg.	19	35	69	140	—	40, 37	6, 53	0, 698	0, 28	0, 522
		Quant.	20	90	30	1	—					
7	393	Deg.	49	95	191	393	—	87, 97	8, 72	0, 763	0, 22	0, 471
		Quant.	140	210	42	1	—					
8	1107	Deg.	69	132	261	533	1106	194, 67	11, 6	0, 816	0, 17	0, 422
		Quant.	70	553	420	55	1					
9	3139	Deg.	181	357	752	1499	3138	437, 38	15, 48	0, 858	0, 14	0, 375
		Quant.	630	1681	755	74	1					

The following notation is used: $I(j)$ —index of specific vertexes degree, $Cc(m)$ —clustering coefficient, $Md(m)$ —average node degree, $MFd(m)$ —average value intersection number for the graphs $G^{cl}(m)$. Two parameters were calculated for graphs without head of TPF(m): $MIdv(m)$ —average index density of graph $I(G^v)$, $MIdCl(m)$ —average index density of graph $I(G^{cl})$.

times larger than the total number of nodes in $G^v(9)$. The reason for this effect is high degree of overlapping of all cliques in TPF graphs. This is why such topological structure of graph $G^v(m)$ allows to get high robustness. To analyze this topological structure in more detail we suggest a method of representing TPF(m) as graph of the cliques $G(V^{cl}(m), E_{cl}) \equiv G^{cl}(m)$, where each “node” $v \in V^{cl}(m)$ is a different clique of size ≥ 3 . If in graph nodes $G^v(m)$ all nodes has different labels, “nodes” of graphs $G^{cl}(m)$ can have a *common* nodes. For any two cliques k, j $G_k^{cl}(m) \cap G_j^{cl}(m) \geq 0$ if some nodes $v \in G^v(m)$ are present in both cliques then J-distance⁵ > 0 . Average density index behavior for $G^{cl}(m)$ is the opposite of that true for $G^v(m)$ (see Table 2) and reminiscent of the “densification power law” [12] that implies that there exist evolving networks for which the density tends to increase.

6 Robustness of TPF(m)

In the general case before investigating problems of robustness, it would make sense to consider different failure types ranging from random errors in network elements and connections to disturbances of “epidemic” type. For example,

⁵ J-distance $d(A, B) = 1 - (A \cap B)/(A \cup B)$ is measure dissimilarity between two sets A and B (see [14]).

scale-free networks are resilient to random failures, but it seems that these networks are quite vulnerable to targeted hub removal. Here we are forced to restrict our considerations to the particular experimental results for the cases when disturbances mean deleting vertices in the graphs $G^v(5)$ or cliques in the graphs $G^{cl}(5)$. Before beginning of the experiments the “catastrophic” values of graph parameters are identified. It is assumed that the network with these topological parameters definitely is defective. Then at each step of experiment the cliques (“nodes” of graph $G^{cl}(m)$) are deleted one by one and the new topological parameters such damaged graph are recorded. The experiment is terminated when a “catastrophic” value is observed. Notice that all experiments below were conducted for partition family without $UGP_2(m)$ partition, i.e. without head of families. The head of the family is connected to each nodes of family. It means that for any graph $G^{cl}(m)$, $m > 3$ the density index will be 1 because each clique $k \in G^{cl}(m)$ can be overlapping with some other clique $k' \in G^{cl}(m)$. **To make the experimental results more adequate we assume that the node with maximum degree (head or family hub) is excluded from the family.**

6.1 Experimental Results: Deleting of Cliques for $G_0^{cl}(5)$

The initial parameters for $G_0^{cl}(5)$ are: 50 nodes, 10 cliques of size 4, 20 cliques of size 9, average node degree is 18, 19, clustery coefficient (Cc) is 0, 56 and diameter (D) is 3. Let the “catastrophic” parameters be: $Cc < 0, 2$, $D > 5$. At each step t some clique is chosen at random from the set of cliques that exist at moment t . The experimental results are represented below (Table 3).

Table 3. Numerical modelling results averaged over 20 experiments.

Average number of steps before catastrophe	7, 6
Average share of remaining nodes on last step before catastrophe	0, 38
Average share of remaining cliques on last step before catastrophe	0, 4

It worth to emphases that for random deleting of nodes from $G_0^{cl}(5)$ average share of remaining nodes on last step before catastrophe are approximately the same. Hence topological parameters which are responsible for connectivity of graphs $G_0^{cl}(5)$ and $G_0^v(5)$ are stable even when the graphs loss during experiments makes up almost two third of all “nodes”. What is more, average share of remaining nodes and cliques on last step before catastrophe practically is not change when m increased. Stability of the model under study is an inner feature of trinomial families’ topology, i.e. a consequence of the creation method rather than a result some special kind of redundancy.

For more descriptive the pictures of the graphs $G_0^{cl}(5)$ and numerical characteristics of these evolving graphs during experiments is represented on Fig. 4.

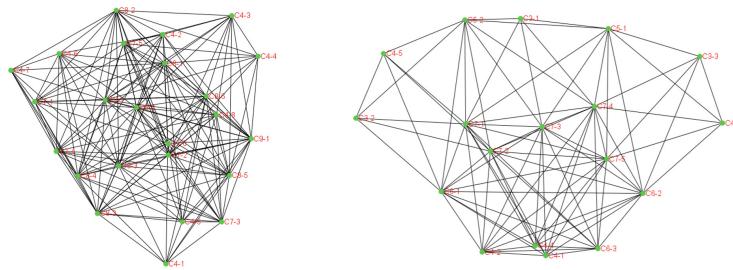
7 Conclusion Remarks

There are two essential notes about feasibility of trinomial partition family. Firstly, one can synthesis different kind structure of networks using *parts* of some trinomial partition set. By fact in this case such part of partition set plays the role of an topological basis for networks graphs. Secondly, there is the *algorithm* to create many graph classes (perfect graphs, chordal graphs, bipartition graphs) as sets of trinomial partitions. In essence, the merits of trinomial partitions approach are in algorithmic synthesis of networks with prearrange topological properties. This circumstance is especially important in technical applications when robustness problems are vital (for example, in high performance interconnect system for supercomputer). Can be shown that designing of interconnect system with help of trinomial partition approach could be no worse (at least) then such decisions as bipartition schemes or Clos networks.

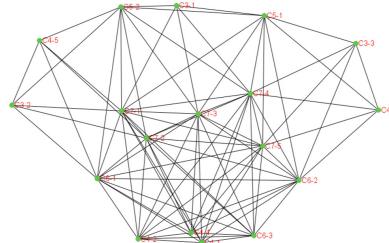
It is not to be supposed that researching of the robustness problems is the only (or most important) area of application of TPF model. A key step to advance our understanding of biological networks' evolving is unravelling the specific features of their network topology such as small K -complexity, scaling, variability and universality.

Changes in the topology of graphs of $\text{TPF}(m)$ caused by deleting their subgraphs set one thinking that we observe "living organisms" which tend to survive at any price and (as kind of biological entities) are ready to change the "genotype" by creating new "phenotypes" to be fitted for new conditions. However, one can suppose that any subgraph (clique) could be in "active state" and "passive state" depending on environmental influences. In this case, graph of partition family could be functioning as the "coder" or the "memory" that mapping input signals to topology of the graph. While this is an assumptions only, its possible that future investigations of the our model might help to understand the nature and nurture of mechanisms ensuring basic properties of biologic networks.

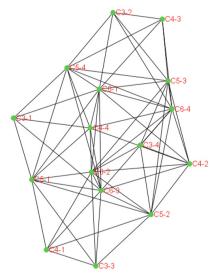
One of effective approaches to establishing connection between "intention" of elements of network and network topology is network formation games [21]. In these games network nodes are "players" and all "players" try to connect with other nodes by buying edges. The players minimize a cost function that depends on number and length of the network paths. It seems that TPF models are suitable for designing effective topological structures as solution of network formation games [5, 10].

**Step 2.**

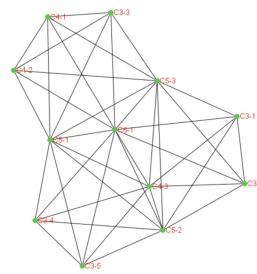
3 cliques deleted.

 $Cc = 0.558$. $D = 3$. Nodes: 41.**Step 3.**

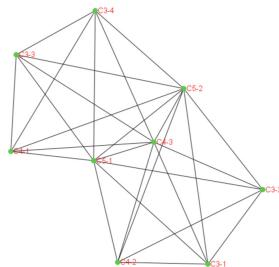
5 cliques deleted.

 $Cc = 0.549$. $D = 3$. Nodes: 31.**Step 4.**

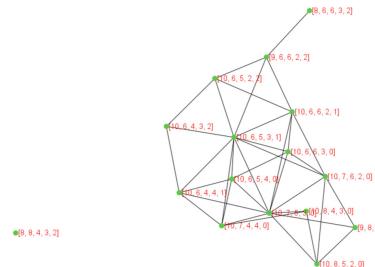
7 cliques deleted.

 $Cc = 0.524$. $D = 3$. Nodes: 27.**Step 5.**

9 cliques deleted.

 $Cc = 0.545$. $D = 4$. Nodes: 20.

Graph of cliques.



Graph of nodes.

Step 6 (last).

10 cliques deleted.

 $Cc = 0.545$. $D = \infty$. Nodes: 16.**Fig. 4.** Numerical characteristics of evolving graphs during the process of deleting cliques in $G_0^{cl}(5)$.

References

1. Andrews, G.E.: Number Theory (1971)
2. Andrews, G.E.: Euler’s “Exemplum memorabile inductionis fallacis” and q-trinomial coefficients. *J. Am. Math. Soc.* **3**(3), 653–669 (1990). <https://doi.org/10.2307/1990932>
3. Barabási, A.L., Ravasz, E., Vicsek, T.: Deterministic scale-free networks. *Physica A: Stat. Mech. Appl.* **299**(3–4), 559–564 (2001). [https://doi.org/10.1016/S0378-4371\(01\)00369-7](https://doi.org/10.1016/S0378-4371(01)00369-7)
4. Bocharov, P.: Partition Games Research Toolbox (2015). <https://github.com/pbo/partition-games>
5. Bocharov, P., Goryashko, A.: Evolutionary dynamics of partition games. In: 2015 International Conference “Stability and Control Processes” in Memory of V.I. Zubov (SCP), pp. 225–228. IEEE (2015). <https://doi.org/10.1109/SCP.2015.7342108>. <http://ieeexplore.ieee.org/document/7342108/>
6. Bollobás, B.: Random graphs. In: Modern Graph Theory, pp. 215–252. Springer, New York (1998)
7. Bollobás, B., Riordan, O.M.: Mathematical results on scale-free random graphs. In: Handbook of Graphs and Networks (2004). <https://doi.org/10.1002/3527602755.ch1>
8. Chaitin, G.J.: On the length of programs for computing finite Binary Sequences: statistical considerations. *J. ACM* **13**(4), 547–569 (1969). <https://doi.org/10.1145/321495.321506>
9. Cvetkovic, D., Rowlinson, P., Simic, S., Beineke, W., Godsil, C., Royle, G., Biggs, N., Doob, M., Wilson, R.J.: Algebraic Graph Theory (2001). <https://doi.org/10.1007/978-1-4613-0163-9>
10. Goryashko, A.: On suboptimal solution of antagonistic matrix games. *ITM Web of Conferences*, vol. 1(03001) (2017). <https://doi.org/10.1051/itmconf/20171003001>
11. Knuth, D.E.: The Art of Computer Programming, Combinatorial Algorithms, Part 1, vol. 4A. Addison-Wesley Professional (2011)
12. Kolmogorov, A.N.: Three approaches to the quantitative definition of information. *Probl. Inf. Trans.* **1**(1), 1–7 (1965)
13. Leskovec, J., Chakrabarti, D., Kleinberg, J., Faloutsos, C., Ghahramani, Z.: Kronecker graphs: an approach to modeling networks. *J. Mach. Learn. Res.* **11**, 985–1042 (2010)
14. Levandowsky, M., Winter, D.: Distance between sets. *Nature* **234**(5323), 34 (1971)
15. Li, M., Vitányi, P.M.B.: Kolmogorov complexity and its applications. Centre for Mathematics and Computer Science (1989)
16. Lovász, L.: Perfect graphs. *Sel. Top. Graph Theory* **2**, 55–87 (1983)
17. Morzy, M., Kajdanowicz, T., Kazienko, P.: On measuring the complexity of networks: kolmogorov complexity versus entropy. *Complexity* (2017)
18. Newman, M.E.J.: The structure and function of complex networks. *SIAM Rev.* **45**(2), 167–256 (2003). <https://doi.org/10.1137/S003614450342480>
19. OEIS Foundation Inc.: The On-Line Encyclopedia of Integer Sequences (2018). <https://oeis.org>
20. Samokhine, L.: Trinomial Family Research Toolbox (2017). <https://github.com/samokhine/gory>
21. Tardos, E., Wexler, T.: Network Formation Games and the Potential Function Method (2007). <https://doi.org/10.1145/1785414.1785439>
22. Watts, D.J., Strogatz, S.H.: Collective dynamics of ‘small-world’ networks. *Nature* **393**(6684), 440 (1998). <https://doi.org/10.1038/30918>
23. Zenil, H., Kiani, N.A., Tegnér, J.: Low-algorithmic-complexity entropy-deceiving graphs. *Phys. Rev. E* (2017). <https://doi.org/10.1103/PhysRevE.96.012308>



Counting Multilayer Temporal Motifs in Complex Networks

Hanjo D. Boekhout^(✉), Walter A. Kosters, and Frank W. Takes

Department of Computer Science (LIACS), Leiden University, Leiden,
The Netherlands

h.d.boekhout@umail.leidenuniv.nl, hanjo.boekhout@gmail.com

Abstract. This paper proposes a novel approach to count temporal motifs in multilayer complex networks. Network motifs, i.e., small characteristic patterns of a handful of nodes and edges, have repeatedly been shown to be instrumental in understanding real-world complex systems. However, exhaustively enumerating these motifs is computationally infeasible for larger networks. Therefore, the focus of this work is on algorithms that efficiently *count* network motifs. This facilitates the discovery of motifs in networks with millions of nodes and edges, enabling the following three contributions of this paper. First, we propose an extension of an existing counting algorithm to also efficiently count *multilayer* temporal motifs. In addition to dealing with the timestamp at which a link (re-)occurs, the algorithm also efficiently counts interaction patterns across different network layers. Second, we demonstrate how partial timing, a common phenomenon in real-world settings where only part of the layers are timed, can be incorporated. Third, we assess the performance of the proposed temporal multilayer counting algorithm on a number of real-world network datasets. Experiments reveal interesting insights in the heterogenous interplay between network layers in, for example, online expert communities, showing how particular temporal motifs are characteristic for certain layers of interaction.

Keywords: Motif counting · Multilayer temporal networks
Multilayer temporal motifs

1 Introduction

Many real-world complex networks evolve over time. For example, an online social communication network evolves as new messages (edges) between users (nodes) are created. This evolution can be modelled by adding timestamps to edges, creating so-called *temporal networks*. Sometimes it is convenient to capture multiple types of relationships between nodes in a single network. For example, in online communication networks we can also capture which type of communication (private or public) took place between users. This is modelled by adding labels to each edge denoting what type of relationship it represents. These networks are referred to as *multilayer networks*, also known as *multiplex networks*.

In recent years, there has been growing interest in identifying frequent small subgraphs. Such subgraphs are often referred to as network *motifs* or graphlets [2, 11]. These motifs can be used to reconfirm existing hypotheses about certain interaction patterns, but they can also provide new insight into previously unknown behaviour in the network [6, 15, 21]. Many different methods have been proposed for finding motifs. Most methods are only focussed on static single layer networks. In recent work, efficient temporal motif counting algorithms have been presented [12]. Next to that, multilayer motif detection algorithms have been explored [1, 16]. However, there are no out-of-the box algorithms for counting *multilayer temporal motifs*, i.e., motifs with edges containing both timestamps and a layer identifier. This is not surprising since both the temporal and multilayer aspect add significant complexity to the task. Yet, a particular order of events with multiple types of interaction can represent a plethora of interesting real-world phenomena. For example, as we will show in our experiments, in the communication network of the online knowledge sharing network Stackoverflow, different motifs contain distinct combinations of different types of user communication (answering, commenting and clarifying).

To address the challenges of counting multilayer temporal motifs, we propose a fast multilayer temporal counting algorithm, based on the efficient temporal motif counting algorithms introduced in [12], able to cope with *multiple layers*. In addition, because real-world multilayer networked systems often do not have known timestamps on all layers, we propose a solution for *partial timing*. The counting algorithms discussed in this paper consider a network as a sequence of edges, which subsequently only has to be traversed a small number of times, resulting in time complexity linear in the number of edges. This enables us to study temporal motifs in multilayer networks with millions of nodes and edges. For example, a network of 48 million edges takes only 5 to 10 minutes to process.

The remainder of this paper is structured as follows: in Sect. 2 we discuss related work and in Sect. 3 we provide definitions for multilayer temporal networks and motifs. Then in Sect. 4 we present our extended algorithms for counting temporal motifs in networks with multiple layers supporting partial timing. Section 5 describes the experiments and their results and finally in Sect. 6 we summarize our findings and conclusions.

2 Related Work

We discuss related work on motif enumeration, motif counting, multilayer motifs and temporal motifs: the ingredients for our multilayer temporal motif counting.

Three categories of static motif *enumeration* exist. In single-motif enumeration and motif-set enumeration, one or a few preselected motifs are enumerated [5, 13], whereas in all-motif enumeration all motifs of a particular size are listed, often exploiting CPU parallelism, e.g., the ESU and FANMOD algorithms [18–20]. Rather than enumerating from each node, the later proposed PSE algorithm starts enumerating from each edge, enabling more balanced parallel tasks [14].

Static motif *counting* typically outperforms enumeration, often by limiting oneself to motifs with a particular structure or size, e.g., as done in [10]. Similarly, in [4] local counting algorithms count only the motifs a single node is involved in.

Work on *multilayer motifs* started relatively recently. Kivela and Porter [7] addressed the problem of detecting subgraph isomorphisms in multilayer networks and temporal networks through time sequence graphs. Takes et al. [16] proposed a multiplex adaptation of Subenum [14] and applied it to corporate networks. Each of these approaches still faced limits of motif enumeration.

Work on *temporal motifs* has developed over the years to more accurately capture timing information. Braha and Bar-Yam [3] used snapshots, ignoring the order of events. Later work by Zhao et al. [22] considers two events (edges) linked if they share a node and are subsequent within a time limit Δt , enforcing local time adjacency. Kovanen et al. [8] called such events Δt -adjacent and considered two events Δt -connected if there is a sequence of Δt -adjacent events joining them. Finally, in 2017, Paranjape et al. [12] count temporal motifs where every pair of edges is at most δ time apart, fully utilising timing information. We build upon these techniques, adding both partial timing and multiple network layers.

3 Multilayer Temporal Motifs

In this section, we provide definitions for the algorithms described in the remainder of this paper, following the notation introduced in [12].

The basic building block of network structure is an *edge*: a (directed) link between an ordered pair of nodes. It is defined as a tuple (u, v) with u denoting the source node and v the target node. Given a node set V , a *static graph* $G = (V, E)$ is defined by a set E containing edges (u_i, v_i) , for $i = 1, 2, \dots, m$, with $u_i, v_i \in V$. For *temporal edges* we add a timestamp t and for *layered edges* we add a layer number l . Thus, in a *multilayer temporal graph* H an edge is defined as (u_i, v_i, t_i, l_i) , where $t_i \in \{-1\} \cup \mathbb{R}^+$ and $l_i \in \{1, \dots, \Lambda\}$, with Λ the number of layers. A timestamp of -1 indicates there is no known timestamp for that edge (in case of partial timing). The *underlying static graph* of a multilayer temporal graph is the graph formed by ignoring all timestamps, layers and duplicate edges. For the algorithms in this paper we assume edges to always be directed. However, results for undirected edges can be obtained through post-processing.

We extend the definition of δ -temporal motifs in [12] with layers as follows:

Definition. A *r-node*, *s-edge*, *δ -temporal*, *λ -layered motif* is a sequence of s edges, $M = ((u_1, v_1, t_1, l_1), (u_2, v_2, t_2, l_2), \dots, (u_s, v_s, t_s, l_s))$ that are time-ordered within a δ duration, i.e., $t_1 < t_2 < \dots < t_s$ and $t_s - t_1 \leq \delta$, and range over at most λ different layers such that the underlying static graph is connected and has r nodes.

Note that multiple edges between the same pair of nodes are possible and individually counted and that timestamps induce an ordering on the edges. Furthermore, this definition allows λ different layers in the motif M , but also allows fewer layers. For example, Fig. 1c ($M_{1,3,3}$) shows a 3-node, 3-edge, δ -temporal,

3-layered motif including just 2 layers, given a suitable δ . We say that a motif $M = ((u_1, v_1, t_1, l_1), \dots, (u_s, v_s, t_s, l_s))$ occurs in a multilayer temporal graph H , when there is a time-ordered sequence $S = ((w_1, x_1, t'_1, l'_1), \dots, (w_s, x_s, t'_s, l'_s))$ of s unique edges in H such that

1. there exists a bijection f such that $f(w_i) = u_i$ and $f(x_i) = v_i$ ($i = 1, \dots, s$),
2. the edges all occur within δ time, i.e., $t'_s - t'_1 \leq \delta$, and
3. there exists a bijection g on the layers such that $g(l'_i) = l_i$ ($i = 1, \dots, s$), which holds for all motifs within a single search.

Each such sequence of edges is called an *instance* of the motif M , and the goal of this paper is to count the number of such instances.

Problem statement. *Given set values for r , s , δ and λ and a multilayer temporal graph H , compute the number of occurrences of each motif.*

The fast algorithms presented in [12] focus on 2,3-node (meaning 2 or 3 nodes), 3-edge δ -temporal motifs, providing the overview in Fig. 1a. Altering an edge's layer does not affect the temporal order or edge configuration, so every δ -temporal λ -layered motif can be associated with a single δ -temporal motif. Figure 1c shows all 3-node, 3-edge, δ -temporal, 3-layered motifs, given a single δ -temporal motif $M_{1,3}$ from Fig. 1a. The number of associated δ -temporal λ -layered motifs for a single δ -temporal motif depends on the number of possible layer permutations. So for each s -edge, δ -temporal motif there exist λ^s δ -temporal, λ -layered motifs. Thus, there are $3^3 \times 36 = 972$ 2,3-node, 3-edge, δ -temporal, 3-layered motifs. To reference one δ -temporal λ -layered motif, we add a layer-related index into the possible permutations of Fig. 1c.

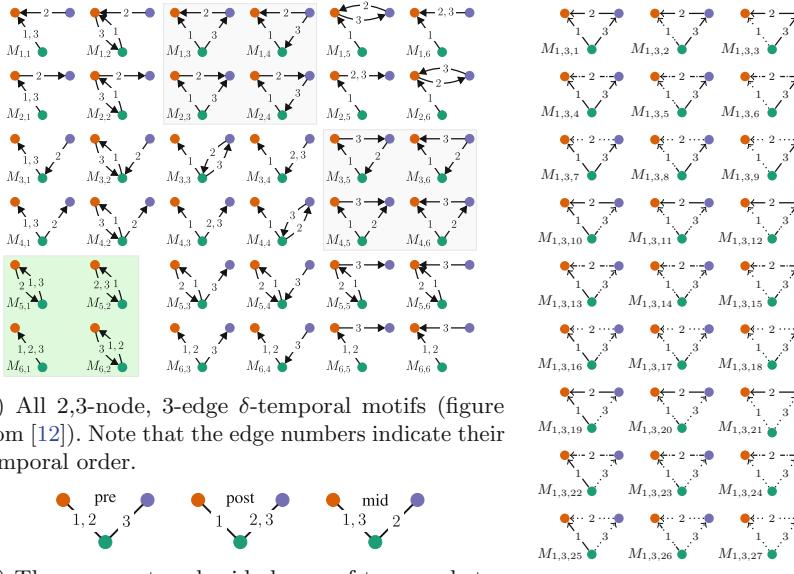


Fig. 1. Overview of different motif types.

4 Extended Algorithms

First a general algorithm is presented in Sect. 4.1, incorporating both the multilayer aspect as well as partial timing into the original algorithms presented in [12]. This algorithm is not optimal for 3-node motifs. Therefore specific extended algorithms for stars and triangles are presented in Sects. 4.2 and 4.3.

4.1 General Motif Counting

The general algorithm consists of a 3-step procedure. First, all instances U' of the static motif U , underlying M , in the static graph G , underlying the multilayer temporal graph H , are identified. This can be accomplished with known algorithms for enumerating static motifs. Second, for each motif instance U' , all temporal edges between pairs of nodes forming an edge in U' are gathered into an ordered sequence S' . We subsequently filter these temporal edges, such that the layers from the edges match those in U . We denote the resulting sequence of edges by S'' . Finally, the number of subsequences of edges in S'' occurring within δ time units that correspond to instances of M are counted using Algorithm 1.

Partial timing. With respect to the original algorithm in [12], the highlighted code in Algorithm 1 denotes the changes for partial timing. From lines 2–3 we can observe that untimed edges are never forgotten, acknowledging that they could have formed at any given time and should be considered part of every δ timeframe. This does mean that in the order of events of the edges, the untimed edges are always first. The additional for loop on lines 14–15 can easily be merged with the preceding for loop. Thus, we only add a small number of operations per edge which should not significantly impact the algorithm’s time complexity.

Multilayer aspect. The addition of multiple layers is realised with by adding a parameter l to each edge-related parameter, which only impacts the number of possible keys for the array counts[\cdot]. Added parameters are shown in grey.

4.2 Star Motif Counting

Star motifs are motifs that consist of a center node u and edges to $r - 1$ neighbours, with no edges connecting these neighbours. Example star motifs are $M_{1,1}$, $M_{1,5}$ and $M_{5,5}$ in Fig. 1a. We define each edge in a star motif by its neighbour node (nbr), its direction towards or away from u (dir), its timestamp (t) and its layer (l). The static motifs underlying 3-node 3-edge λ -layered star motifs can be divided into three classes: pre, post and mid, as shown in Fig. 1b. While processing the time-ordered sequence of edges, we consider the current edge being processed as the singular edge in the motifs, i.e., edge 3 for pre. Algorithm 2 provides the algorithmic framework for the fast counting algorithms, with full extended implementations of $Push()$, $Pop()$ and $ProcessCurrent()$ in Algorithm 3.

Partial timing. The highlighted code indicates the changes required for handling partially timed networks. Just like for the general algorithm, we first

require the untimed edges to be preprocessed. During this preprocessing the fully untimed motifs are counted. For partially timed pre motifs additional operations are added on lines 21, 27 and 31. With partially timed mid motifs, we must distinguish between the cases where the single edge is either timed or untimed. Lines 2–3 account for the second case as well as partially timed post motifs. For the timed single edge case, we require an additional type of mid motif counter (`ppre_mid`), which is used to count the number of combinations of edges 1 and 3 of the mid type motif found.

Multilayer aspect. We can see that adding layers does not change the operation of the original algorithm [12], but only requires us to add a layer index for every direction index to each counter. In Sect. 4.4, we will discuss how including layers does impact the space and time complexities of the algorithm.

Furthermore, note that, like the original algorithm [12], our extended algorithm also includes instances of 2-node motifs, which we subtract from the count using the extended general algorithm (as this algorithm is still optimal for size 2). The full process of counting multilayer temporal star motifs is then: (1) for each node u in the multilayer temporal graph H , get a time-ordered list of all

Algorithm 1: Algorithm for counting the number of instances of all possible s -edge δ -temporal λ -layered motifs in an ordered sequence of multilayer temporal edges. We assume the keys of `counts[.]` are accessed in order of length.

```

Input: Sequence  $S''$  of edges  $(e_1 = (u_1, v_1), t_1, l_1), \dots, (e_L, t_L, l_L)$  with
 $t_1 \leq \dots \leq t_L$ , time window  $\delta$ ,  $l \in \{0, \lambda - 1\}$ 
Output: Number of  $s$ -edge  $\delta$ -temporal  $\lambda$ -layered motifs  $M$  in sequence  $S''$ 
1 start  $\leftarrow 1$ , counts  $\leftarrow$  Counter(default = 0), pcounts  $\leftarrow$  Counter(default = 0)
2 while  $start < L$  and  $t_{start} == -1$  do
3   IncrementCounts(pcounts, estart, lstart, 0);
4 for  $end = start, \dots, L$  do
5   while  $t_{start} + \delta < t_{end}$  do
6     DecrementCounts(estart, lstart, start += 1
7     IncrementCounts(counts, eend, lend, 1)
8 counts += pcounts
9 Procedure DecrementCounts(e,l)
10  counts[(e,l)] -= 1
11  for suffix in counts.keys of length  $< s - 1$  do
12    counts[concat((e,l), suffix)] -= counts[suffix]
13  for prefix in counts.keys.reverse() of length  $< s - 1$  do
14    counts[concat(prefix, (e,l))] -= pcounts[prefix];
15 Procedure IncrementCounts(counts,e,l, type)
16  for prefix in counts.keys.reverse() of length  $< s$  do
17    counts[concat(prefix, (e,l))] += counts[prefix] (+ pcounts[prefix] if
18      type == 1)
    counts[(e, l)] += 1

```

edges containing u ; (2) use Algorithms 2 and 3 to count star motifs; and (3) for each neighbour v of a star center u , subtract the 2-node motif counts using Algorithm 1.

4.3 Triangle Motif Counting

Triangle motifs are motifs that, as the name suggests, form a triangle (see Fig. 1c). We define each triangle by nodes u and v and a common neighbour. Each edge in a triangle motif is defined by a neighbour node, an indicator whether it is connected to u or v ($uv|v$), a direction, a timestamp and a layer. The same framework used for counting star motifs, can be used to count triangle motifs.

The biggest difference between counting star and triangle motifs is that, for triangle motifs, we must distinguish between behaviour for updating using an edge connected to u or v . Therefore, all counters are updated with an additional field which distinguishes between counts for the first edge connecting to u and v . Edges between u and v are used as the final edge for forming triangles. Therefore, these edges are ignored in $\text{Push}()$ and $\text{Pop}()$. Furthermore, $\text{ProcessCurrent}()$ splits its updates between updating the mid counters and the complete counters for edges (u, v) . The implementations of $\text{Push}()$, $\text{Pop}()$ and $\text{ProcessCurrent}()$

Algorithm 2: Algorithmic framework for faster counting of 3-node, 3-edge, δ -temporal, λ -layered star (and triangle) temporal motifs.

Input: Sequence of edges $(e_1 = (u_1, v_1), t_1, l_1), \dots, (e_L, t_L, l_L)$ with
 $t_1 \leq \dots \leq t_L$, time window δ , $l \in \{0, \lambda - 1\}$

- 1 Initialise counters pre_nodes, post_nodes, mid_sum, pre_sum, post_sum,
- 2 ppre_nodes, ppost_nodes, pmid_sum, ppre_sum, ppost_sum and ppre_mid
- 3 end $\leftarrow 1$, i $\leftarrow 1$
- 4 **while** $end < L$ **do**
 - 5 $\lfloor \text{Push}(ppost_nodes, ppost_sum, e_{end}, l_{end}, 0), end += 1$
- 6 **while** $i < L$ *and* $t_i == -1$ **do**
 - 7 $\lfloor \text{Pop}(ppost_nodes, ppost_sum, e_i, l_i, 0)$
 - 8 $\lfloor \text{ProcessCurrent}(ppre_nodes, ppost_nodes, pmid_sum, ppre_sum,$
 $\text{ppost_sum}, e_i, l_i, 0)$
 - 9 $\lfloor \text{Push}(ppre_nodes, ppre_sum, e_i, l_i, 0), i += 1$
- 10 start $\leftarrow i$, end $\leftarrow i$
- 11 **for** $j = i, \dots, L$ **do**
 - 12 **while** $t_{start} + \delta < t_j$ **do**
 - 13 $\lfloor \text{Pop}(pre_nodes, pre_sum, e_{start}, l_{start}, 1), start += 1$
 - 14 **while** $t_{end} \leq t_j + \delta$ **do**
 - 15 $\lfloor \text{Push}(post_nodes, post_sum, e_{end}, l_{end}, 2), end += 1$
 - 16 $\lfloor \text{Pop}(post_nodes, post_sum, e_j, l_j, 0)$
 - 17 $\lfloor \text{ProcessCurrent}(pre_nodes, post_nodes, mid_sum, pre_sum, post_sum,$
 $e_j, l_j, 1)$
 - 18 $\lfloor \text{Push}(pre_nodes, pre_sum, e_j, l_j, 1)$

are available at https://bitbucket.org/Fractals-/count_mult_temp_motifs. Again, we note that the addition of multiple layers does not impact the mode of operation of the counting algorithm. Furthermore, the same type of operations are used for counting partially timed star and triangle motifs.

Algorithm 3: Implementation of Algorithm 2 functions for efficiently counting 3-node, 3-edge, δ -temporal, λ -layer star motif instances. The “`:`” notation indicates selection of all array indices.

```

19 Initialise counters count_pre, count_post, count_mid
20 Procedure Push(node_count, sum, e = (nbr,dir), l, type)
21   sum[:, :, dir, l] += node_count[:, nbr, :] (+ ppre_nodes[:, nbr, :] if type == 1)
22   if type == 2 then ppre_mid[:, :, dir, l] += ppre_nodes[:, nbr, :]
23   node_count[dir, nbr, l] += 1
24 Procedure Pop(node_count, sum, e = (nbr,dir), l, type)
25   node_count[dir, nbr, l] -= 1
26   sum[dir, l, :, :] -= node_count[:, nbr, :]
27   if type == 1 then sum[:, :, dir, l] -= ppre_nodes[:, nbr, :]
28 Procedure ProcessCurrent(pre_n, post_n, mid_s, pre_s, post_s, e = (nbr,dir),
29   l, type)
30   mid_s[:, :, dir, l] -= pre_n[:, nbr, :]
31   if type == 1 then ppre_mid[:, :, dir, l] -= ppre_nodes[:, nbr, :]
32   count_pre[:, :, :, dir, l] += pre_s[:, :, :, :] (+ ppre_sum[:, :, :, :] if type == 1)
33   count_post[dir, l, :, :, :] += post_s[:, :, :, :]
34   count_mid[:, :, dir, l, :, :] += mid_s[:, :, :, :, :] (+ ppre_mid[:, :, :, :, :] if type == 1)
35   mid_s[dir, l, :, :] += post_n[:, nbr, :]
35 return count_pre, count_post, count_mid

```

Analogously to [12] we assign each triangle to the pair of nodes with the largest edge count, so that as many triangles as possible are processed at once.

4.4 Complexity of Extended Fast Algorithms

Our extended algorithm has time complexity $O(|S''|)$, as the mode of operation is essentially the same as the original algorithms [12]. This is different for Algorithms 2 and 3. Our extended star algorithm performs $O(\lambda)$ operations for both *Push* and *Pop* functions and $O(\lambda^2)$ for *ProcessCurrent*, adding $O(\lambda^2)$ for each edge. This also holds for partially timed networks. However, for small λ , λ^2 is negligible with respect to time complexity, i.e., $O(\lambda^2) = O(1)$, and the extended algorithm remains linear in the size of the input sequence. Note that for a single layer network $O(\lambda^2) = O(1)$. Similarly, for our extended triangle algorithm we go from an original complexity of $O(1)$ to $O(\lambda^2)$.

Compared to the original algorithm, the sizes of the “sum” and “count” counters increase respectively by a factor of λ^2 and λ^3 . With small λ and the largest of these datastructures being $8\lambda^3$, the space requirements for these counters are

negligible. However, the “nodes” counters require a far greater amount of space. In our extended algorithm, we increase the size of the “nodes” counters by a factor λ . Thus, each “nodes” counter consists of $4\lambda k$ integers, where k is the number of neighbours. Worst case, all other nodes are neighbours and k equals $|V| - 1$. Therefore, the much smaller factor λ is negligible in space complexity.

Table 1. Network dataset statistics.

Dataset	Nodes	Edges	Static edges	Layers	Max. deg.
EMAIL-EU-CORE	985	24,929	24,929	2	345
MATH-OVERFLOW	24,759	390,441	228,215	3	2,172
FACEBOOK/WOSN2009	63,792	2,401,228	1,592,562	2	1,100
ASK-UBUNTU	157,222	726,661	544,774	3	5,401
SUPER-USER	192,409	1,108,739	854,377	3	14,294
STACK-OVERFLOW	2,584,164	47,903,266	34,901,115	3	44,065

5 Experiments

In this section we describe the experimental setup, followed by a description of our data and the results of applying the proposed algorithms (see Sect. 4).

5.1 Experimental Setup

Algorithms were implemented as a component of the SNAP network analysis package [9] and are available at https://bitbucket.org/Fractals-/count_mult_temp_motifs. Experiments were run on a machine with 16 Intel Xeon E5-2630v3 CPUs at 2.40 GHz (32 threads) and 512 GB RAM. Execution runtimes do not include the time required for reading the graph from disk. All runtimes were averaged over 10 runs. Time window δ was set to a percentage (1%, 5%, 10%, 20%, 50% and 100%) of the full timespan covered by the temporal network.

5.2 Data

Of the six datasets, descriptive statistics are shown in Table 1, where column “Max. deg.” contains the largest degree values over all nodes and “Static edges” the number of edges in the underlying static graph. Self-edges are excluded.

MATH-OVERFLOW, ASK-UBUNTU, STACK-OVERFLOW and SUPER-USER capture communication within the respective expert knowledge exchange websites. On these websites, topic-specific expert questions are answered and commented on by other users. An edge (u, v, t, l) describes that at time t , user u : ($l = 0$) answers a question posed by user v ; ($l = 1$) comments on a question posed by v (requesting clarification); or ($l = 2$) comments on an answer posted by v (participates in discussion). EMAIL-EU-CORE represents a network of email communication, where an edge (u, v, t, l) represents an email sent by user u at

time t to user v , where for $l = 0$ user u and v are in the same department and where $l = 1$ represents an email between different departments. Due to data quality issues in timing information, the temporal aspect of this data was ignored. Single layer variants of the datasets above were also used in [12]. In the FACEBOOK dataset of users in the Facebook New Orleans region, an edge (u, v, t, l) describes that ($l = 0$) user v appears in user u 's friendlist or ($l = 1$) that user u posts on the wall of user v at time t [17]. Layer 0 is partially timed.

5.3 Results

Below, we report on performance experiments in single and multilayer data and the effect of different δ values. Finally the discovered motifs are evaluated.

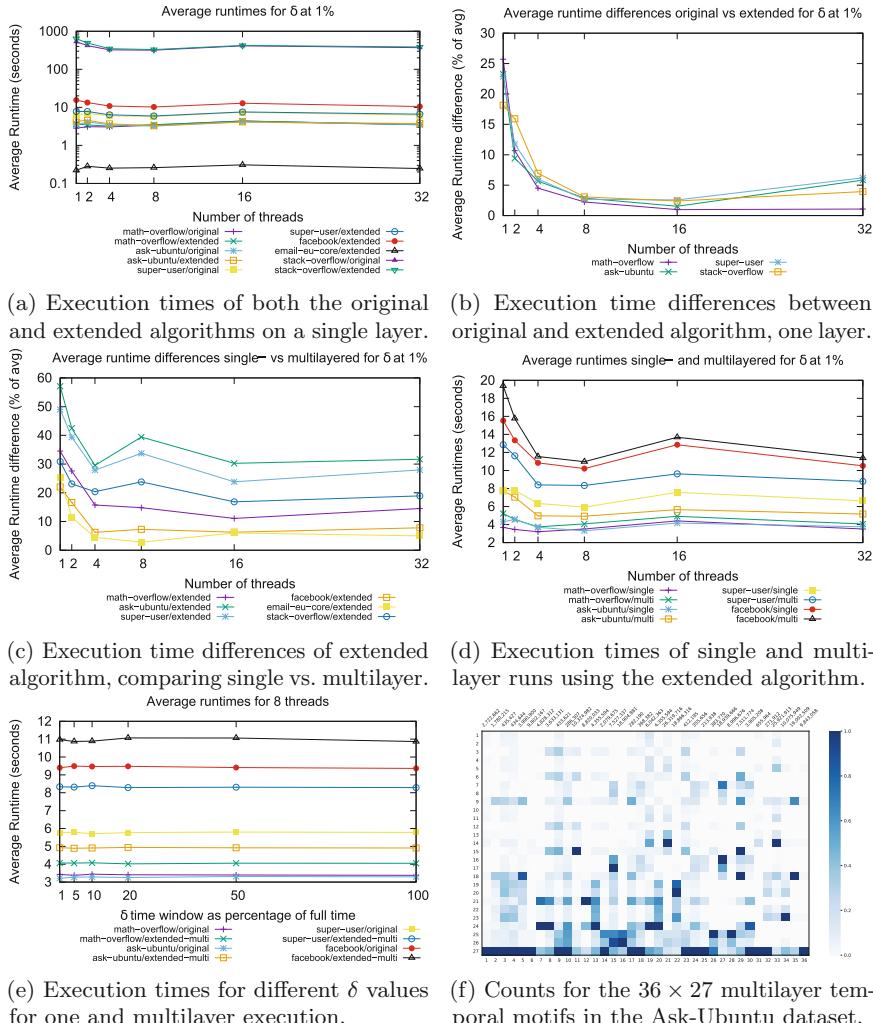


Fig. 2. Experimental results on the performance.

Performance. Figure 2a, b first compare the runtimes of the original and extended algorithms, given a single layer. With eight threads the runtime percentage difference for all fully timed datasets is at most 3.06%, warranting the use of the extended algorithm for a single layer dataset without a performance loss. Furthermore, Fig. 2a shows that for both the original and our extended algorithm the best performance is achieved at either four or eight threads. The performance improvement is most evident for the large network (STACK-OVERFLOW), which we believe is related to the maximum degree. A higher value for this metric means that different center nodes and node pairs on different threads has a more substantial effect.

Figure 2c displays the difference in execution times as a *percentage difference*, comparing single- and multilayer data using the extended algorithm. The single-layer data was constructed from the multilayer data by considering all layers to be identical, so that the same size dataset is used in comparisons. As expected, the lowest runtime differences are for the 2-layered datasets (EMAIL-EU-CORE and FACEBOOK). We also see that the four “exchange website” datasets follow the same trend with the minimum percentage difference at 16 threads. However, Fig. 2d shows that this minimum is aided by the fact that at 16 threads the algorithm encounters a performance drop, whilst the actual numerical runtime difference is similar to four threaded execution. This leads to a lower percentage difference. Therefore, the more relevant minimum is at four threads, which coincides with a minimum in runtime. Finally, we note that theoretically, the value of δ should have no impact on performance. After all, each edge is processed at most three times per connected node. Figure 2e confirms this.

Discovered motifs. Here we discuss the results of applying the multilayer temporal motif counting algorithm with a 1% time window to the 3-layer ASK-UBUNTU dataset (see Sect. 5.2). A total of $3^3 \times 36 = 972$ different motifs were found, as shown in Fig. 2f, where each column is one of the 36 motifs (counted from left to right, top to bottom in Fig. 1a) and each row denotes one of the 27 layer permutations listed in Fig. 1c. Values above columns denote the number of temporal motifs found over all layer permutations. More importantly, cells are coloured per column, proportional to the percentage of motifs of that row.

Most notably, the results show that there is heterogeneity with respect to which layers are involved in which temporal motifs. The bottom row of the figure shows that most 2, 3-user interaction takes place in layer 2 (discussion). However, this is not always the case. For example, similar motifs $M_{5,1}$ and $M_{5,2}$ in Fig. 1a both indicate back-and-forth communication between two users. Their counts (columns 25 and 26 of Fig. 2f) are similar (200 and 400 thousand times). However, taking into account the multilayer aspect, for $M_{5,1}$ all communication appears to be based on comments to questions (bottom row of Fig. 2f, permutation 27 in Fig. 1c). On the contrary, $M_{5,2}$ shows how after a question on the knowledge exchange website is answered (layer 0, first link), often comments arise from the person who asked the question to the user who answered it (layer 2, second and third link). The matrix furthermore highlights how it rarely happens that three users are all involved in answering each other’s questions (layer 0),

hence the virtually empty row 10. Concluding, we see that particular temporal motifs never occur with a certain combination of layers, particular combinations of layers never result in motifs, and some temporal motifs are overrepresented for certain combinations of layers.

6 Conclusion and Future Work

In this paper, we proposed temporal motif counting algorithms to deal with partially timed multilayer temporal networks. We found that with s edges in the motif there exist λ^s so-called δ -temporal, λ -layered motifs for each δ -temporal motif, where λ denotes the number of layers and δ the time window within which the motif occurs. Furthermore, we proposed a general methodology and implementation for partial timing. Although the multilayer aspect adds a factor of λ^2 to the computation, when the number of layers is small, the runtime of our multilayer algorithms only increases by a constant factor. For networks with a single layer, the extended algorithm is around 3% slower than the original algorithm. For networks with multiple layers the difference is 10% for two layers and 30% for three layers. Considering that we count λ^3 times more 3-edge motifs, this is acceptable. The space complexity went up from $O(k)$ to $O(\lambda k)$, with k the number of neighbours, again not affecting practical situations with small numbers of layers. We showed that the choice of time window δ has no impact on performance. Experiments on real-world social communication networks demonstrated heterogeneity with respect to the involvement of certain layers in the discovered motifs.

Whereas this paper focused on the algorithms and taming their computational complexity, in future work we plan to do an in-depth investigation of the discovered motifs. Moreover, we aim to compare the obtained motif counts with counts in null models to determine motif significance. In addition, we aim to do more experiments to better understand the effect of layer densities on performance.

References

1. Battiston, F., Nicosia, V., Chavez, M., Latora, V.: Multilayer motif analysis of brain networks. *Chaos: an Interdisciplinary J. Nonlinear Sci.* **27**(4), 047,404 (2017)
2. Benson, A.R., Gleich, D.F., Leskovec, J.: Higher-order organization of complex networks. *Science* **353**(6295), 163–166 (2016)
3. Braha, D., Bar-Yam, Y.: Time-dependent complex networks: dynamic centrality, dynamic motifs, and cycles of social interactions. In: *Adaptive Networks*, pp. 39–50. Springer (2009)
4. Gonen, M., Shavitt, Y.: Approximating the number of network motifs. *Internet Math.* **6**(3), 349–372 (2009)
5. Grochow, J.A., Kellis, M.: Network motif discovery using subgraph enumeration and symmetry-breaking. In: *Proceedings of the 11th Annual International Conference on Research in Computational Molecular Biology*, pp. 92–106. Springer (2007)

6. Kamaliha, E., Riahi, F., Qazvinian, V., Adibi, J.: Characterizing network motifs to identify spam comments. In: Proceedings of the 8th IEEE International Conference on Data Mining Workshops, pp. 919–928. IEEE (2008)
7. Kivelä, M., Porter, M.A.: Isomorphisms in multilayer networks. *IEEE Trans. Netw. Sci. Eng.* **5**(3), 198–211 (2018)
8. Kovanen, L., Karsai, M., Kaski, K., Kertész, J., Saramäki, J.: Temporal motifs in time-dependent networks. *J Stat. Mech.: Theory Exp.* **2011**(11), P11,005 (2011)
9. Leskovec, J., Sosić, R.: SNAP: a general-purpose network analysis and graph-mining library. *ACM Trans. Intell. Syst. Technol.* **8**(1), 1 (2016)
10. Marcus, D., Shavitt, Y.: Efficient counting of network motifs. In: Proceedings of the 30th IEEE International Conference on Distributed Computing Systems Workshops, pp. 92–98 (2010)
11. Milo, R., Shen-Orr, S., Itzkovitz, S., Kashtan, N., Chklovskii, D., Alon, U.: Network motifs: simple building blocks of complex networks. *Science* **298**(5594), 824–827 (2002)
12. Paranjape, A., Benson, A.R., Leskovec, J.: Motifs in temporal networks. In: Proceedings of the 10th ACM International Conference on Web Search and Data Mining, pp. 601–610 (2017)
13. Ribeiro, P., Silva, F.: G-tries: an efficient data structure for discovering network motifs. In: Proceedings of the 2010 ACM Symposium on Applied Computing, pp. 1559–1566 (2010)
14. Shahrvari, S., Jalili, S.: Fast parallel all-subgraph enumeration using multicore machines. *Sci. Program.* **2015**, 6 (2015)
15. Shellman, E.R., Burant, C.F., Schnell, S.: Network motifs provide signatures that characterize metabolism. *Mol. BioSyst.* **9**(3), 352–360 (2013)
16. Takes, F.W., Kosters, W.A., Witte, B., Heemskerk, E.M.: Multiplex network motifs as building blocks of corporate networks. *Appl. Netw. Sci.* **3**(1), 39 (2018)
17. Viswanath, B., Mislove, A., Cha, M., Gummadi, K.P.: On the evolution of user interaction in Facebook. In: Proceedings of the 2nd ACM Workshop on Social Networks, pp. 37–42 (2009)
18. Wernicke, S.: A faster algorithm for detecting network motifs. In: Proceedings of the 5th International Workshop on Algorithms in Bioinformatics, pp. 165–177. Springer (2005)
19. Wernicke, S.: Efficient detection of network motifs. *IEEE/ACM Trans. Comput. Biol. Bioinform.* **3**(4), 347–359 (2006)
20. Wernicke, S., Rasche, F.: FANMOD: a tool for fast network motif detection. *Bioinformatics* **22**(9), 1152–1153 (2006)
21. Yeger-Lotem, E., Kashtan, N., Itzkovitz, S., Milo, R., Pinter, R.Y., Alon, U., Margalit, H.: Network motifs in integrated cellular networks of transcription-regulation and protein-protein interaction. *Proc. Natl. Acad. Sci.* **101**(16), 5934–5939 (2004)
22. Zhao, Q., Tian, Y., He, Q., Oliver, N., Jin, R., Lee, W.C.: Communication motifs: a tool to characterize social communications. In: Proceedings of the 19th ACM International Conference on Information and Knowledge Management, pp. 1645–1648 (2010)



Quasi-cliques Analysis for IRC Channel Thread Detection

Jocelyn Bernard^{1(✉)}, Sicong Shao², Cihan Tunc², Hamamache Kheddouci¹, and Salim Hariri²

¹ Université Lyon 1, Villeurbanne, France

{jocelyn.bernard,hamamache.kheddouci}@liris.cnrs.fr

² University of Arizona, Tucson, USA

{sicongshao,cihantunc,salimhariri}@email.arizona.edu

Abstract. Internet Relay-Chat (IRC) is a real-time communication protocol that allows broadcasting and direct messages in the form of text. Hence, IRC has been widely used especially by hacker communities to communicate and plan malicious activities. Even though widely used for malicious intent, little research has been done on the analysis of the social network among hacker communities in IRC. Hence, it is crucial to analyze IRC communities and their connection. In this paper, we classified IRC messages based on their intent and created their communication graphs to compute metadata on the relation between hackers. For this purpose, we apply autonomic computing for IRC monitoring and data collection, perform deep learning to classify IRC messages into different threat levels, and then apply the quasi-clique model to analyze hacker social networks, and identify the hidden relations between them.

1 Introduction

Internet Relay-Chat (IRC) has become a popular and important form of real-time communication, especially among the security organizations (both malicious and non-malicious) to share knowledge and get cooperation because IRC provides many professional channels (chatrooms) to discuss various topics including security, vulnerabilities, malicious activities. Some IRC channels even contain black market where visitors can buy security exploits, hacking services, stolen credit card information, etc. [12]. Analysis of such IRC channels can help understand cybercriminals mind, behaviors, and even predict possible cyber-crime event. For example, British cybersecurity analysts identify botnet operators actively involving cyberattacks through analysis of an anonymous hacking organization in IRC channel [11].

Further research suggests that hackers often congregate with hacker organizations, most commonly in the form of IRC networks or online forums [2]. There exist different communities in IRC based hacker organizations in terms of their thread knowledge, capabilities, and crime interests. While some users and channels are interested in and experienced in carding crime, some are specialized in developing vulnerability detection, exploitation, and sharing and some focus on

disseminating hacking knowledge and recruiting new members. Using the chat logs of the IRC users and channels, we can predict cybercrime and understand the hidden relations among hackers organizations.

IRC uses a protocol that facilitates real-time text communications. Therefore, on the contrary to the off-line collection that is typically utilized by website information analysis, IRC contents must be collected in real-time and are not normally archived for later retrieval[3]. IRC provides a method of communication between users in distinct channels publicly as a broadcasted message or privately (i.e., hidden to other users). For the public messages, IRC users can also point out the receiver, which highlights the messages. This paper focuses on obtaining the user information through the public messages as finding means of receiving private messages is out of the scope of this work. Using the graph-theory approach, we can create subgraphs to understand the connections of the users based on the intent and frequency of the messages between them. Finding dense subgraphs is a well studied approach in network communities as there are applications in biological network [16] or community detection [7]. However, in this work, the question is how to use network relations between dangerous users (i.e., potentially malicious users) to understand their hacker network. To answer this question, we propose to use IRC channels to construct a graph based on the dangerousness of exchanged messages. Then, we use the quasi-clique model to compute dense subgraph. Among community enumeration models, using the results produced by quasi-cliques permit us to determine with which other user, a determined user is the most susceptible to communicate.

The rest of the paper is organized as follows: In Sect. 2, we give the terminology for the quasi-cliques briefly. Section 3 provides the related work on IRC channels and maximal clique computations. Next (in Sect. 4), we explain our autonomic computation approach for data gathering and graph composition. We provide experimental results in Sect. 5. And, we conclude the paper in Sect. 6.

2 Terminology

In this paper we consider graph $G = (V, E)$ with a set of nodes V and a set of edges E . For $v \in V$, let $\Gamma(v)$ be the set of neighbors of v in G . The *degree* of v in V is denoted by $|\Gamma(v)|$ and corresponds to the size of its neighborhood.

A *clique* is a complete sub-graph, i.e., each node of the clique is connected to the remaining nodes of the clique. A clique K is *maximal* in G if no nodes $u \in V - K$ can be added to K to create a larger clique.

A *quasi-clique* is an almost complete sub-graph. We distinguish several models of quasi-cliques. In this paper, we use the $\lambda\text{-}\gamma$ -*cliques* model [5]. Indeed we choose to constrain our quasi-cliques with a minimum node and edge connectivity. γ is a parameter estimating the minimum ratio of edges so that the quasi-clique is valid. A clique K possesses exactly $(|K| \cdot (|K| - 1))/2$ edges. A γ -clique possesses at least $\gamma \cdot ((|K| \cdot (|K| - 1))/2)$ edges, $\gamma \in [0; 1]$. λ is a parameter estimating the rate of connectivity of every node within the quasi-clique. In a

clique K , every node $v \in V(K)$ has exactly $|K| - 1$ neighbors. Every node of a λ -clique must have at least $\lambda \cdot (|K| - 1)$ neighbors in the clique to be valid, $\lambda \in [0; 1]$.

Figure 1 present various λ - γ -cliques where the **colored** node represents the node which has the smallest number of neighbors and consequently the node used to defined the λ parameter of the quasi-clique. The dotted edges represent the edges which are missing to complete the clique. The Fig. 1a is a λ_a - γ_a -clique with $\lambda_a = 1.0$ and $\gamma_a = 1.0$. The Fig. 1b is a λ_b - γ_b -clique with $\lambda_b = \frac{3}{4} = 0.75$ and $\gamma_b = \frac{8}{10} = 0.8$. The Fig. 1c is a λ_c - γ_c -clique with $\lambda_c = \frac{2}{4} = 0.5$ and $\gamma_b = \frac{8}{10} = 0.8$.

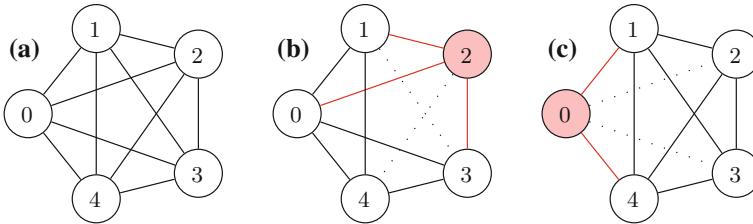


Fig. 1. Presentation of various λ - γ -cliques with Fig. 1a a 1-1-clique, Fig. 1b a 0.75-0.8-clique and Fig. 1c a 0.5-0.8-clique.

3 Related Work

IRC is a popular communication method, especially in the cyber-domain. In order to provide messaging capabilities, IRC requires a server for the users to connect through a protocol that facilitates real-time text communications. IRC has been traditionally utilized for legitimate functions, but it has also been extensively used by hackers, anonymities, and terrorists over the years [2]. By the help of many hacker channels, users can hide in the IRC channels to plan and commit cybercrime [3]. Therefore, developing the methods for knowing hacker social networks, as well as collection and identification of threat messages are greatly needed. In previous studies, Benjamin et al. used the chat-logging listeners to collect IRC message data [2]. These listeners simply use basic IRC client mechanism and passively log the data. For content analysis, research mainly relies on statistics and natural language processing to understand discussion in virtual communities. Garas et al. analyzed the emotional expressions (positive, negative, neutral) of users and revealed a remarkable persistence both for individual users and channels [6]. Benjamin et al. proposed a framework to identify potential IRC threats based on a list of keywords [3]. For IRC social network analysis, a social network approach was developed for online language variation and change through qualitative and quantitative analysis in [10]. This research revealed a highly structured relationship between participants social positions in

a channel and their use of linguistic variants. Mutton [9] used a heuristic analysis of events to produce a visualization of the inferred social network structure in IRC channel, highlighting connectivity, clustering the relationships between IRC users.

Maximal Clique Computation (MCE) is a well-known problem which consists of enumerating all maximal cliques of a given graph. Some algorithms permit to solve this problem, [4, 15] are the main reference. In [14], authors propose to distribute *Tomita* with *Map-Reduce* paradigm. Each node makes computation according to their neighborhood to find maximal cliques. We adopt this algorithm to compute quasi-cliques [5].

4 Data Gathering

4.1 Autonomic IRC Bot for Message Collection and Classification

For the IRC message collection and classification, we created an Autonomic IRC bot as shown in Fig. 2. Our autonomic IRC bot includes the features of robust continuous monitoring, comprehensive information collection, pre-processing, and classification in real-time. With these capabilities, the bot monitors the IRC channel and transforms the unstructured IRC message to structured data as CSV files in the following format: ***Threat Level + Username + Chat Content + Date + Time***.

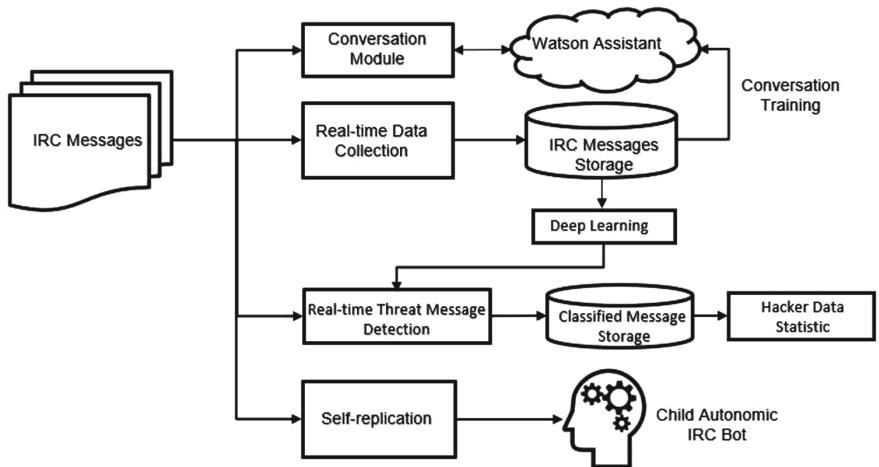


Fig. 2. The architecture of autonomic IRC bot

IRC Channel Monitoring and Data Collection Multiple challenges need to be overcome when we monitor IRC channels, especially for cybersecurity related ones. First of all, in some critical channels, especially hacker channels, the IRC channel operators remove the user and even block the IP if the user is identified as an inactive user or as a bot. Therefore, we created a conversation module providing basic respond for the chat for ensuring that the bot can escape the identification by IRC channel operators. With the IBM Watson Assistant service [1], we built a machine learning model that understands the chat input and respond to IRC users in a way that simulates conversations between humans. The procedure of conversation implementation is as follows: (1) We create a workspace which is a container for dialog flow and training data. (2) The workspace is configured by setting up the training data collected from the chat messages in monitored channels with the easy-to-use graphical environment. We transform the chat messages into the Intent, Entity, and Dialog content in workspace for training the conversation model. The Intent is the goal that we anticipate IRC users will have when they chat with our bot. For example, we define an intent named DDoS that discusses questions about DDoS attack. For each intent, we add training samples that reflect what IRC users might ask for the information they need; e.g., “*What’s the tool available to perform a DDoS attack?*”. The entity represents a term, an object or a data type that provides context for an intent. We identify the entity mentioned by IRC user’s input, such as, an entity might be a city name that helps the bot to distinguish which the timezone the IRC users wants to know local time for. And, a nature language classifier is automatically generated for the workspace and is trained to understand the types of inputs of IRC users. The dialog is a branching conversation flow that defined intents and entities. Through the use of the dialog tool for building dialog flow which incorporates IRC user’s intents and entities, we can add a branch to process every intent that we want the service to respond. (3) After the creation of the chat model for IRC monitored channels, we integrate Watson Assistance into the conversation module of the bot.

In addition, many hacker channels publish self-signed certificate and no longer allow any non-SSL connections to their network. To overcome this issue, the functions that can trust all the self-signed certificate and specify the port number for SSL/TLS connections are added to the IRC bot. It is also possible that the cyber-criminals create a temporary channel where these users can disseminate hacking knowledge or share malicious tool, and even launch cyberattacks such as Denial of Service (DoS). Hence, to continuously track such activities, a self-replication module is developed allowing parent bot to generate a new (child) bot that inherits all the capability.

Deep Learning of IRC Threat Message Detection To determine the threat level of the IRC messages, we use CoreNLP provided by Stanford NLP team [8]. Using the Recursive Neural Tensor Network (RNTN) [13], the IRC bot can automatically distinguish between normal messages and threat messages, and further classify the threat level of the IRC messages. For the machine learn-

ing model of RNTN, we use the IRC messages collected from IRC channel to train the model. We classify each IRC message into three levels: Normal, Warning, and High with each level is given the scores of 0, 1, 2, respectively. Normal (level 0) represents normal messages being interactive with other contacts. Warning level (level 1) indicates potential risk that has been detected because of the use hacking terms. High level (level 2) denotes the user who sends a message that appears this user's own hacking behaviors or intentions to perform malicious activities. The labeling rule for the current node score depends on the threat level of the whole phrase that current node dominates under the parse of threat tree as shown in Fig. 3.

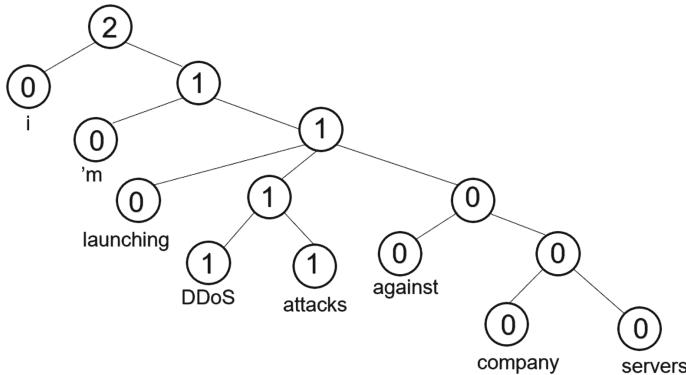


Fig. 3. An example of our labeling rule, from normal to high (0, 1, 2), at every node of the threat tree construction

The procedures of RNTN-based IRC threat message detection method are described as follows: (1) Tokenize the IRC message into a sequence of words which are represented as numeric vectors. (2) Generates the lemmas (base forms) for words. (3) Tag words with part of speech tagger (POS) (4) Parse the IRC message into its constituent phrases and build a syntactic threat tree. (5) Classify the IRC message threat level using the recursive neural tensor network. All the nodes of the IRC message, especially, the root node, are given a threat level score.

After the creation of the RNTN-based IRC threat message detection model, we integrate the model to IRC bot that can be deployed in the IRC channel to monitor, collect and classify threat messages in real-time. Figure 4 shows an example of a hacking topic that was discussed in the monitored IRC hacker channel. The recall and precision and percentage of each threat level in our test dataset are shown in Table 1. The recall achieved 90.15%, 86.93%, and 81.82%, with normal level, warning level, and high level messages. The precision results of normal level, warning level, and high level are 98.69%, 71.12%, and 39.13%, respectively. The decreases of the precision and recall with high threat level are predictable results due to the reason: the number of training samples of high threat message is significantly less than normal and warning messages.

After we capture and label more high threat messages, the performance will be improved. Table 2 shows the number of each threat level IRC messages in 5 different channels.

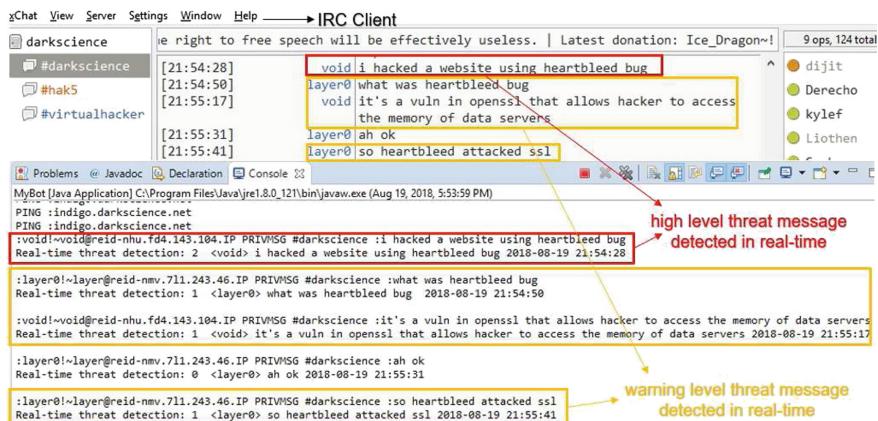


Fig. 4. An example of real-time IRC message classification in the monitored channel

Table 1. Recall and precision of each threat level in the test dataset.

	Normal (root node)	Warning (root node)	High (root node)
Percentage in test set	0.7929	0.1811	0.0260
Recall	0.9015	0.8693	0.8182
Precision	0.9869	0.7112	0.3913

Table 2. Threat messages distribution in 5 different IRC channels.

	Normal (root node)	Warning (root node)	High (root node)
1	36 940	3 379	35
2	6 554	775	12
3	12 276	1 261	16
4	33 486	3 197	58
5	3 760	365	5

4.2 Graph Transformation

Using the data collected by the autonomic IRC monitoring tool, we can create a graph $G = (V, E)$ with a set of nodes V and a set of edges E . The nodes corresponding to the users and an edge $(u, v) \in E$ existing between two users when one user u has spoken on the channel and user v is connected. The edges are labeled and weighted. The label $(u, v)_x$ of an edge corresponding to the dangerousness of the message, $(u, v)_0$ corresponding to the normal message, $(u, v)_1$ to warning threat level message and $(u, v)_2$ to high threat level message. Consequently, there can be 3 edges between two users. The weight $w((u, v)_x)$ of an edge corresponding to the number of messages that u has sent when v is connected. All exchanged messages are counted for 1 except when u speaks directly to v on the channel through mentioning v 's nickname (a user's IRC client receives a special remind when other user mentions this user's nickname). In that case, we add a weight of 3 for the edge between u and v and 1 for edges between u and other connected users.

We evaluate five graphs from the IRC channel. We compute the number of each edge by each type, the maximum and average degree of nodes (the minimum degree is 1 for each graph). We finally evaluate the diameter, the longest shortest path, and the local clustering coefficient that represents the connectivity of the graph.

Table 3. Statistics of graph from IRC channels' data.

Graphs	Channels				
	1	2	3	4	5
$ V $	605	712	435	481	528
$ msg $	9 726 448	1 670 016	1 360 963	6 391 941	669 360
$ E_0 $	41 475	44 117	23 021	37 940	23 265
$ E_1 $	23 035	21 023	11 051	21 868	9 983
$ E_2 $	4 033	1 827	1 223	5 085	513
$ \Gamma _{avg.}$	113.29	94.05	28.86	12.76	65.81
$ \Gamma _{max}$	1 426	1 464	945	1 389	815
Diameter	3	4	4	3	4
Local clust. coef.	0.88	0.45	0.49	0.70	0.43

The Table 3 presents the statistics of the graphs drawn from IRC channels. From the messages, we compute the different edges and their weight between the nodes. We notice that the maximum degree is always superior to the number of nodes in the graph. The fact can explain that there can be maximum 3 edges between each node. Consequently, the diameter of the graphs are short.

4.3 Quasi-clique Computation

To evaluate the malicious conversation between users, we enumerate all quasi-cliques. As in PEKO [14], the quasi-clique computation is distributed. Each node made a computation according to λ , γ and its neighbors (and the neighbors of its neighbors, which is enough for $\lambda \geq \frac{1}{2}$). The enumeration of quasi-cliques permits to give a probability of exchanges between users even if they do not share an edge. Indeed, if users share message when we do not collect information we miss this information. Moreover, in IRC channels there are private messages that we can not collect, they are hidden from other users; hence it is not the focus of this work.

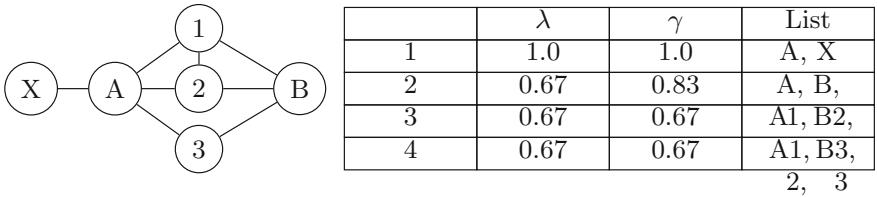


Fig. 5. Enumeration of quasi-cliques in a graph with $\lambda \geq \frac{2}{3}$ and $\gamma \geq \frac{2}{3}$. Quasi-Cliques found and their score are presented in the table.

Figure 5 presents a graph and the list of quasi-cliques found if we enumerate all the quasi-cliques with $\lambda \geq \frac{2}{3}$ and $\gamma \geq \frac{2}{3}$. In that case, we found 4 quasi-cliques. Each of them is maximal, we can not add a new node to respect the constraint of parameters λ and γ . For each quasi-clique, we give their respective λ and γ values. These values are higher if the quasi-clique is more connected. We notice that, even if they do not share an edge, the nodes A and B are present in 3 common quasi-cliques while X , that share an edge with A is only present in 1 quasi-cliques.

Table 4 presents matrices of co-occurrence of nodes in enumeration of quasi-cliques. For each node n , corresponding to a line, we look for each other node v the number of time they appeared in the same quasi-clique K . We provide 2 matrices. The first matrix is computed by strict co-occurrence, corresponding to Eq. (1). The second matrix corresponds of ponderate co-occurrence as in Eq. (2), the value of the γ parameter of each quasi-clique is used as reference, like that quasi-cliques which are more connected provide higher score.

$$matrix_1[n, v] = \frac{\sum_{\{n, v\} \in K} 1}{\sum_{n \in K} 1} \quad (1) \quad matrix_2[n, v] = \frac{\sum_{\{n, v\} \in K} \gamma_K}{\sum_{n \in K} \gamma_K} \quad (2)$$

In the case of the node A , we notice that the node B has the highest co-occurrence score, corresponding to the fact that there is a lot of common quasi-cliques between them. The second matrix, which is ponderated by the γ value

of each quasi-clique, sort the nodes according to the connectivity: for the node 1 the value of nodes 2 and 4 are the same in the first matrix but not in the second: the score of the node 2 is higher than the node 3. This is a consequence of the fact there is an edge between 1 and 2 but not between 1 and 3. Indeed, the value of the γ parameter of the 2nd quasi-clique is higher than the value of the γ parameter of the 3rd quasi-clique. Consequently the value of the node 2 is higher than the value of node 3.

Table 4. Table of co-occurrence of nodes. The left table presents scores with strict co-occurrence. The right table presents scores with co-occurrence ponderate by the γ value in the quasi-clique.

	A	B	1	2	3	X
A	0.00	0.75	0.50	0.50	0.50	0.25
B	1.00	0.00	0.67	0.67	0.67	0.00
1	1.00	1.00	0.00	0.50	0.50	0.00
2	1.00	1.00	0.50	0.00	0.50	0.00
3	1.00	1.00	0.50	0.50	0.00	0.00
X	1.00	0.00	0.00	0.00	0.00	0.00
	A	B	1	2	3	X
A	0.00	0.68	0.47	0.47	0.42	0.32
B	1.00	0.00	0.69	0.69	0.62	0.00
1	1.00	1.00	0.00	0.56	0.44	0.00
2	1.00	1.00	0.56	0.00	0.44	0.00
3	1.00	1.00	0.50	0.50	0.00	0.00
X	1.00	0.00	0.00	0.00	0.00	0.00

5 Results

Table 5 presents the main results after quasi-clique enumeration. 2nd and 3rd lines present the minimum λ and γ define for the enumeration which are defined with the size of the maximal clique to be sure to have at least some quasi-cliques but not too much. Line 5 presents the number of quasi-clique found. Lines 6 and 7 present the mean λ and γ for each quasi-clique while line 8 presents the size of maximal quasi-clique (clique in parentheses). Last line present the number of relations which are hidden and suspicious: there is no edge between two nodes, but they are present in more than half of the quasi-clique of the other node.

We notice even if there is a small number of cliques initially, we can produce a lot of quasi-cliques when our computation stops. We can use the score of each relation to sort the most probable relations for a node and determine hidden relation, in our case when there are no edges but a score greater than 50%.

Table 5. Results for quasi-clique enumeration

Graph	1	2	3	4	5
λ_{min}	0.9	0.7	0.7	0.9	0.7
γ_{min}	0.95	0.7	0.7	0.9	0.7
$ clique $	353	1 022	685	3 078	275
$ quasi-clique $	71 565	2 101 860	1 936 878	130 232	3 758
λ_{mean}	0.912	0.733	0.703	0.928	0.762
γ_{mean}	0.988	0.817	0.778	0.976	0.903
$ K _{max}$	23 (21)	11 (6)	11 (6)	18 (15)	8 (5)
<i>hidden</i>	0	9	16	0	1

Our method permit to sort relations in a graph by a score function. With a determined threshold, we can find *hidden relations*. To improve our results we can:

- use the weight of the arc in our score or to determine users' profile with the fraction of malicious and normal messages
- evaluate an ideal value for parameters with the characteristics of the graph
- evaluate the quality of our approach

6 Conclusion

In this paper, we proposed the evaluation of interactions between hackers. To solve our problem, we collect messages from IRC channel using our autonomous IRC bots. Next, we determine the threat level for each message. Using this information we create graphs for each channel and enumerate all maximal quasi-cliques. We use co-occurrence to determine the connection of each user with respect to other users based on their communication frequency and intent. Next, we compute metadata using these graphs in order to answer if we can determine which users are the most active ones with the highest connections and highest importance value compared to the others.

Acknowledgements. This work is partly supported by the Air Force Office of Scientific Research (AFOSR) Dynamic Data-Driven Application Systems (DDDAS) award number FA9550-18-1-0427, National Science Foundation (NSF) research projects NSF-1624668 and SES-1314631, and Thomson Reuters in the framework of the Partner University Fund (PUF) project (PUF is a program of the French Embassy in the United States and the FACE Foundation and is supported by American donors and the French government).

References

1. Ibm watson Assistant Service. <https://www.ibm.com/watson/services/conversation/> (2017). Accessed Dec 2017

2. Benjamin, V., Chen, H.: Securing cyberspace: Identifying key actors in hacker communities. In: 2012 IEEE International Conference on Intelligence and Security Informatics (ISI), pp. 24–29. IEEE (2012)
3. Benjamin, V., Li, W., Holt, T., Chen, H.: Exploring threats and vulnerabilities in hacker web: forums, irc and carding shops. In: 2015 IEEE International Conference on Intelligence and Security Informatics (ISI), pp. 85–90. IEEE (2015)
4. Bron, C., Kerbosch, J.: Algorithm 457: finding all cliques of an undirected graph. *Commun. ACM* **16**(9), 575–577 (1973)
5. Brunato, M., Hoos, H.H., Battiti, R.: On effectively finding maximal quasi-cliques in graphs. In: International conference on learning and intelligent optimization, pp. 41–55. Springer (2007)
6. Garas, A., Garcia, D., Skowron, M., Schweitzer, F.: Emotional persistence in online chatting communities. *Sci. Rep.* **2**, 402 (2012)
7. Kim, J., Lee, J.G.: Community detection in multi-layer graphs: a survey. *ACM SIGMOD Rec.* **44**(3), 37–48 (2015)
8. Manning, C., Surdeanu, M., Bauer, J., Finkel, J., Bethard, S., McClosky, D.: The stanford corenlp natural language processing toolkit. In: Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations, pp. 55–60 (2014)
9. Mutton, P.: Inferring and visualizing social networks on internet relay chat. In: Proceedings of the Eighth International Conference on Information Visualisation, 2004. IV 2004, pp. 35–43. IEEE (2004)
10. Paolillo, J.C.: The virtual speech community: social network and language variation on irc. In: Proceedings of the 32nd Annual Hawaii International Conference on Systems Sciences, 1999. HICSS-32, pp. 10–pp. IEEE (1999)
11. Schone, M., Esposito, R., Cole, M., Greenwald, G.: War on anonymous: British spies attacked cybercriminals, snowden docs show. NBC News (2014)
12. Shao, S., Tunc, C., Satam, P., Hariri, S.: Real-time irc threat detection framework. In: 2017 IEEE 2nd International Workshops on Foundations and Applications of Self* Systems (FAS* W), pp. 318–323. IEEE (2017)
13. Socher, R., Perelygin, A., Wu, J., Chuang, J., Manning, C.D., Ng, A., Potts, C.: Recursive deep models for semantic compositionality over a sentiment treebank. In: Proceedings of the 2013 conference on empirical methods in natural language processing, pp. 1631–1642 (2013)
14. Svendsen, M., Mukherjee, A.P., Tirthapura, S.: Mining maximal cliques from a large graph using mapreduce: tackling highly uneven subproblem sizes. *J. Parallel Distrib. Comput.* **79**, 104–114 (2015)
15. Tomita, E., Tanaka, A., Takahashi, H.: The worst-case time complexity for generating all maximal cliques and computational experiments. *Theor. Comput. Sci.* **363**(1), 28–42 (2006)
16. Xu, C., Su, Z.: Identification of cell types from single-cell transcriptomes using a novel clustering method. *Bioinformatics* **31**(12), 1974–1980 (2015)



Triad-Based Comparison and Signatures of Directed Networks

Xiaochuan Xu and Gesine Reinert^(✉)

Department of Statistics, University of Oxford, Oxford, UK
reinert@stats.ox.ac.uk

Abstract. We introduce two methods for comparing directed networks based on triad counts, called *TriadEuclid* and *TriadEMD*. *TriadEuclid* clusters the Euclidean distance between triad counts, whereas *TriadEMD* is an adaptation of *NetEMD* for directed networks. We apply both methods to cluster synthetic networks, a set of web networks including google, twitter, peer-to-peer, amazon, slashdot and citation networks, as well as world trade networks from 1962–2000. Furthermore, we find signature triads and signature orbits for each type of networks in our data, which show the main triad and orbit contributions of the networks when comparing them to the other networks in the respective data set.

Keywords: Network comparison · Triad · Signature triad · Signature orbit

1 Introduction

Networks as representation of complex interaction dataset have become increasingly common. Friendships, world trade flow and protein-protein interactions can all be described as networks [14]. A key question to be addressed is network comparison, see for example [1, 15, 25, 26]. Subject specific network comparison may provide new insights; for example, comparing protein-protein interaction networks may increase our understanding of evolutionary processes [1].

Network comparison is addressed in different ways. In machine learning, graph kernels are used to obtain classifiers to predict the class membership of networks. There are many applications in computer vision, personalized medicine e.g. [3] and drug discovery e.g. [22]. Other computational algorithms are based on network alignment [9, 11, 15] which can be quite computer-intensive. Instead algorithms which compare counts of small subgraphs have become popular, such as [17] and *NetEMD* [24]. Network comparison based on small subgraphs is motivated by the observation that many real networks contain some character-

Electronic supplementary material The online version of this chapter (https://doi.org/10.1007/978-3-030-05411-3_48) contains supplementary material, which is available to authorized users.

istic small subgraphs, sometimes called *motifs*, which relate to the function of the network, see [13]. In [13], motifs are introduced as over-or under-represented subgraphs, where the over-or under-representation is usually judged by comparison to an underlying network null model. When there is no suitable underlying network model available, often a configuration model is used as null model. The significance of motifs depends crucially on the underlying null model, see [16].

In [20] a non-parametric method based on subgraph counts of up to 4-node subgraphs is developed, but excluding graphlets with bi-directional edges. Their two main methods are *DGCD129*, including all directed graphlets on 2–4 nodes, and *DGCD13*, which uses only directed graphlets on 2–3 nodes and is deemed superior to *DGCD13* in [20]. The fact that directed graphlets contain more information than undirected graphlets has been confirmed in [2], where the graphlet degree distribution GDD is generalised to cluster directed networks, using graphlets on up to 5 nodes. The underlying GDD agreement measure for undirected networks is known to suffer from a pronounced dependence on the number of nodes though, see [18], and is not recommended for comparing networks with very different numbers of nodes.

In the social network analysis paradigm, directed triads are often used as unit of information, see for example [7, 23] and [5]. A triad is a directed graph composed of 3 nodes. Taking account of the different directions and configurations, there are 16 types of triads, of which 13 are connected. In [12], these 13 triads in networks are attached a significance score which is obtained by comparison to a configuration model, yielding a vector of length 13 which is called the *Triad Significance Profile* (TSP). The TSP again depends crucially on the underlying null model.

In this paper we propose two different methods for network comparison which are both based on counting the different triads as induced sub-graphs of networks. Both methods are non-parametric; instead of requiring a null model, they crucially depend on the other networks in the set of networks to be compared, as different features may be distinguishing for different comparisons. The first triad method, which we call *TriadEuclid*, uses the Euclidean distance between the count vectors and hierarchical clustering to group networks. The second method, *TriadEMD*, is a generalization of *NetEMD*. For the comparison of undirected networks of different sizes and densities, *NetEMD* outperforms other graphlet-based methods on some test data sets, see [24]. This paper provides an extension to directed networks.

Here, network comparison is used first, to classify networks according to generating mechanism, and second, to recover the time-ordering of networks. We test *TriadEuclid* and *TriadEMD* on three data sets and compare them to *DGCD13* and *DGCD129* from [20] using their default parameter settings.

The first data set a synthetic data set of networks generated under three different models, with different numbers of nodes and different densities. All four methods cluster these networks flawlessly according to generating mechanism.

The second data set is a set of 31 sparse directed social networks from [10]. The networks in the different clusters differ considerably in size and density, and hence here the focus is whether *TriadEuclid* and *TriadEMD* make use

of information which is not covered by nodes and degree distributions already. For this task, *TriadEuclid* outperforms *TriadEMD* as well as *DGCD13* and *DGCD129*.

The third case study concerns dense World Trade networks from [6]; here the task is to recover the time ordering of the networks. This data set is analysed first using a single-layer analysis which compares aggregated yearly trade networks, and then using a multi-layer approach which compares the trade networks for different commodities separately. For this task, *TriadEMD* outperforms *TriadEuclid* as well as *DGCD13* and *DGCD129*.

In all cases, both *TriadEuclid* and *TriadEMD* pick some topological structure beyond network size and density. This topological structure is described using triad and orbit counts. Motivated by the search of network motifs without underlying network models, we also identify the triads and orbits which contribute most to the dissimilarity scores of a network compared to all other networks in the collection of networks which is to be clustered. We call these *signature triads* and *signature orbits*. Thus our notion of signatures depends on the networks which the network in question is compared against.

This paper thus has two main contributions. Firstly, it provides two new methods for network comparison, which outperform the often used *DGCD* methods. Secondly, it adds to the discussion on motifs by introducing signatures, which can be viewed as context-dependent motifs. The context is given by the networks which are involved in the network comparisons.

The paper is structured as follows. Section 2 gives some notation, reviews *NetEMD*, and details the performance measures which are used in this paper. Section 3 introduces *TriadEuclid* and *TriadEMD* as well as the signature calculations. Section 4 gives the results for synthetic networks, Sect. 5 shows the results for the sparse directed real-world networks, and Sect. 6 considers the trade networks. The results are discussed in Sect. 7. More details of the results are found in the Supplementary Material; together with the R code for this paper it is available at www.stats.ox.ac.uk/all-people-gesine-reinert-network-comparison/.

2 Background

NetEMD. In this paper all directed networks $G = (V, E)$ are simple, finite, and subgraph counts are understood to be induced subgraph counts. We compare two networks using *NetEMD*, a pseudo-distance which is based on the Earth Mover's Distance (EMD, also known as the 1st Wasserstein metric). The EMD between two probability distributions p and q on the real line is given by $EMD(F, G) = \int_{-\infty}^{\infty} |F(x) - G(x)|dx$, where F and G are the cumulative distribution functions of p and q respectively, see for example [19] and [21].

In [24], the EMD is used to construct a score called *NetEMD* which compares two undirected networks through their distributions of network features, such as the degree distribution or the distribution of the number of triangles which a node belongs to. When the aim is to compare two finite networks G and G' , and $p_t(G)$ and $p_t(G')$ are the distributions of feature t on G and G' which we assume to have non-zero variance, then we re-scale p_t and q_t so that they have

empirical variance 1; denote the re-scaled distributions by \bar{p} and \bar{q} . The NetEMD pseudo-distance between two networks G and G' based on feature t is defined as

$$\text{NetEMD}_t(G, G') = \inf_{c \in \mathbb{R}} (\text{EMD}(\bar{p}(.) + c), \bar{q}(.))). \quad (1)$$

Taking the infimum over all $c \in \mathbb{R}$ ensures that $\text{NetEMD}_t(G, G')$ is location-invariant; a proof as well as a calculation example can be found in the Supplementary Material. This property implicitly adjusts for different expectation values for different networks to render the method robust for comparing networks of different size and densities. In [24], two networks G and G' are compared using a set of features t_1, \dots, t_m ; the corresponding NetEMD is then the average of $\text{NetEMD}_{t_1}(G, G'), \dots, \text{NetEMD}_{t_m}(G, G')$. In this paper we adapt NetEMD to directed networks and use counts of triads as features; we call the resulting method *TriadEMD*.

Evaluation of the output. We use three different methods to assess the similarity between a partition which is obtained through a clustering method and the true assignment to clusters. The first method is the Adjusted Rand Index (ARI) from [8] which compares two partitions. If the networks are time-ordered, then how well the clustering reflects the time-ordering is assessed using the second method, which is based on two nearest neighbour scores. The third method is a Monte Carlo test to assess whether the method captures topological information beyond size and density of the networks.

To assess how well the time ordering of networks is recovered, we use two quality scores. The first score, a Nearest Neighbour Score (NNscore) which ranges from 0 to 1, is constructed as follows. First we describe it for a time series of items. For a given $n \times n$ dissimilarity matrix, we start with $\text{NNscore} = 0$. For each item in the dissimilarity matrix, we find the item with shortest dissimilarity to the chosen item (ties are broken at random). If the shortest dissimilarity neighbour of an item is one of its two true neighbours, then we count it as correct and add $\frac{1}{n}$ to NNScore (otherwise we count it as false). Our second quality score, the Nearest Two Neighbour Score (N2Nscore) considers the closest two neighbours. Note that in a time series of networks, with time ordering as ground truth, the first and last item only have one neighbour. For a given $n \times n$ dissimilarity matrix, the N2N score is calculated like the NNscore but replacing the second step by finding the two items with shortest dissimilarity to the chosen item (ties are broken at random). For the first and the last item, only find the item with shortest dissimilarity to the chosen item. If the two (or one) shortest dissimilarity neighbours of the selected item are its two true neighbours, then we add $\frac{1}{n}$ to N2NScore . The N2NScore is thus bounded above by the NNscore.

In order to see whether our method captures topological information beyond than density and size of the networks we apply a Monte Carlo Test [4], as follows. We use the ARI as test statistics and test the null hypothesis that an ARI at least as high as the one observed is consistent with the assumption that the underlying network is random (in the sense of an ER network or a configuration model). We simulate many networks under the null hypothesis (which is either an ER graph or a configuration model); the p -value of the Monte Carlo test is the proportion of (simulated+observed) test statistic which are at least as extreme

as the observed test statistic. If we can reject the null hypothesis then there is evidence that the network comparison methods pick up information from the network which goes beyond the information used to build the random networks under the null hypothesis.

3 Triad Methods, Signature Triads and Signature Orbits

TriadEuclid combines the Euclidean distance between the feature vectors with standard hierarchical clustering to cluster networks using features based on the triad census data. The triad censuses are induced sub-network counts of triads. The 16 different types of triads are shown in Fig. 1a. We exclude the disconnected triads 1, 2 and 5.

For a directed network G , we inspect all of its induced sub-networks with three nodes and count the different triads on these sub-networks. On any set of three nodes there will be exactly one triad from the set of 16 triads present. Denote by $n_i(G)$ the number of induced sub-graphs in G which match triad i , for $i \in \mathcal{T} := \{1, \dots, 16\} \setminus \{1, 2, 5\}$. Set $N(G) = \sum_{i \in \mathcal{T}} n_i(G)$ and

$$T(G) = (f_i(G), i \in \mathcal{T}) \quad \text{where} \quad f_i(G) = n_i(G)/N(G). \quad (2)$$

Then T maps the space of finite simple directed networks to the feature space $([0, 1] \cap \mathbb{Q})^{13}$. *TriadEuclid* gives a pseudo-distance between two different networks by calculating the Euclidean distance between their feature vectors. For a finite set of networks $\{G_i, i \in I\}$ which are to be clustered, their pairwise *TriadEuclid* dissimilarities constitute the entries of a dissimilarity matrix D . Then, we cluster the networks $\{G_i, i \in I\}$ based on the dissimilarity matrix D using hierarchical clustering according to a pre-set number of clusters.

Following [24], we further classify the nodes into different orbits according to automorphisms. An orbit is a specific location of a node in a triad. Figure 1b shows the 30 different triad orbits. For example, in a bi-directed 2-star, the centre node corresponds to orbit 27, whereas the two nodes which are not connected to each other correspond to orbit 26. They have the same orbit because they could be interchanged without changing the triad.

Analogous to the degree distribution, *TriadEMD* is based on degree distributions of orbits in triads. For each node in the graph, we count the *orbit degree* of the node. The orbit degree k_i of orbit i of a node is the number of appearances of orbit i that the node is involved in. The orbit i degree distribution $P(k_i)$ of a network is then defined to be the fraction of nodes in the network with orbit i degree k_i . A worked example can be found in the Supplementary Material. Given two networks, we calculate *NetEMD* between their triad orbit degree distributions of the 30 different orbits and take their average as their dissimilarity *TriadEMD*.

To calculate *signature triads*, we disentangle *TriadEuclid* as follows. For each network $G_i, i \in I$ in the collection of interest, we record the index $e_{i,j}$ in \mathcal{T} of the triad (or triads) which give the highest contribution to the $\text{TriadEMD}(G_i, G_j)$ -score, for each $j \in I \setminus \{i\}$. The index is given a sign; it is positive when the

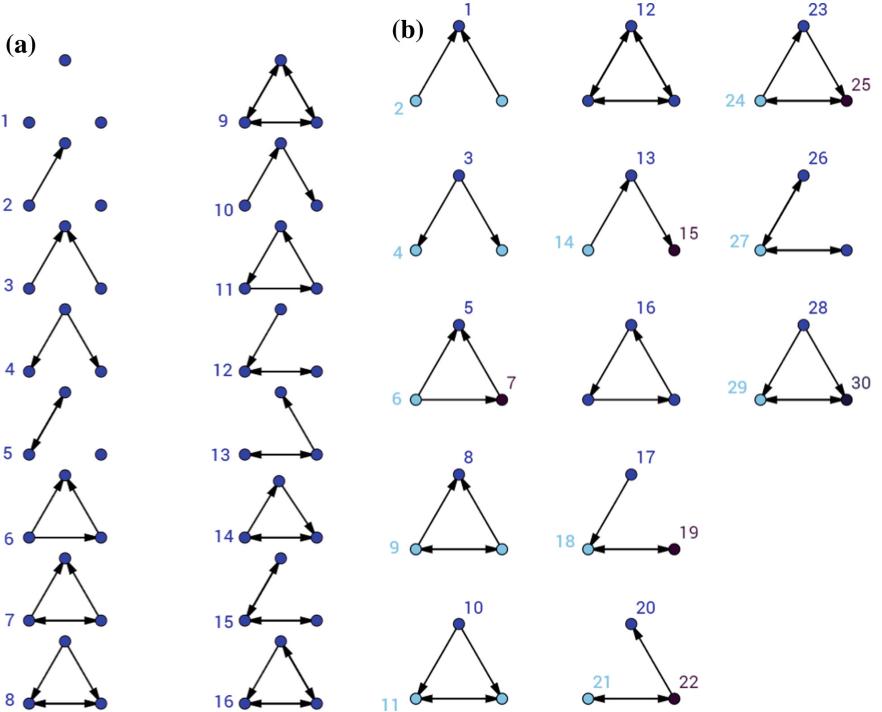


Fig. 1. (a) 16 different triads and their labels. (b) 30 triad orbits. The different shades in each triad represent different automorphism orbits, numbered from 1 to 30.

network G_i over-represents the corresponding triad, and negative when it under-represents the corresponding triad. We then choose the triad index for G_i with the largest frequency of appearance among $e_{i,j}, j \in I \setminus \{i\}$. If there is a tie, then all indices which achieve the highest frequency are reported. We call the corresponding triad the *signature triad* for network G_i . Similarly, disentangling *TriadEMD* gives *signature orbits* for networks. The signature triads and signature orbits are thus calculated relative to the other networks in the set of networks to be clustered, and are hence cluster-dependent.

4 Case Study I: Synthetic Networks

To test our methods we use the R package *igraph* to generate 27 synthetic directed network which are versions of the BarabásiAlbert (BA) models, the ErdsRnyi (ER) model, and the 2-D geometric (GEO) model. For each model, we simulated three networks with different sizes (500 and 1000 nodes) and densities, with 3 replicas each. The detailed parameter settings for the directed networks are;

for BA: 1 edge or 2 edges added for every new node;

for ER: edge probability 0.1 or 0.2 for 500 nodes, $0.1 * 499/999$ for 1000 nodes; for GEO: radius r such that $\pi r^2 = \frac{1}{2} \times (\text{number of nodes}) \times (\text{ER edge probability})$.

The BA models start with a single edge and are naturally directed networks with a single direction per edge. For the ER networks, each directional edge has probability p of appearing, and bi-directional edges are possible. For the GEO networks, first an undirected network is generated, and then two directed edges are created for each undirected edge, one in each direction. For all of these networks, replicas are given subscripts indicating their parameter settings.

Table 1. Signature Triads of Synthetic Networks

Network	Signature Triads No.	Signature Orbits No.
All BA	3	6, 12, 19
ER on 1000 nodes	10	1, 4, 16
ER on 500 nodes	-15	11, 13, 19, 26
GEO	9, 15	4, 12, 19



Fig. 2. The signature triads for Case Study I

Both *TriadEuclid* and *TriadEMD* classify all of the networks correctly according to generating model, so all the Adjusted Rand Indices and the nearest neighbour scores are 1. Table 1 shows the labels of the signature triads of different types of networks, and Fig. 2 shows these triads. All of the BA networks have as signature triad Triad 3, two directed edges both pointing to the same node, and this triad is over-represented. That Triad 3 is the overwhelming triad structure in the BA method is not surprising, as BA networks are close to directed trees. In all of the ER models on 500 nodes, Triad 15 is under-represented - two nodes connected by bi-directional edges to a third node, without an edge between the two nodes. In the ER models on 1000 nodes, the directed path triad 10 is over-represented. The GEO models do not show quite as clear a pattern: triads 9 (3 bi-directional edges) and 15 (2 bi-directional edges) are the only ones appearing as signature triads, both are over-represented, but there appears to be no density or size dependence on which of the two patterns is the signature pattern. These signature patterns for GEO are plausible as in the generated GEO networks, all edges are bidirectional, whereas in the BA networks all edges except the

initial edge are uni-directional; in the ER networks edges can be bi-directional or uni-directional.

The signature orbits for the BA networks are 6 (two edges emanating from a central node with the other two nodes linked by a directed edge), 12, and 19. These orbits do not sit on the signature triad 3 but instead pick out parts of other triads. Orbit 6, 12 and 19 pick up on the lack of triangles in BA networks which have a reciprocal edge. The signatures are unusual compared to ER and GEO networks as constructed, but may not appear unusual when compared to other networks. The complete results can be found in the Supplementary Material.

5 Case Study II: Clustering of Different Types of Networks

For this case study, we selected 31 networks of 6 different clusters from [10] - the directed social networks with less than 10 M nodes, as well as the co-purchasing networks and the internet peer-to-peer networks. The selected networks are google networks, twitter networks, amazon networks, a citation network, p2p-Gnutella networks and Slashdot networks. The edge density regions for the google networks and the twitter networks overlap, and the edge density of the citation network is in the range of that of the p2p-Gnutella networks. Detailed information of the networks can be found in the Supplementary Material. The networks vary considerably in the number of nodes and density; here the research question is whether *TriadEuclid* and *TriadEMD* detect information beyond the number of nodes and the density. Note that we assume as ground truth that networks from the same source belong to the same cluster. For comparison, *DGCD13* achieves an ARI of 0.38, and *DGCD129* achieves an ARI of 0.45.

Results for *TriadEuclid*. We map the networks to the feature space using the map T in (2) to obtain the dissimilarity matrix, and assign the networks to 6 clusters using hierarchical clustering. The ARI between the true partition and the clustering result is 0.70. To assess whether such a large ARI could be obtained just based on size and densities of the networks, we carry out Monte Carlo tests. First we generate 49 groups of 31 random networks by the ER model, which clone the sizes and densities of the original group. The p -value of the Monte Carlo test is 0.02, hence the null hypothesis is rejected under a significance level of 5%, indicating that *TriadEuclid* captures information beyond the number of nodes and the densities of the networks. Indeed this Monte Carlo test rejects any ARI value which is larger than 0.2 at the 5% level.

In the second test we generate 49 groups of 31 random networks which clone the degree distributions of the original group and are generated by the configuration model. Again, the p -value is 0.02, hence the null hypothesis is still rejected under a significance level of 5%. In contrast, an ARI of 0.45 as obtained for *DGCD129* has a p -value of 0.22, so that the hypothesis that *DGCD129* picks up no more information from the networks than is provided by the degree distribution cannot be rejected.

We obtain the signature triads by comparing the pairwise Euclidean distance between the feature vectors; the details are found in the Supplementary Material. Amazon and Cit-Hep have signature triad 3, a hierarchical structure with two reciprocal edges. The google networks tend to have signature triad 4, a 2-star with both edges pointing outward. However, Slashdot – a community of technology-related users, has triad 15 as signature triad.

Results for *TriadEMD*. First a dissimilarity matrix is obtained using *TriadEMD*. Then, as for *TriadEuclid*, we apply hierarchical clustering to cluster the networks. The ARI is 0.34, which is lower not only than that of *TriadEuclid*, but also of the *DGCD* methods. For this case study with a clear separation between the network types, *TriadEuclid* outperforms the *DGCD* methods, which in turn outperform *TriadEMD*.

6 Case Study III: Analysis of World Trade Networks

In this case study we compare networks in an example with fairly high density, namely 39 world trade networks, one per year between the years 1962 to 2000, from [6]; details are given in the Supplementary Material. There are 507 commodities which are common across the years, and we include only the trade of these commodities. Our aim is to recover the time ordering according to the dissimilarity between years.

We employ two ways to compute the dissimilarity between two years, a single-layer analysis and a multi-layer analysis. In the single-layer analysis we combine the networks of different commodities into one network and compute the dissimilarity based on the whole network of all the commodities; in the multi-layer analysis each commodity trade network is considered separately.

First we carry out the single-layer analysis. Note that the ARI does not apply as there are no pre-defined clusters which could be recovered. Instead the focus is on recovering the time ordering of the networks. For comparison, the NNscore for *DGCD13* is 0.61, and the NNscore for *DGCD129* is 0.64, whereas the N2Nscore for *DGCD13* is 0.20, and the N2Nscore for *DGCD129* is 0.25.

Results for *TriadEuclid*. The NNscore and the N2Nscore of *TriadEuclid* are 0.58 and 0.23. The N2Nscore is considerably lower than the NNscore, but higher than that of *DGCD13*. We also generate 49 random groups of 39 networks according to the ER model and 49 random groups of 39 networks according to the configuration model for the Monte Carlo Test. The null hypothesis is that our method does not give higher scores than on random networks. All p -values of the tests are 0.02 and hence we reject the null hypothesis at 5% level for both models. We conclude that this data set, *TriadEuclid* captures more information than only network size and degree distributions.

The network comparison result for *TriadEuclid* is visualised using a heatmap; different dissimilarity values are given different colours. For a good temporal ordering we would see diagonal lines. The heatmap can also help identify sudden changes in the network. Figure 3 is the heatmap of the dissimilarity matrix. Years that are close to each other are more similar than years that are

far away, which is what we expected to see. There is a sudden change from 1983 to 1984. Indeed, the way the dataset was built changed in 1983, and that is a plausible explanation for the sudden change. Also, some small changes observed from the heatmap can be related to some historical events. For instance, the Soviet Union dissolved in 1991, and the World Trade Organization (WTO) was established in 1995.

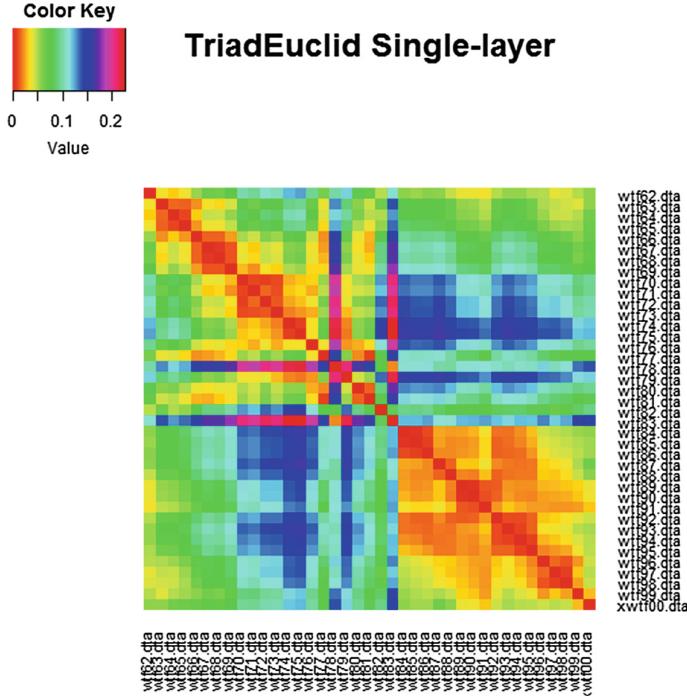


Fig. 3. Heatmap of the Single-layer World Trade Network for *TriadEuclid*

Results for *TriadEMD*. The NNscore and the N2Nscore of *TriadEMD* are 0.76 and 0.30, thus outperforming *TriadEuclid* as well as the *DGCD* methods. We also carry out the Monte Carlo test here, using the same random networks as generated before. The null hypothesis is that our method does not give higher scores than on random networks. The p -values for all tests are 0.02 and hence again we reject the null hypothesis at 5% level for both models. Again we conclude that for this data set, *TriadEMD* captures more information than only network size and degree distributions.

Similar to the heatmap obtained by *TriadEuclid*, the heatmap in Fig. 4 reflects the time evolution of the World Trade Network. The sudden change from 1983 to 1984 and the changes in 1991 and 1995 are even clearer here.

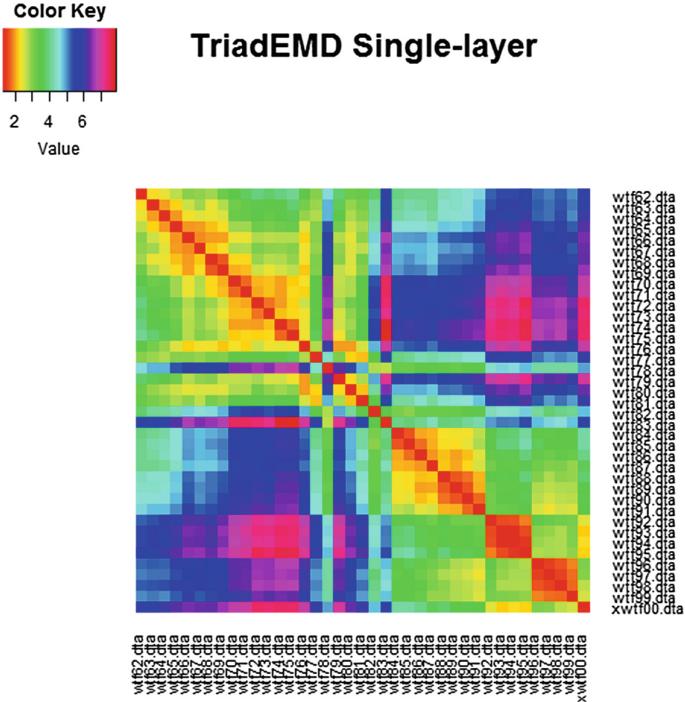


Fig. 4. Heatmap of the Single-layer World Trade Network using *TriadEMD*

Result of the multi-layer analysis. Finally we also carried out a multi-layer analysis of the world trade networks. Recall that in each year, there are 507 commodities, corresponding to 507 trade networks. We assume that the networks have different layers and each layer represents the network of one commodity. We compare the networks of each commodity to get the sub-dissimilarities for each commodity or layer. Then, we sum up the sub-dissimilarities as the dissimilarity measure between the two years. Due to computation complexities, we do not conduct the Monte Carlo test here, nor carry out the comparison to the *DGCD* scores.

The NNscore of *TriadEuclid* is 0.94, and its N2Nscore is 0.53. Both scores are considerably higher than those in the single-layer analysis. Due to computation complexities, we do not conduct the Monte Carlo test here. The heatmap for the multi-layer analysis is in the Supplementary Material. The time ordering are clearer in the multi-layer analysis than in the single-layer analysis. The sudden change from 1983 to 1984 is even more obvious.

The NNscore of *TriadEMD* is 0.97 and the N2Nscore of *TriadEMD* is 0.82. As is also the case for *TriadEuclid*, both scores are considerably higher than those in the single-layer analysis,. They are also higher than the *TriadEuclid* scores in the multi-layer analysis. The heatmap in the Supplementary Material shows clear diagonal lines.

7 Conclusion

Network comparison has many valuable applications. In this paper, we focus on comparison of directed networks. We develop two triad-based methods, *TriadEuclid* and *TriadEMD*. *TriadEuclid* uses the Euclidean distance between feature vectors of triad frequencies to measure dissimilarity. *TriadEMD* further divides the triads into 30 orbits and uses *NetEMD* to measure the dissimilarity.

We use ARI, NNscore and N2Nscore to measure the performance of our methods, and we compare these values to those obtained by the *DGCD* methods. All four methods give perfect clustering on the synthetic data set (Case Study I); *TriadEuclid* performs better than the *DGCD* scores and *TriadEMD* in Case Study II, while *TriadEMD* performs better than the *DGCD* scores and *TriadEuclid* in Case Study III. We also apply a Monte Carlo test to test whether our methods can ignore the influence of size and density and find evidence that the best-performing methods use information which goes beyond degree distributions. Moreover, we are able to relate changes in the heatmap of the dissimilarity matrix of the World Trade networks to some historical events. For the World Trade networks, a multi-layer analysis resulted in higher classification scores than the single-layer analysis. It would be interesting to carry out a fine-scale analysis to identify and interpret the layers which contribute most to the classification.

In addition, we find the signature triad and signature orbit of each type of network by looking at the largest component in the dissimilarity matrix. These signatures can be viewed as a non-parametric analog of network motifs; instead of depending on an underlying model, they depend on the other networks in the comparison. In future work, an in-depth analysis of such signatures for different network comparisons may reveal more insights into the building blocks of complex networks.

Acknowledgements. This work was partially supported by the Alan Turing Institute. GR acknowledges the COST Action CA15109. The authors would like to thank Luis Ospina-Forero and Martin O'Reilly for helpful discussions, and Andrew Elliott for helpful discussions as well as computing support. They would also like to thank the anonymous referees for many helpful comments.

References

- Ali, W., Rito, T., Reinert, G., Sun, F., Deane, C.M.: Alignment-free protein interaction network comparison. *Bioinformatics* **30**(17), i430–i437 (2014)
- Aparicio, D., Ribeiro, P., Silva, F.: Extending the applicability of graphlets to directed networks. *IEEE/ACM Trans. Comput. Biol. Bioinform. (TCBB)* **14**(6), 1302–1315 (2017)
- Borgwardt, K.M., Kriegel, H.P., Vishwanathan, S., Schraudolph, N.N.: Graph kernels for disease outcome prediction from protein-protein interaction networks. *Pacific Symp. Biocomput.* **12**, 4–15 (2007)
- Dwass, M.: Modified randomization tests for nonparametric hypotheses. *Ann. Math. Stat.* 181–187 (1957)

5. Faust, K.: A puzzle concerning triads in social networks: graph constraints and the triad census. *Soc. Netw.* **32**(3), 221–233 (2010)
6. Feenstra, R.C., Lipsey, R.E., Deng, H., Ma, A.C., Mo, H.: World trade flows: 1962–2000. Technical report, National Bureau of Economic Research (2005)
7. Holland, P.W., Leinhardt, S.: Local structure in social networks. *Sociol. Methodol.* **7**, 1–45 (1976)
8. Hubert, L., Arabie, P.: Comparing partitions. *J. Classif.* **2**(1), 193–218 (1985)
9. Kuchaiev, O., Pržulj, N.: Integrative network alignment reveals large regions of global network similarity in yeast and human. *Bioinformatics* **27**(10), 1390–1396 (2011)
10. Leskovec, J., Krevl, A.: SNAP Datasets: Stanford large network dataset collection. <http://snap.stanford.edu/data> (2014). Accessed 21 May 2017
11. Mamano, N., Hayes, W.B.: Sana: Simulated annealing far outperforms many other search algorithms for biological network alignment. *Bioinformatic* (2017)
12. Milo, R., Itzkovitz, S., Kashtan, N., Levitt, R., Shen-Orr, S., Ayzenshtat, I., Sheffer, M., Alon, U.: Superfamilies of evolved and designed networks. *Science* **303**(5663), 1538–1542 (2004)
13. Milo, R., Shen-Orr, S., Itzkovitz, S., Kashtan, N., Chklovskii, D., Alon, U.: Network motifs: simple building blocks of complex networks. *Science* **298**(5594), 824–827 (2002)
14. Newman, M.: Networks: An Introduction. Oxford University Press (2010). <https://books.google.co.uk/books?id=7LmNAQAAQAAJ>
15. Neyshabur, B., Khadem, A., Hashemifar, S., Arab, S.S.: Netal: a new graph-based method for global alignment of protein-protein interaction networks. *Bioinformatics* **29**(13), 1654–1662 (2013)
16. Picard, F., Daudin, J.J., Koskas, M., Schbath, S., Robin, S.: Assessing the exceptionality of network motifs. *J. Comput. Biol.* **15**(1), 1–20 (2008)
17. Pržulj, N.: Biological network comparison using graphlet degree distribution. *Bioinformatics* **23**(2), e177–e183 (2007)
18. Rito, T., Wang, Z., Deane, C.M., Reinert, G.: How threshold behaviour affects the use of subgraphs for network comparison. *Bioinformatics* **26**(18), i611–i617 (2010)
19. Rubner, Y., Tomasi, C., Guibas, L.J.: A metric for distributions with applications to image databases. In: Sixth International Conference on Computer Vision, pp. 59–66. IEEE (1998)
20. Sarajlić, A., Malod-Dognin, N., Yaveroğlu, Ö.N., Pržulj, N.: Graphlet-based characterization of directed networks. *Sci. Rep.* **6**, 35,098 (2016)
21. Villani, C.: Optimal Transport: Old and New, vol. 338. Springer Science & Business Media (2008)
22. Wale, N., Watson, I.A., Karypis, G.: Comparison of descriptor spaces for chemical compound retrieval and classification. *Knowl. Inf. Syst.* **14**(3), 347–375 (2008)
23. Wasserman, S., Faust, K.: Social Network Analysis: Methods and Applications, vol. 8. Cambridge University Press (1994)
24. Wegner, A.E., Ospina-Forero, L., Gaunt, R.E., Deane, C.M., Reinert, G.: Identifying networks with common organizational principles. *J. Complex Netw.* (2017)
25. Wilson, R.C., Zhu, P.: A study of graph spectra for comparing graphs and trees. *Pattern Recognit.* **41**(9), 2833–2841 (2008)
26. Yaveroğlu, Ö.N., et al.: Revealing the hidden language of complex networks. *Sci. Rep.* **4**, 4547 (2014)



Mining Patterns with Durations from E-Commerce Dataset

Mohamad Kanaan¹(✉) and Hamamache Kheddouci²

¹ Sistema Strategy, Lyon, France

mohamad.kanaan@sistema-strategy.com

<http://www.sistema-strategy.com/Home>

² Université Claude Bernard Lyon 1, Laboratoire LIRIS, Lyon, France

hamamache.kheddouci@univ-lyon1.fr

<http://perso.univ-lyon1.fr/hamamache.kheddouci>

Abstract. Given a dataset of clickstream extracted from e-commerce logs, can we find a clear usage of the website? Are there hidden relationships between the purchased products? Are there any discriminatory behaviors leading to the purchase? To answer these questions, we propose in this paper a new **Sequential Event Pattern Mining** algorithm (SEPM). The endeavor is to mine clickstream data in order to extract and analyze useful sequential patterns of clicks. Also, in order to make these patterns clearer, the time spent on each page is taken into account. SEPM maintains the items durations during the mining process and extracts patterns with the average durations of these items without multiple scans of the dataset. Our experimental results on both real and synthetic datasets indicate that SEPM is efficient and scalable.

Keywords: Data mining · Frequent pattern · Customer behavior · E-commerce

1 Introduction

Over the last years, electronic (e)-commerce has revolutionized the retail by changing the way in which the clients purchase. User's choices are no longer limited by the product availability in the stores of his region. Today, he can search and buy products from any international store, anywhere and anytime, by using e-commerce websites. However, e-merchants have some difficulties in understanding their customers' behaviour. They want to know how customers purchase products, how they navigate through the products catalogs or even why they abandon their purchase process.

Such behaviors are very complex since they are influenced by (1) supplying factors of the e-commerce website such as prices, product availability, delivery time, rating, users' opinion, etc., (2) and external factors, such as international events (e.g. Black Friday), holidays, customer's budget, etc. These

behaviors help e-merchants to understand and discover customers' needs. Therefore, e-merchants can recommend relevant products to their customers, predict future purchases, and also ensure products' availability. Their key tasks are more focused on identifying and understanding the abandonment and purchasing processes, starting from the first stimulus of customer until his decision.

To achieve this goal, users' actions could be traced. These actions are known as clickstream. To analyze them and extract useful information, several mining tools have been developed. They aim to analyze the user's journey and discover their hidden behaviors. Some well known data mining techniques are: pattern mining, trend discovery, customers clustering and classification, collaborative filtering in recommending systems, and purchase prediction.

In this paper, we are interested in pattern mining. This mining tools is used to discover the hidden relationships between the products (the product network), and to extract discriminatory behaviors. Therefore, two aspects were taken into consideration: the order of browsed products and the time spent by customer on the page of this product. These aspects can help e-merchant to find more discriminatory behaviors and mine more precisely the pattern's skeleton. In data mining, this task is known as "Sequential pattern mining".

This paper is organized as follows. Section 2 presents a state of art of the sequential pattern mining. In Sect. 3, a new algorithm is developed to mine sequential event pattern. The experimental results are shown in Sect. 4. Finally, the conclusion is elaborated in Sect. 5.

2 Related Work

Sequence pattern mining is a challenging task that aims to discover frequent patterns from a sequential database. It was proposed in [5] as a problem of mining customer's sales transactions in order to discover frequent sequences of purchasing constrained by a user-specified minimum support. Later, several efficient algorithms were developed [7], and can be divided into two categories: candidate generation and pattern growth.

1. Candidate generation: GSP [4] and SPADE [1] are the two well-known algorithms in this category. These algorithms are extensions of Apriori algorithm, and they are based on two main steps: candidate generation and support counting.
 - (a) GSP algorithm: it extracts the patterns level by level. At each level k , all possible frequent patterns with k items (called k -patterns) are mined. To start, GSP scans the database at the level $k = 1$ to identify all single frequent items. Then, it generates for next level $k + 1$ all potentially patterns ($k + 1$)-candidates. The ($k + 1$)-candidates are generated by joining k -patterns with itself. For example, considering $P_1, P_2 \in k\text{-patterns}$ such

as P_1 and P_2 have the same items except the last one: a new $(k + 1)$ -candidate is generated by joining P_1 with the last item of P_2 , and another $(k + 1)$ -candidate is generated by joining P_2 with the last item of P_1 . When all $(k + 1)$ -candidates are generated, GSP re-scans the database to count their supports and keeps only the most frequent among them. This process is repeated until no more patterns are found.

- (b) SPADE algorithm: it converts the original sequences database (in horizontal format) to a vertical database. In this vertical database, each table is called **IDList** and contains a frequent item with all its positions in the original database. The vertical database is built by scanning the original database twice. The main steps of SPADE are similar to those of GSP but without multi-scan. In fact, IDList is used also to represent a pattern. For example, the IDList of a pattern P contains all items' positions in the original database where P occurs. Thereby, the support of a pattern can be defined as the number of distinct id of the sequence list in its IDList (described in details in Sect. 3). Also, by using IDList, the joining operation between two patterns can be done without re-scanning the database. It becomes a simple join of their IDList.
- 2. Pattern growth: most of algorithms in this category ([2], [3] and [6]) are inspired by depth-first search algorithm. One of well-known algorithms is PrefixSpan [3]. It avoids the recursive exploring of its original database (which is very expensive) by projecting it to a set of smaller databases. Then, in each local database, patterns are grown by appending items to them to create larger patterns. To avoid creating the same pattern twice during the pattern growing, the items are appended according to a total order \prec which can be a lexicographical order or any other total order on items.

3 Sequential Event Pattern Mining Algorithm

3.1 Definitions

Definition 1. *Event: An event is denoted by $E = (\text{label}, \text{duration})$ where $E.\text{label}$ and $E.\text{duration}$ denotes the event label and duration (>0) respectively.*

Definition 2. *List of sequential events: A list of sequential events $SE = \{E_1, E_2, \dots, E_n\}$ is a sequence of events sorted by their starting times in ascending order (we assume that events will be appended to the SE by order).*

SE is identified in the original database by a unique identifier called **SID**, and each of its events E has also a unique identifier called **EID**. The EID indicates the position of E in SE . The length of SE is given by the number of its events $l = |SE|$.

A canonical representation of SE can be obtained by merging all the labels of its events in their orders (e.g., in Table 1, SE with $SID = 3$ can be represented by $< ACBD >$).

At this point, it is worth noting that for two given events E_i and E_j , we denote $E_i \prec E_j$, if $1 \leq \{i, j\} \leq l$, $\{E_i, E_j\} \in SE$ and E_i occurs before E_j in SE . As events are appended by order of their start time to SE , the total order \prec on events can be reduced to a simple comparison test of their EID $\{i, j\}$ in SE .

Table 1. Example database of event lists and patterns detected

SID	Events	Event list (Label _{eid})	Patterns detected
1	$E_1(A, 7), E_2(B, 15)$ $E_3(D, 6), E_4(C, 3)$	$EL_1 = A_1 \rightarrow B_2 \rightarrow D_3 \rightarrow C_4$	A (8.5) , B (7.67) C (5.25) , D (6.25)
2	$E_1(D, 2), E_2(C, 6)$	$EL_2 = D_1 \rightarrow C_2$	A (8.5) B (10)
3	$E_1(A, 10), E_2(C, 8)$ $E_3(B, 5), E_4(D, 8)$	$EL_3 = A_1 \rightarrow C_2 \rightarrow B_3 \rightarrow D_4$	A (8.5) C (5.5) A (8.5) D (7)
4	$E_1(B, 3), E_2(D, 9)$ $E_3(C, 4)$	$EL_4 = B_1 \rightarrow D_2 \rightarrow C_3$	B (9) C (3.5) B (7.67) D (7.67) D (5.67) C (4.33) A (8.5) B (10) D(7) B (9) D (7.5) C(3.5)

The input database (list of SE) is represented in vertical format [1]. Each distinct event in the database becomes an **IDList** containing the label and the positions list $< SID, EID >$ of the event.

Definition 3. Subsequence: given two sequence list $SE = \{E_1, E_2, \dots, E_k\}$ and $SE' = \{E'_1, E'_2, \dots, E'_k\}$, $SE \subseteq SE'$, if and only if there exists an injective function $f: SE.events \rightarrow SE'.events$ such that for any $\{E_i, E_j\} \in SE$, if $E_i \prec E_j \rightarrow f(E_i) \prec f(E_j)$ and $\{f(E_i), f(E_j)\} \in SE'$. For example, $EL_4 \subset EL_1$ and $EL_2 \not\subseteq EL_3$ can be deduced from Table 1.

Definition 4. Pattern: a pattern is a frequent k -subsequence where k is the number of its items.

Patterns are also represented in vertical format. Each pattern is associated with an **IDList** containing a list of labels of its items, and a list of $< SID, EID >$ where its items occur in the database. This vertical representation enables a very fast compute of support of the pattern $PSupp$, by simply counting the number of distinct SID in its **IDList**.

A valid pattern is constrained as follows: (1) *frequency*: $PSupp \geq minSupp$, where $minSupp$ is a user-specified minimum support threshold, (2) *event order*: $\forall \{Item_i, Item_j\} \in pattern$, if $1 \leq i < j \leq k$ ($Item_i$ is discovered before $Item_j$) and $Item_i.SID = Item_j.SID \rightarrow Item_i.EID < Item_j.EID$.

3.2 Problem Reformulation

Let D be an input database of SE and $minSupp$ a user-specified minimum support threshold. The goal is to find all possible and valid patterns including the average durations of their items. To achieve this goal, we choose to maintain the durations of the items in patterns during the mining process to accelerate the computing of the average durations at the end. For this reason, we choose to augment the representation of $IDList$ by introducing $IDListExt$.

Definition 5. *IDListExt*: is an extension of $IDList$. It contains in addition, a list of neighbors $INeighbors$ and a list of durations $IDurations$.

Definition 6. *INeighbors*: is the list of items that follows the last item in pattern (or the item of event) in at least $minSupp$ sequence lists. For example, in Table 1, for $minSupp = 2$, the neighbors of A are $\{< B, freq = 2 >, < C, freq = 2 >, < D, freq = 2 >\}$ and those of D are $\{< C, freq = 3 >\}$. The list of neighbors can help pattern growing and avoid false candidates (cf. Sect. 3.4).

Definition 7. *IDurations*: is the list of durations of the item in each $< SID, EID >$ where it occurs. *IDurations* of a pattern is the list of durations of its last item, collected from the original $lastItem.IDList$ where the pattern occurs. An exemple of $IDListExt$ is shown in Table 2 for items $\{A, B, C, D\}$ and pattern $\{AB\}$ extracted from database in Table 1.

Table 2. Example of $IDListExt$ built from Table 1 with $minSupp = 2$

A			B			C		
SID	EID	Duration	SID	EID	Duration	SID	EID	Duration
1	1	7	1	2	15	1	4	3
3	1	10	3	3	5	2	2	6
INeighbors: { B, C, D }			INeighbors: { C, D }			INeighbors: { }		
D			AB					
SID	EID	Duration	SID	EID	Duration			
1	3	6	1	2	15			
2	1	2	3	3	5			
3	4	8	INeighbors: { C, D }					
4	2	9						
INeighbors: { C }								

3.3 Algorithms

We describe here our proposed algorithms to build IDListExt and detect patterns with SEPM.

Build IDListExt: Algorithm 1 is proposed to build the IDListExt of each frequent event in the database. It starts by scanning the database to obtain all single frequent events f_1 (infrequent events can be removed from database to accelerate the second scan). A second scan is performed to build the IDListExt of each frequent event (line 5–13). The last operation (line 15) will remove all infrequent neighbors from all INeighbor.

Algorithm 1 Build all IDListExt

Input: Event List db

```

1:  $f_1 \leftarrow$  all single frequent events
2: for all ( $el \in db$ ) do
3:    $antecedents \leftarrow \emptyset$ 
4:    $EID \leftarrow 1$ 
5:   for all ( $e \in el.events$ ) do
6:     if  $f_1$  contains  $e$  then
7:        $IDListExt \leftarrow findOrCreateIDListExt(e.label)$ 
8:        $IDListExt.add(el.SID, EID, e.duration)$ 
9:       add  $IDListExt.item$  to all  $INeighbor$  in  $antecedents$ 
10:       $antecedents \leftarrow antecedents \cup IDListExt$ 
11:       $EID \leftarrow EID + 1$ 
12:    end if
13:   end for
14: end for
15: remove infrequent neighbors from all  $INeighbors$ 

```

Extract pattern with SEPM: Algorithm 2 is proposed to extract all patterns without scanning the original database. It is based on two main steps: generating candidates, and then filtering them. It starts by generating the 1-patterns from every item IDListExt (line 1). For each item and for all its SID, SEPM picks the minimum EID and the duration associated with it.

Then at each level k , $(k + 1)$ -candidates are generated from the found k -patterns. From each pattern in k -patterns, new candidates are generated by joining the pattern with every item in its IDListExt.INeighbor (line 5–18) called neighbor. When the support of a candidate meets the requirements (line 10), a new pattern is created from: (1) current pattern, (2) neighbor, (3) and IDListExt resulting from the join between pattern.IDListExt and neighbor.IDListExt.

The average durations of the items in a pattern P can be obtained from the list of durations in its IDListExt and those in all its predecessors. This operation

can be performed by recursively exploring the predecessors of P and obtaining duration of each position $\langle \text{SID}, \text{EID} \rangle$ where SID appears in P.IDListExt. For example, we obtain from the database in Table 1, a 1-pattern $\langle A \rangle$ that has a neighbor B. When $\langle A \rangle$ joins its neighbor B, a new pattern $\langle AB \rangle$ is created (see Table 3). In this new pattern $\langle AB \rangle$, we keep only the durations of the last item B. We can get those of item A, by referring to the pattern $\langle A \rangle$ the predecessor of $\langle AB \rangle$.

Algorithm 2 SEPM

Input: Set of IDListExt $setIDListExt$

```

1: patterns  $\leftarrow$  obtain 1-pattern from  $setIDListExt$ 
2: while patterns  $\neq \emptyset$  do
3:   Print patterns
4:   candidates  $\leftarrow \emptyset$ 
5:   for all (pattern  $\in$  patterns) do
6:     for all (neighbor  $\in$  pattern.IDListExt.INeighbors) do
7:       if neighbor  $\notin$  pattern.blackList and pattern.canJoin(neighbor) then
8:         neighborIDListExt  $\leftarrow setIDListExt.find(neighbor)$ 
9:         newIDListExt  $\leftarrow pattern.IDListExt \text{ join } neighborIDListExt$ 
10:        if newIDListExt.|SID|  $\geq minSupp$  then
11:          newPattern  $\leftarrow createPattern(pattern, neighbor, newIDListExt)$ 
12:          candidates  $\leftarrow candidates \cup newPattern$ 
13:        else
14:          pattern.blacklist  $\leftarrow pattern.blacklist \cup neighbor$ 
15:        end if
16:      end if
17:    end for
18:  end for
19:  patterns  $\leftarrow$  candidates
20: end while

```

Table 3. Example of 2-pattern $\langle AB \rangle$ and 3-pattern $\langle ABD \rangle$ with $\text{minSupp} = 2$

AB			ABD		
SID	EID	Duration	SID	EID	Duration
1	2	15	1	3	6
3	3	5	3	4	8
Output: A (8.5) B (7.67)			Output: A (8.5) B (10) D (7)		

3.4 Pruning

Two pruning mechanisms are proposed to reduce the search space and accelerate the execution time. These mechanisms eliminate false candidates and avoid unnecessary join operations.

1. The first mechanism is based on the following property:

If joining a pattern P with an item I generates an infrequent pattern $P' \rightarrow$ adding I to any superpattern of P will generate an infrequent pattern

To handle this property, a *blacklist* of items is built and if an item cannot join a pattern, it will be added to the blacklist of this pattern. Then we can avoid joining items with patterns when they appear in their *blacklists*.

2. The second mechanism is based the following property:

Let the item N be the neighbor of a pattern P . If N is not neighbor of all items in $P \rightarrow$ joining P with N will generate an infrequent pattern

Based on this property, we can safely reject candidates resulting from joining a pattern with an item that do not appear in all INeighbors of items in P .

Figure 1 shows the tree of patterns mined from the database in Table 1. Red nodes represent the candidates rejected by the pruning mechanism without any join operation.

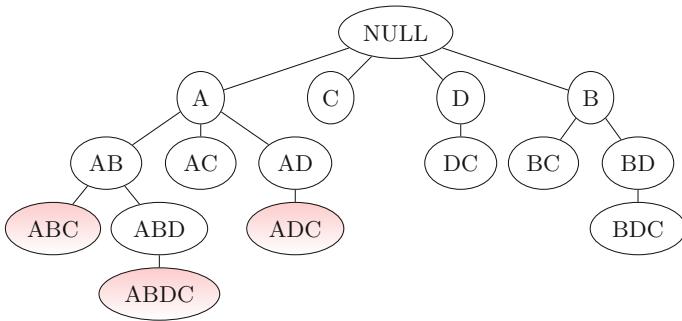


Fig. 1. Tree showing the patterns and their children. The red nodes are rejected.

4 Experimental Evaluation

SEPM was applied on several real and synthetic datasets to study its performance. The real datasets were used to demonstrate the relevance of the mined patterns, and the synthetic datasets to prove the efficiency and the scalability of the algorithm. All experiments were performed on a computer with Core i5 processor, running on Windows 10 and 16 GB of free RAM.

4.1 E-commerce Datasets

Datasets. The experiments were carried out on three real datasets presented in Table 4. These datasets only contain clicks on the products. The minimum support is set to 0.01% for experimental purposes. When it changes, it affects directly the number and length of the found patterns. Since it is hard to define a common value for all datasets, this threshold must be fixed by analysts according to their needs and their dataset dimensions.

Table 4. Three real datasets used in experiments

Dataset	Event lists	Events	Nb patterns	Max length	Reference
cikm	238k	2,121k	25,855	6	[9]
yochoose	670k	3,870k	32,474	9	[10]
alibaba	1,119k	13,966k	10,923	6	[11]

Preparing data. To be able to apply SEPM, raw clickstream data must be converted into sequential event data by following these steps: (1) every session is converted into an event-list, (2) and every click in a session is converted into an event attached to its corresponding event-list. The duration of an event is defined as the time between the start time of a click and that of the next click in the same session (we suppose that the user will not be on two pages at the same time). All the clicks durations in a session can be obtained except the duration of the last one as it is hard to know when the user left the website. This duration can be replaced by the average duration of all clicks in the current session.

Patterns categorization. Table 5 shows some 4-patterns found. As we can see, this temporal representation of patterns is very useful to analyze the product network and discover the discriminatory behaviours. We notice that the durations of items in the same pattern are not equal all the time. The gap between them is sometimes small, and sometimes very important. Based on this observation, we categorize the patterns according to two variants: the standard deviation of the durations called **sd**, and the number of variations of the durations called **variation**, such as:

$$sd_p = \sqrt{\frac{\sum_{i=1}^{l_p} (d_i - v)^2}{l_p}} \quad (1)$$

where l_p is the length of pattern p , d_i is the duration of its i^{th} item, and v is the mean of all durations in p ,

$$variation_p = \sum_{i=1}^{l_p-1} (v_i) \quad (2)$$

where v_i is a Boolean indicating whether there is a significant gap between the durations of $(i)^{\text{th}}$, and $(i+1)^{\text{th}}$ items, such as $v_i = 1$ if $|d_{i+1} - d_i| \geq \sigma$, and 0 otherwise (σ is a user-specified threshold, set to 1 min in our tests). To distinguish the direction of variation, we denote v_i^+ the positive variation if $d_{i+1} \geq d_i$ and v_i^- the negative variation in the other case.

Table 5. Example of 4-patterns found

Label of product (duration in minutes)						
45925(0.6)	→	90884(0.18)	→	36425(3.97)	→	10450(1.9)
8644(2.74)	→	768(0.77)	→	8644(10.92)	→	48580(3.92)
36343(5.18)	→	36343(8.78)	→	31163(4.11)	→	31163(0.73)
54421(0.32)	→	11338(2.36)	→	33890(1.23)	→	4759(1.35)
11338(0.95)	→	4926(1.35)	→	54421(0.52)	→	11338(3.12)

Based on these variants, five important categories are identified:

Category-A: includes standard patterns, with $variation = 0$ and $sd \simeq 2.57$. The products found in these patterns have strong relationships with each other. Users spend almost the same duration when they consult these products together. These patterns can be used to recommend products to users during their navigations.

Category-B: includes patterns with $variation = 1$ and $sd \lesssim 12.57$. In this category, two subcategories of patterns are discovered depending on the direction of variation of their durations (positive or negative variation). The patterns with positive variation indicate that users, before focusing on their main products, consult some related ones (e.g. mobile phone accessories before mobile phones, printer cartridges before printers, ...). These related products, even if they are not bought in the same session, can influence directly the users' opinions. These patterns can help e-merchants to offer new products collections. In the other case, when the patterns contain a negative variation, we notice that users are moving away from their target products (most sessions containing these patterns end with quick navigation without focusing on a particular product).

Category-C: includes patterns with $variation = 2$ and $sd \lesssim 25.13$. Many patterns in this category contain noises especially when the first variation is negative and the second one is positive. In this case, the users may consult non-target products during their navigations.

Category-D: includes patterns with $variation \geq 3$ and $sd \lesssim 30.33$. These patterns reflect different types of customer's behaviors, e.g. product comparison, random navigation without purchase intention, etc.

Category-E: includes patterns with $sd \geq 31$. Most of these patterns reflect abnormal behaviours. This can be due to several reasons: bugs in tracing tool, error while page loading, etc.

Based on these categories, several mining techniques can be applied to understand more the customers' behaviors. E.g. customers can be categorized based on frequent pattern categories in their sessions, and then each customer's profile can be treated separately. Also, various customers' demographics data can be combined with the patterns categories to recommend more personalized websites.

4.2 Synthetic Datasets

In this section, we apply four algorithms (SPEM, PrefixSpan, SPAM and SPADE) on synthetic datasets. The source codes of algorithms (PrefixSpan, SPAM and SPADE) are provided by [8].

Datasets. We have examined the effect of varying some dimensions of the database. We used for these experiments the “IBM data quest generator” to generate synthetic datasets. The generation of datasets can be controlled through several parameters:

1. ncrust (number of customers in the database) to vary the number of event lists (i.e. \mathbf{D})
2. slen (average of transactions per customer) to vary the number of events per list (i.e. \mathbf{L})
3. nitems (the number of different items available) to vary the type (label) of events (i.e. \mathbf{T})

To analyze the effect of D variation, we set minSupp to 10% and we vary D between 250k and 400k. In the case of L , minSupp is set to 10% and L is varied between 8 and 15. Lastly, for that of T , minSupp is set to 5% and T is varied between 3 and 10. Figure 2 shows the results. In all cases, SEPM behaves in the same way as the other algorithms when these parameters change. Results show us that SEPM behaves correctly according to the different types of database.

Pruning. Finally, we investigate the effectiveness of the proposed pruning mechanisms. For this test, two variations of SEPM are implemented with and without pruning. Figure 3 shows the results when they are applied on a synthetic dataset of 300k event list. As we can see, the pruning mechanisms are very efficient and can decrease the execution time by avoiding the generation of false candidates.

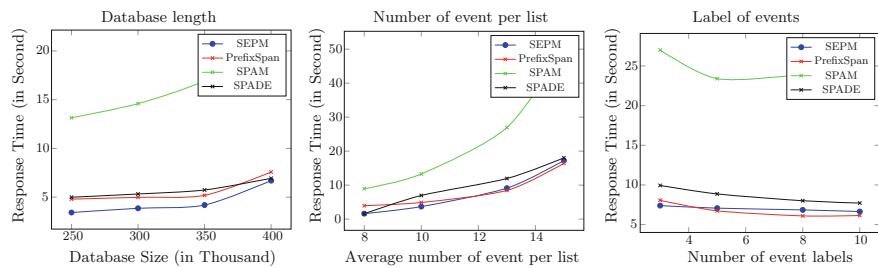


Fig. 2. Effect of varying the dimensions of the database

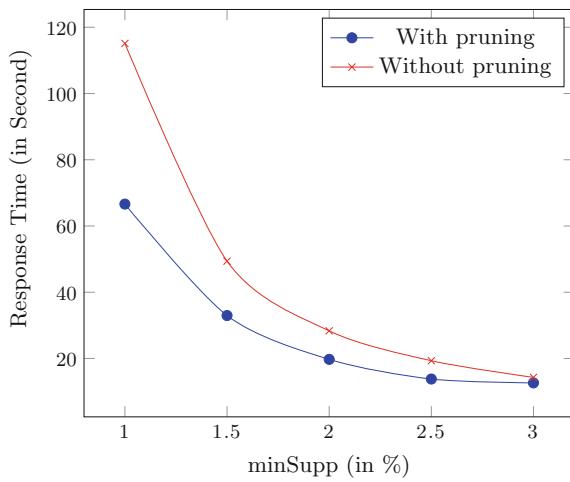


Fig. 3. Effect of pruning mechanisms

5 Conclusion

The duration that takes the customer to check products in an e-commerce website is crucial to understand his preference. Motivated by this reason, we have examined in this paper the problem of mining sequential pattern from datasets of clickstream. We augmented the existing vertical database representation with additional duration information to detect patterns with the average durations of its items. Then based on this new representation, we have proposed a new algorithm called SEPM to detect sequential patterns including the average durations of their items without multiple scans of the database. These patterns are very useful for building the product network and discovering the strong and hidden relationships between products. They are also categorized to simplify their interpretations and detect the discriminatory behaviors. Experimental results on real and synthetic datasets show the efficiency and the scalability of our proposed algorithm.

References

1. Zaki, M.J.: SPADE: an efficient algorithm for mining frequent sequences. *Mach. Learn.* **42**(1), 31–60 (2001)
2. Han, J., Pei, J., Mortazavi-Asl, B., Chen, Q., Dayal, U., Hsu, M.C.: FreeSpan: frequent pattern-projected sequential pattern mining. In: ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 355–359 (2000)
3. Srikant, R., Agrawal, R.: Mining sequential patterns: generalizations and performance improvements. In: Apers, P.M.G., Bouzeghoub, M., Gardarin, G. (eds.) EDBT 1996. LNCS, vol. 1057, pp. 3–17. Springer, Heidelberg (1996)
4. Srikant, R., Agrawal, R.: Mining sequential patterns: generalizations and performance improvements. In: The International Conference on Extending Database Technology, pp. 1–17 (1996)
5. Agrawal, R., Srikant, R.: Mining sequential patterns. In: Proceedings of the 11th International Conference on Data Engineering, Taipei, Taiwan, March 1995
6. Han, J., Pei, J., Ying, Y., Mao, R.: Mining frequent patterns without candidate generation: a frequent-pattern tree approach. *Data Min. Knowl. Discov.* **8**(1), 53–87 (2004)
7. Fournier-Viger, P., Lin, J.C.W., Kiran, R.U., Koh, Y.S., Thomas, R.: A survey of sequential pattern mining. *Data Sci. Pattern Recognit.* **1**(1), 54–77 (2017)
8. <http://www.philippe-fournier-viger.com/spmf/>
9. <https://competitions.codalab.org/competitions/11161>
10. <http://recsys.yoochoose.net/challenge.html>
11. <https://tianchi.aliyun.com/datalab/dataSet.html?dataId=649>

Network Models



Relating Emerging Network Behaviour to Network Structure

Jan Treur^(✉)

Behavioural Informatics Group, Vrije Universiteit Amsterdam, Amsterdam,
The Netherlands
j.treur@vu.nl

Abstract. Emerging behaviour of a network is usually considered to be a consequence of the network's structure, although it may be challenging to find out how this relation between structure and behaviour exactly is. In this paper a number of results are presented on how certain properties of network structure determine network behaviour, in particular what limit behaviour emerges. These network structure properties include in how far states of the network are reachable from a given state, and properties of functions to aggregate multiple impacts on a state.

1 Introduction

Networks are usually defined by their structure, and their emerging behaviour is often considered an interesting and sometimes fascinating consequence of this network structure. Although in general the emerging behaviour is assumed to be entailed by the network structure, the relation between network structure and network behaviour is not always easy to determine and can pose a real challenge. Whether or not such relations between structure and emerging behavior can actually be found is sometimes a topic for discussion. Often only simulations under varying settings of the structure parameters reveal just a glimpse of this relation. But sometimes it is possible to derive certain properties of the emerging behaviour from certain properties of the network structure in a mathematical manner. In this paper such cases are shown. This challenge is addressed using the Network-Oriented Modeling approach from [10] as a vehicle. This approach provides opportunities for mathematical analysis, and enables to derive theoretical results that predict behavior that is observed in specific cases of simulations.

Usually the network structure is described by a number of characteristics or parameters. For temporal-causal networks in particular, such characteristics are connection weights, combination functions aggregating impact of multiple states on a given state, and speed factors defining the speed of change of a state. So, the challenge is to find out how properties of connection weights, combination functions and speed factors relate to emerging behavior. In particular, in this paper it will be addressed what limit behaviour emerges; for example: Between which bounds do equilibrium values of states occur? How much variation between these bounds occurs? Under which conditions on the network structure will the values of different states converge to a common equilibrium value? Such results can be relevant, for example, to predict the

effects of social dynamics, such as spread of information or opinions, or social contagion of emotions; e.g. [1, 3].

A number of properties of a network structure have been identified here such that any network with a structure satisfying these properties show similar emergent behavior. These structure properties include in how far other states of the network are reachable from a given state, and properties of combination functions used to aggregate the impact of multiple incoming connections to a node.

In this paper, in Sect. 2 some basic concepts are briefly introduced. Section 3 shows simulation examples to get the idea of the phenomena that can be observed. In Sect. 4 properties of network structure are defined that are relevant for the considered types of emerging behaviour. Section 5 discusses a number of results that have been proven mathematically for the relation between network structure and network behaviour addressing the questions above. Finally, Sect. 6 is a discussion.

2 Conceptual and Numerical Representation of a Network

For the modeling perspective [10] used in this paper, the interpretation of connections based on causality and dynamics forms a basis of the structure and semantics of the considered *temporal-causal networks*. Nodes in a network are interpreted here as states that vary over time, and the connections are interpreted as causal relations that define how each state can affect other states over time. A conceptual representation of a temporal-causal network model includes a number of labels: connection weights, combination functions, and speed factors; see the upper part (first 5 rows) of Table 1.

The notion of combination function is similar to the functions used in a static manner in the (deterministic) Structural Causal Model perspective described, for example, in [6], but here they are used in a dynamic manner. For example, [6], p. 203, denotes nodes by V_i and his variant of combination functions by f_i ; he also points at the problem of underspecification for aggregation of multiple connections, as in the often used graph representations the role of combination functions f_i for nodes V_i , is lacking, and are therefore not a full specification of the network structure. To provide sufficient flexibility, for each state its combination function can be chosen that specifies how multiple causal impacts on this state are aggregated. For this a number of standard combination functions are available as options, but also own-defined functions can be used.

In the lower part of Table 1 the numerical representation describing the dynamics of a temporal-causal network is displayed, and it is shown how it is defined based on the network structure, thus associating dynamic semantics to any temporal-causal network specification in a numerical-mathematically defined manner. This provides a well-defined relation between network structure and network dynamics at this base level. In Fig. 1 this relation is indicated by the horizontal arrow in the lower part representing the base level. The upper part will be addressed in Sects. 4 and 5.

Table 1. Conceptual and numerical representations of a temporal-causal network model

Concept	Conceptual representation	Explanation
States and connections	$X, Y, X \rightarrow Y$	Describes the nodes and links of a network structure (e.g., in graphical or matrix format)
Connection weight	$\omega_{X,Y}$	The <i>connection weight</i> $\omega_{X,Y} \in [-1, 1]$ represents the strength of the causal impact of state X on state Y through connection $X \rightarrow Y$
Aggregating multiple impacts on a state	$\mathbf{c}_Y(\cdot)$	For each state Y (a reference to) a <i>combination function</i> $\mathbf{c}_Y(\cdot)$ is chosen to combine the causal impacts of other states on state Y
Timing of the effect of causal impact	η_Y	For each state Y a <i>speed factor</i> $\eta_Y \geq 0$ is used to represent how fast a state is changing upon causal impact
Concept	Numerical representation	Explanation
State values over time t	$Y(t)$	At each time point t each state Y in the model has a real number value in $[0, 1]$
Single causal impact	$\mathbf{impact}_{X,Y}(t)$ $= \omega_{X,Y} X(t)$	At t state X with a connection to Y has an impact on Y , using connection weight $\omega_{X,Y}$
Aggregating multiple causal impacts	$\mathbf{agimpact}_Y(t)$ $= \mathbf{c}_Y(\mathbf{impact}_{X_1,Y}(t), \dots, \mathbf{impact}_{X_k,Y}(t))$ $= \mathbf{c}_Y(\omega_{X_1,Y} X_1(t), \dots, \omega_{X_k,Y} X_k(t))$	The aggregated causal impact of multiple states X_i on Y at t , is determined using combination function $\mathbf{c}_Y(\cdot)$
Timing of the causal effect	$Y(t + \Delta t) = Y(t)$ $+ \eta_Y [\mathbf{agimpact}_Y(t) - Y(t)] \Delta t$ $= Y(t)$ $+ \eta_Y [\mathbf{c}_Y(\omega_{X_1,Y} X_1(t), \dots, \omega_{X_k,Y} X_k(t)) - Y(t)] \Delta t$	The causal impact on Y is exerted over time gradually, using speed factor η_Y ; here the X_i are all states with outgoing connections to state Y

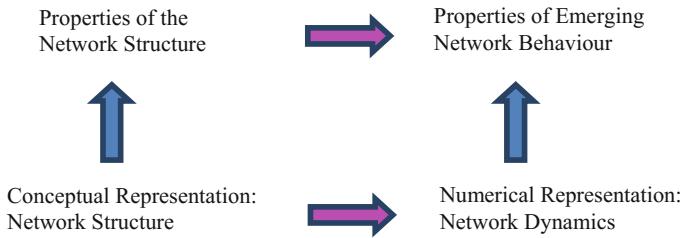


Fig. 1. Bottom layer: the conceptual representation defines the numerical representation. Top layer: properties of network structure entail properties of emerging network behaviour.

The difference equations in Table 1 can be used for simulation and mathematical analysis. Often used examples of combination functions are the *identity* function **id(.)** for states with impact from only one other state, the **max(..)** and **min(..)** function, the *scaled sum* function **ssum_λ(..)** with scaling factor λ , and the *advanced logistic sum* combination function **alogistic_{σ,τ}(..)** with steepness σ and threshold τ :

$$\begin{aligned}\mathbf{id}(V) &= V & \mathbf{ssum}_\lambda(V_1, \dots, V_k) &= (V_1, \dots, V_k)/\lambda \\ \mathbf{alogistic}_{\sigma,\tau}(V_1, \dots, V_k) &= \left[(1/(1+e^{-\sigma(V_1 + \dots + V_{k-\tau})})) - 1/(1+e^{\sigma_\tau}) \right] (1+e^{-\sigma_\tau})\end{aligned}$$

In addition to the above functions, a *Euclidean combination function* is defined as

$$\mathbf{c}(V_1, \dots, V_k) = \mathbf{eucl}_{n,\lambda}(V_1, \dots, V_k) = ((V_1^n + \dots + V_k^n)/\lambda)^{1/n}$$

where n is the *order* (which can be any positive natural number but also any positive real number), and λ is a scaling factor. Note that for $n = 1$ (first order) we get the scaled sum function $\mathbf{eucl}_{1,\lambda}(V_1, \dots, V_k) = \mathbf{ssum}_\lambda(V_1, \dots, V_k)$. For $n = 2$ it is the second-order Euclidean combination function defined by $\mathbf{eucl}_{2,\lambda}(V_1, \dots, V_k) = \sqrt((V_1^2 + \dots + V_k^2)/\lambda)$. This second-order Euclidean combination function is also often applied in aggregating the error value in optimisation and in parameter tuning using the root-mean-square deviation (RMSD).

3 Examples of Emerging Network Behaviour

To get the idea, a few examples of a Social Network addressing social contagion are briefly discussed. Consider the (fully connected) network with connection weights and speed factors shown in Table 2.

Table 2. Settings of the example Social Network: connection weights and speed factors

connections	X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8	X_9	X_{10}
X_1		0.1	0.2	0.1	0.2	0.15	0.1	0.25	0.25	0.1
X_2	0.25		0.25	0.2	0.1	0.2	0.15	0.25	0.25	0.25
X_3	0.1	0.25		0.1	0.2	0.15	0.1	0.25	0.1	0.15
X_4	0.25	0.15	0.25		0.15	0.8	0.25	0.15	0.25	0.25
X_5	0.25	0.2	0.1	0.2		0.25	0.2	0.1	0.2	0.15
X_6	0.25	0.1	0.25	0.25	0.25		0.1	0.25	0.25	0.1
X_7	0.2	0.1	0.2	0.15	0.2	0.2		0.2	0.15	0.25
X_8	0.1	0.25	0.1	0.25	0.05	0.15	0.25		0.1	0.25
X_9	0.25	0.15	0.25	0.15	0.2	0.1	0.2	0.15		0.15
X_{10}	0.2	0.25	0.2	0.2	0.1	0.2	0.15	0.8	0.2	
speed factors	0.8	0.5	0.8	0.5	0.5	0.5	0.8	0.5	0.5	0.5

For this network simulations have been performed for different combination functions. In Fig. 2 three different simulations are shown. In the upper graph advanced logistic sum combination functions are used, in the middle graph normalized scaled sum functions, and in the lower graph scaled sum functions while two states are independent and are used as inputs only, and remain constant. It turns out that in one case convergence to one common equilibrium value takes place, but in the other two cases that does not happen and instead some form of clustering seems to take place. How can we explain these differences in emerging behavior from the structure of the networks? This question will be answered by a number of theorems in Sect. 5.

4 Relevant Properties of Network Structure

In Sect. 2 it has been explained how the basic difference equations describing network behaviour relate to network structure at a base level (lower part of Fig. 1). The difference equations describe the small steps in the dynamics. But emerging behaviour usually shows itself over long time durations. Therefore, to relate emerging network behaviour to network structure, the gap between small steps and long time durations has to be bridged. This is discussed in the current section and in Sect. 5. It has been found out how the following properties of network structure (see Fig. 1, left upper

corner) underly the behavioural differences (right upper corner in Fig. 1) shown in Sect. 3 and Fig. 2. Proofs can be found in the Appendix¹.

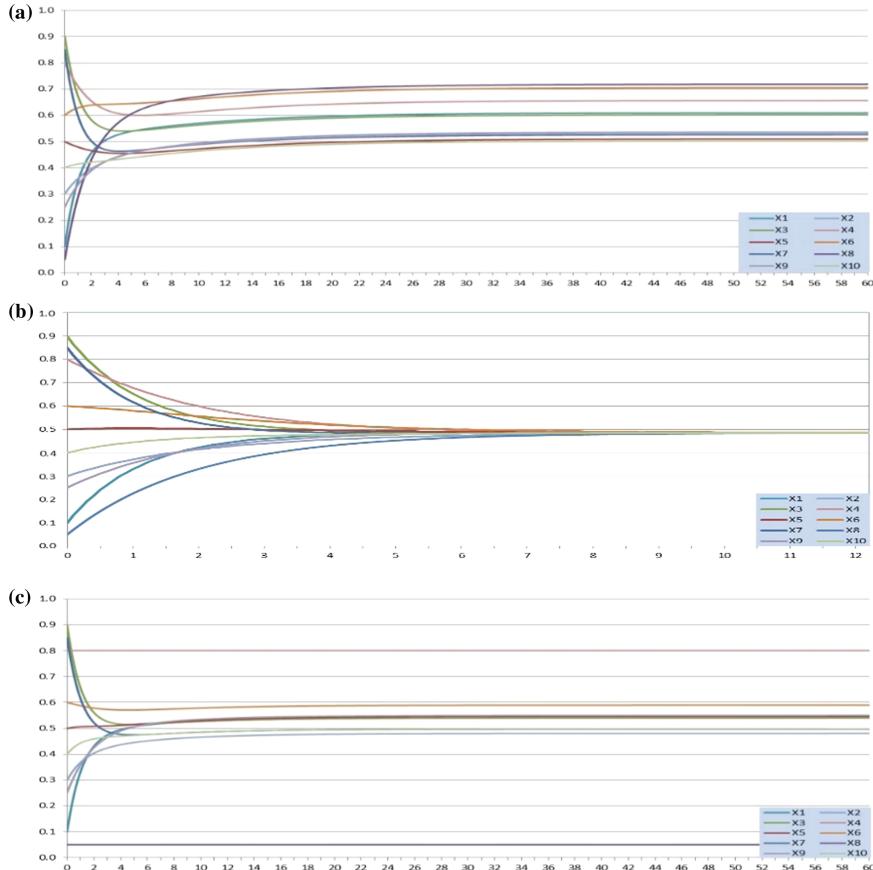


Fig. 2. The example network with (a) upper graph: advanced logistic sum combination functions with steepness $\sigma = 1.5$, threshold $\tau = 0.3$ (no common equilibrium value), (b) middle graph: normalised scaled sum functions (common equilibrium value), (c) lower graph: normalised scaled sum functions with constant X_4 (at 0.8) and X_8 (at 0.05) (no common value)

Definition 1 (Properties of combination functions).

- (a) A function $c(\cdot)$ is called *nonnegative* if $c(V_1, \dots, V_k) \geq 0$ for all V_1, \dots, V_k
- (b) A function $c(\cdot)$ respects 0 if $V_1, \dots, V_k \geq 0 \Rightarrow [c(V_1, \dots, V_k) = 0 \Leftrightarrow V_1 = \dots = V_k = 0]$
- (c) A function $c(\cdot)$ is called *monotonically increasing* if

¹ <http://www.few.vu.nl/~treur/ComplexNetworks18Structure2BehaviourProofs.pdf>.

$$U_i \leq V_i \text{ for all } i \Rightarrow c(U_1, \dots, U_k) \leq c(V_1, \dots, V_k)$$

- (d) A function $c(\cdot)$ is called *strictly monotonically increasing* if

$$U_i \leq V_i \text{ for all } i, \text{ and } U_j < V_j \text{ for at least one } j \Rightarrow c(U_1, \dots, U_k) < c(V_1, \dots, V_k)$$

- (e) A function $c(\cdot)$ is called *scalar-free* if $c(\alpha V_1, \dots, \alpha V_k) = \alpha c(V_1, \dots, V_k)$ for all $\alpha > 0$

The properties (a) and (b) are basic properties silently assumed to hold for all combination functions considered here. Sometimes combination functions are defined in such a way that (a) automatically holds: $c^*(V_1, \dots, V_k) = c(V_1, \dots, V_k)$ if $c(V_1, \dots, V_k) \geq 0$, and 0 otherwise.

Proposition 1. Linear combinations with positive coefficients of functions that are monotonic or scalar-free also are monotonic or scalar-free, respectively.

Properties (d) and (e) define a specific class of combination functions; this class includes all Euclidean combination functions, but logistic combination functions do not belong to this class, as they are not scalar-free. Also maximum-based combination functions do not belong to this class as they are monotonic but not strict. A number of results on emergent behaviour will be discussed for this class.

Proposition 2 (Proportional outcomes). If in a temporal-causal network all combination functions are scalar-free and in some scenario 1 the initial values for the states are a factor ρ times the initial values in a scenario 2, then for every t the state values in scenario 1 are ρ times the corresponding state values in scenario 2, assuming $\eta_Y \Delta t \leq 1$ for all states Y . This also holds for the equilibrium values when an equilibrium is reached.

Proposition 3 (Order preservation). If in a temporal-causal network all combination functions are monotonically increasing and in some scenario 1 the initial values for the states are \leq the initial values in a scenario 2, then for every t the state values in scenario 1 are \leq the corresponding state values in scenario 2, assuming $\eta_Y \Delta t \leq 1$ for all states Y . This also holds for the equilibrium values when an equilibrium is reached. Similarly for decreasing.

Definition 2 (normalised network). A network is *normalised* or uses normalised combination functions if for each state Y it holds $c_Y(\omega_{X_1,Y}, \dots, \omega_{X_k,Y}) = 1$, where X_1, \dots, X_k are the states with outgoing connections to Y .

For example, for a Euclidean combination function $\lambda_Y = \omega_{X_1,Y} + \dots + \omega_{X_k,Y}$ will provide a normalised network. This can be done in general:

- (1) **normalisation by adjusting the combination functions**

If any combination function $c_Y(\cdot)$ is replaced by $c'_Y(\cdot)$ defined as $c'_Y(V_1, \dots, V_k) = c_Y(V_1, \dots, V_k) / c_Y(\omega_{X_1,Y}, \dots, \omega_{X_k,Y})$, then the network is normalised: $c'_A(\omega_{B_1,A}, \dots, \omega_{B_k,A}) = 1$

(2) **normalisation by adjusting the connection weights**

For scalar-free combination functions also normalisation is possible by adapting the connection weights; define $\omega'_{X_i,Y} = \omega_{X_i,Y}/\mathbf{c}_Y(\omega_{X_1,Y}, \dots, \omega_{X_k,Y})$. Then the network becomes normalised; indeed $\mathbf{c}_Y(\omega'_{X_1,Y}, \dots, \omega'_{X_k,Y}) = 1$.

Proposition 4. Any function composed of monotonically increasing or decreasing functions including an even number of monotonically decreasing functions is monotonically increasing. The same holds for strictly monotonically increasing or decreasing.

Proposition 5. For every $n > 0$ a Euclidean combination function of n -th degree is strictly monotonic, scalar-free, symmetric and respects 0.

Another important determinant for emerging behaviour is in how far the network has paths connecting any two states:

Definition 3 (forward reachable, strongly connected and symmetric network).

- (a) State Y is *forward reachable* from state X if there is a directed path from X to Y with nonzero connection weights and speed factors.
- (b) A network is *connected* if between every two states there is a (nondirected) path with nonzero connection weights and speed factors. It is *strongly connected* if any state Y is forward reachable from any state X .
- (c) A network is *fully connected* if for any states X, Y there is a connection from X to Y
- (d) A network is called *weakly symmetric* if for all nodes X and Y it holds $\omega_{X,Y} = 0 \Leftrightarrow \omega_{Y,X} = 0$ or, equivalently: $\omega_{X,Y} > 0 \Leftrightarrow \omega_{Y,X} > 0$. The network is called *fully symmetric* if $\omega_{X,Y} = \omega_{Y,X}$ for all nodes X and Y . An adaptive network is called *continually (weakly/fully) symmetric* if at all time points it is (weakly/fully) symmetric.
- (e) A state Y is called *independent* if there is no incoming connection with connection weight $\omega_{X,Y} > 0$ or the speed factor of Y is 0.

Note that an independent state for no other state is forward reachable. The term independent refers to the fact that its behaviour over time is not affected by the other states. Its value can remain constant (when the speed factor is 0), or it can show any autonomously defined dynamics (see, for example, state X_6 in Fig. 3).

Definition 4 (symmetric combination function). A combination function is *symmetric* in a subset S of its arguments if for any U_1, \dots, U_k is obtained from V_1, \dots, V_k by a permutation of the arguments in S , it holds $\mathbf{c}(U_1, \dots, U_k) = \mathbf{c}(V_1, \dots, V_k)$. It is *fully symmetric* if S is the set of all arguments.

5 Results Relating Emerging Behaviour to Network Structure

In this section it is shown how the network structure properties discussed in Sect. 4 relate to emerging network behaviour (the horizontal arrow in the upper part of Fig. 1).

Definition 5 (stationary point and equilibrium). A state Y has a *stationary point* at t if $\mathbf{d}Y(t)/\mathbf{d}t = 0$. The network is in *equilibrium* at t if every state Y of the model has a stationary point at t .

Considering the specific differential equation format for a temporal-causal network model, and assuming a nonzero speed factor a more specific criterion can be found:

Lemma 1 (Criterion for a stationary point in a temporal-causal network). Let Y be a state and X_1, \dots, X_k the states with outgoing connections to state Y . Then Y has a stationary point at t if and only if $\mathbf{c}_Y(\omega_{X_1,Y}X_1(t), \dots, \omega_{X_k,Y}X_k(t)) = Y(t)$.

The following proposition and theorem show that always when all states have the same value, an equilibrium occurs.

Proposition 6. Suppose a network with nonnegative connections has normalised scalar-free combination functions.

(a) If X_1, \dots, X_k are the states with outgoing connections to Y , and $X_1(t) = \dots = X_k(t) = V$ for some common value V , then also $\mathbf{c}_Y(\omega_{X_1,Y}X_1(t), \dots, \omega_{X_k,Y}X_k(t)) = V$.

(b) If, moreover, the combination functions are monotonic, and X_1, \dots, X_k are the states with outgoing connections to Y , and $V_1 \leq X_1(t), \dots, X_k(t) \leq V_2$ for some values V_1 and V_2 , then also $V_1 \leq \mathbf{c}_Y(\omega_{X_1,Y}X_1(t), \dots, \omega_{X_k,Y}X_k(t)) \leq V_2$ and if $\eta_Y \Delta t \leq 1$ and $V_1 \leq Y(t) \leq V_2$ then $V_1 \leq Y(t + \Delta t) \leq V_2$.

Proof.

(a) This follows from

$$\begin{aligned}\mathbf{c}_Y(\omega_{X_1,Y}X_1(t), \dots, \omega_{X_k,Y}X_k(t)) &= \mathbf{c}_Y(\omega_{X_1,Y}V, \dots, \omega_{X_k,Y}V) \\ &= V \mathbf{c}_Y(\omega_{X_1,Y}, \dots, \omega_{X_k,Y}) = V\end{aligned}$$

(b) This follows from $V_1 = V_1 \mathbf{c}_Y(\omega_{X_1,Y}, \dots, \omega_{X_k,Y}) = \mathbf{c}_Y(\omega_{X_1,Y}V_1, \dots, \omega_{X_k,Y}V_1)$

$$\begin{aligned}&\leq \mathbf{c}_Y(\omega_{X_1,Y}X_1(t), \dots, \omega_{X_k,Y}X_k(t)) \leq \mathbf{c}_Y(\omega_{X_1,Y}V_2, \dots, \omega_{X_k,Y}V_2) \\ &= V_2 \mathbf{c}_Y(\omega_{X_1,Y}, \dots, \omega_{X_k,Y}) = V_2\end{aligned}$$

Theorem 1 (common state values provide equilibria). Suppose a network with nonnegative connections is based on normalised and scalar-free combination functions. Then the following hold.

(a) Whenever all states have the same value V , the network is in an equilibrium state.

(b) If for every state for its initial value V it holds $V_1 \leq V \leq V_2$, then for all t for every state Y it holds $V_1 \leq Y(t) \leq V_2$. In an achieved equilibrium for every state for its equilibrium value V it holds $V_1 \leq V \leq V_2$.

Proof.

(a) It follows from Proposition 6(a) that the criterion of Lemma 1 is fulfilled.

(b) From Proposition 6(b) it follows by induction over the time steps that during a simulation for every state Y it holds $V_1 \leq Y(t) \leq V_2$ and therefore in a limit

situation in an achieved equilibrium for every state for its equilibrium value V it holds $V_1 \leq V \leq V_2$.

Theorem 1 shows that common values of the states always provide equilibria. However, it does not tell whether other types of equilibria are possible as well. In subsequent theorems it is shown that in many cases no other types of equilibria emerge. As a first case, consider a network without cycles. Then the following theorem has been proven by applying induction over the acyclic graph connections starting from the independent states and thereby using Proposition 6 and Lemma 1.

Theorem 2 (equilibrium states provide common state values; acyclic case). Suppose an acyclic network with nonnegative connections is based on normalised and scalar-free combination functions.

- (a) If in an equilibrium state the independent states all have the same value V , then all states have the same value V .
- (b) If, moreover, the combination functions are monotonic, and in an equilibrium state the independent states all have values V with $V_1 \leq V \leq V_2$, then all states have values V with $V_1 \leq V \leq V_2$.

Next, a basic Lemma for dynamics of normalised networks with combination functions which are monotonically increasing and scalar-free.

Lemma 2. Let a normalised network with nonnegative connections be given with combination functions that are monotonically increasing and scalar-free; then:

- (a) (i) If for some node Y at time t for all nodes X with $\omega_{X,Y} > 0$ it holds $X(t) \leq Y(t)$, then $Y(t)$ is decreasing at $t : \frac{dY(t)}{dt} \leq 0$.
(ii) If the combination functions are strictly increasing and a node X exists with $X(t) < Y(t)$ and $\omega_{X,Y} > 0$, and the speed factor of Y is nonzero, then $Y(t)$ is strictly decreasing at $t : \frac{dY(t)}{dt} < 0$.
- (b) (i) If for some node Y at time t for all nodes X with $\omega_{X,Y} > 0$ it holds $X(t) \geq Y(t)$, then $Y(t)$ is increasing at $t : \frac{dY(t)}{dt} \geq 0$.
(ii) If, the combination function is strictly increasing and a node X exists with $X(t) > Y(t)$ and $\omega_{X,Y} > 0$, and the speed factor of Y is nonzero, then $Y(t)$ is strictly increasing at $t : \frac{dY(t)}{dt} > 0$.

Using Lemma 1 and 2 the following proposition has been proven for strongly connected networks with cycles.

Theorem 3 (common equilibrium state values; strongly connected case). Suppose the network has normalised, scalar-free and strictly monotonic combination functions, then:

- (a) If the network is strongly connected, then in an equilibrium state all states have the same value.

- (b) Suppose the network has one or more independent states and the subnetwork without these independent states is strongly connected. If in an equilibrium state all independent states have values V with $V_1 \leq V \leq V_2$, then all states have values V with $V_1 \leq V \leq V_2$. In particular, when all independent states have the same value V , then all states have this same value V .

Using Lemma 1 and 2 the following slightly more general theorem has been proven for (connected) networks with cycles.

Theorem 4 (equilibrium states provide common state values). Suppose a (possibly cyclic) network with nonnegative connections is based on normalised, strictly monotonically increasing and scalar-free combination functions, then:

- (a) If in an equilibrium state, a state Y with nonzero speed factor has highest state value or lowest state value, then all states X from which Y is forward reachable have the same equilibrium state value as Y .
- (b) Suppose except for at most one independent state, every state Y is forward reachable by all other states X . Then in an equilibrium state all states have the same state value.
- (c) Under the conditions of (b) the equilibrium state is attracting, and the common equilibrium state value is between the highest and lowest previous or initial state values.

Theorems 2, 3 and 4 can be used to prove for many cases that in an equilibrium all states have the same value. This includes cases in which the combination functions used are Euclidean combination functions. This explains why for the second simulation in Fig. 2 convergence to one common value takes place, but not for the first and third case. The first case does not satisfy the scalar-free condition, and the third case does not satisfy the condition on forward reachability; one exceptional independent state is allowed but not two, as occurs in the third example in Fig. 2. Note that the one exceptional state allowed in Theorem 4(b) can have any independently preset constant value, and all other state values converge to this value. It is even possible to give this state an autonomous pattern over time that converges to some limit value \underline{V} for $t \rightarrow \infty$; then when time passes, all state values will more or less follow this pattern and end up in the same limit value \underline{V} , all according to Theorem 4(b). Therefore:

Corollary 1. Assume the conditions of Theorem 4 hold, and one state X is independent, for which its value over time is described by the function $f(t)$, so $X(t) = f(t)$ for all t .

If $\lim_{t \rightarrow \infty} f(t) = V$, then this V is the common equilibrium value for all states.

This Corollary 1 is illustrated by Fig. 3 where X_6 has independent dynamics based on $f(t) = b_2 + b_1 e^{-a_2 t} \sin(2\pi a_1 t)$ with $a_1 = 0.03$, $a_2 = 0.035$, $b_1 = 0.5$, $b_2 = 0.6$. In this case the outgoing connections of X_6 have been increased to get a stronger effect.

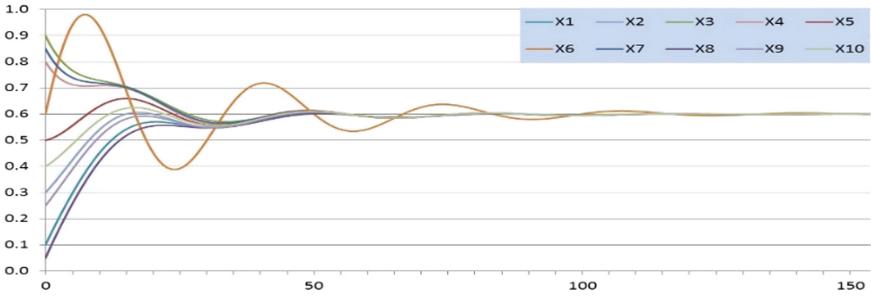


Fig. 3. How the dynamics of one independent state X_6 affects all states over time

In Fig. 4 two cases are shown in which condition (b) of Theorem 4 is not fulfilled as there are two independent states. It is shown that even if these states both have very low or very high values, still the other state values end up between these values. In the upper graph X_3 and X_7 are constant at values 0.85 and 0.9, respectively, and all other equilibrium values are between these values. In the lower graph, X_1 and X_8 are constant at values 0.1 and 0.05, respectively, and also here all other equilibrium values are between these values. Note that as in Fig. 2(c) the equilibrium values are not equal, although in this case they are close to each other. In Theorem 3 this is explained and predicted. Theorem 4(c) shows that under the conditions assumed there, the common equilibrium value lies in between the highest and lowest initial values of independent states, which is illustrated in Fig. 3. The following Theorem 5 is a further refinement of this; it shows that under some assumptions *any* value between the highest and lowest initial value can be the common equilibrium value.

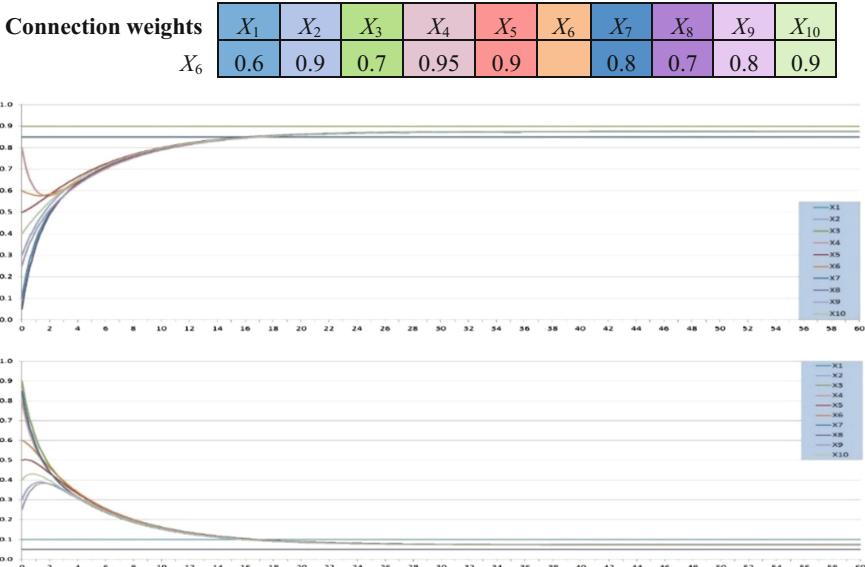


Fig. 4. How multiple independent states affect all equilibrium values

Theorem 5 (Variability of the common equilibrium value). Suppose a connected network with n states and only nonnegative connections is based on normalised, strictly monotonically increasing and scalar-free combination functions, then:

(a) A function $\underline{\text{eqf}} : [0, 1]^{2n} \rightarrow [0, 1]$ exists that assigns in an achieved equilibrium the common equilibrium value $\underline{\mathbf{V}}_{X_i} = \underline{\mathbf{V}}$ of the states to the values of the speed factors η_{X_i} , $i = 1, \dots, n$, and the initial state values V_{X_i} , $i = 1, \dots, n$ of all states:

$$\underline{\text{eqf}}(\eta_{X_1}, \dots, \eta_{X_n}, V_{X_1}, \dots, V_{X_n}) = \underline{\mathbf{V}}_{X_i} = \underline{\mathbf{V}} \text{ for all } i$$

(b) This function $\underline{\text{eqf}}(\eta_{X_1}, \dots, \eta_{X_n}, V_{X_1}, \dots, V_{X_n})$ is surjective: every value $V \in [0, 1]$ can occur as some achieved equilibrium value $\underline{\mathbf{V}} = \underline{\text{eqf}}(\eta_{X_1}, \dots, \eta_{X_n}, V_{X_1}, \dots, V_{X_n})$. More specifically, it holds:

(i) For any value $V \in [0, 1]$ and any values of speed factors $\eta_{X_1}, \dots, \eta_{X_n}$, initial values V_{X_1}, \dots, V_{X_n} exist such that

$$\underline{\mathbf{V}} = \underline{\text{eqf}}(\eta_{X_1}, \dots, \eta_{X_n}, V_{X_1}, \dots, V_{X_n}) = V$$

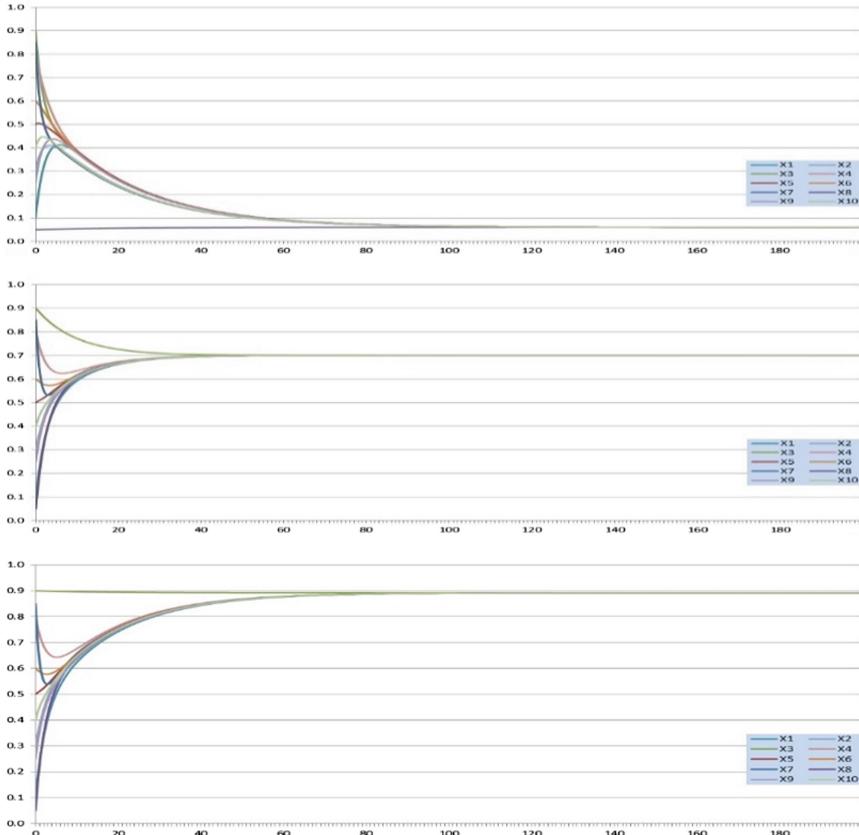
(ii) For any initial values V_{X_1}, \dots, V_{X_n} , values of speed factors $\eta_{X_1}, \dots, \eta_{X_n}$ exist such that

$$\underline{\mathbf{V}} = \underline{\text{eqf}}(\eta_{X_1}, \dots, \eta_{X_n}, V_{X_1}, \dots, V_{X_n}) = V_{X_i}$$

(iii) Moreover, if it is assumed that $\underline{\text{eqf}}(\eta_{X_1}, \dots, \eta_{X_n}, V_{X_1}, \dots, V_{X_n})$ is a continuous function of the speed factors $\eta_{X_1}, \dots, \eta_{X_n}$, then for any initial values V_{X_1}, \dots, V_{X_n} , and any value V with $\min(V_{X_1}, \dots, V_{X_n}) \leq V \leq \max(V_{X_1}, \dots, V_{X_n})$, values of speed factors $\eta_{X_1}, \dots, \eta_{X_n}$ exist such that

$$\underline{\mathbf{V}} = \underline{\text{eqf}}(\eta_{X_1}, \dots, \eta_{X_n}, V_{X_1}, \dots, V_{X_n}) = V$$

This Theorem 5 shows that both the initial values and the speed factors are important determinants of the common equilibrium value. Note that Theorem 5(b)(iii) depends on the assumption that the common equilibrium value is a continuous function of the speed factors. This will presumably depend on characteristics of the combination functions, but it is not clear yet by which types of combination functions this assumption is satisfied, in addition to being normalised, strictly monotonically increasing and scalar-free combination functions. Is it sufficient that they are continuous? Or should they be differentiable, with continuous partial derivatives which are bounded, or smooth of a certain order, maybe even of infinite order? There is still an open question here. However, as illustrated in Fig. 5, Theorem 5(b)(iii) has been confirmed in simulation experiments.



State	X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8	X_9	X_{10}
Speed factor	0.4	0.5	0.045605	0.3	0.5	0.5	0.8	0.454395	0.5	0.5

Fig. 5. How any value can become a common equilibrium value by using appropriate speed factors (here normalised scaled sum functions were used)

Here for any choice for V indeed by adapting values of speed factors in a smooth manner always values could be determined such that V became the common equilibrium value \underline{V} , thereby clearly showing that the function $\text{eqf}(\cdot)$ was continuous.

An example of that where two speed factors (of X_3 , resp. X_8) are adapted is shown in Fig. 5; here normalised scaled sum functions are used. The upper graph shows what happens if the speed factor of X_3 is 0.001, and the middle graph when the speed factor of X_8 is 0.001. In both cases it can be seen that the common equilibrium value is quite close to the initial value of X_3 resp. X_8 ; actually they differ 0.0078, resp. 0.0104 from these initial values. Next, an arbitrary value $V = 0.7$ in between was chosen, and it was found that for speed factor 0.045605 for X_3 and 0.454395 for X_8 (see also Fig. 5) the common equilibrium value was 0.70000193. This illustrates Theorem 5(b)(iii).

The theorems above all only apply to combination functions that are scalar free, such as Euclidean combination functions. An often used combination function which is not scalar free is the advanced logistic sum combination function, so we don't have any results for that type of function yet. But also some result has been obtained without the scalar free assumption. That is summarized in Theorem 6. Here it is only assumed that the combination functions are symmetric and (not necessarily strictly) monotonically increasing, and the logistic one satisfies that condition. But in this case there is an additional condition on the connection weights: they all should be equal, and the network should be fully connected. This result still applies to Euclidean combination functions, but, also to logistic sum combination functions, which indeed are symmetric and monotonic combination functions.

Theorem 6. Suppose in a network with nonnegative connections the combination functions are symmetric and monotonically increasing and the network is fully connected with equal weights: $\omega_{X,Y} = \omega$ for all X and Y . Then in an equilibrium state all states have the same value.

6 Discussion

Emerging behaviour of a network is often considered to be a consequence of the network structure, although it may be challenging to find out how this relation between structure and behaviour exactly is. In this paper a number of theorems and proofs were presented on how certain identified properties of network structure determine network behaviour, in particular what limit behaviour emerges concerning equilibrium values. These network structure properties include in how far other states of the network are reachable from a given state, and properties of combination functions used to aggregate multiple impacts on a single state. This challenge was addressed from the perspective of a Network-Oriented Modeling approach based on temporal-causal network models [10], and the analysis they allow; e.g., [9]. Within Mathematics the analysis of limit behaviour of dynamical systems goes back to [7, 8]; see also [2, 5].

In the literature the challenge to relate network behaviour to network structure is usually only addressed for specific models and functions. Here it was addressed in a more general way for general properties of network structure covering a variety of models or functions, thereby providing extra insight in what properties exactly make that specific network structures lead to certain network behaviour.

As an example, a special case of Theorem 3, for one specific model with one specific combination function and speed factor, and only for fully connected networks appeared in [1]. In that paper it does not become clear what it exactly is that makes that the considered model generates that type of behaviour. The much more general formulation presented and proven here is new; it was first announced without proof in the overview paper [12]. It shows that the structure-behaviour relation depends on the one hand on connectivity of the network, but on the other hand also on the ways in which multiple impacts on a single state are aggregated, where it turns out that structure properties such as monotonicity and being scalar-free are crucial.

Theorems 4 and 5 are also new and explore the bounds and range for equilibrium values. Theorem 6 first appeared in [11] in the context of an adaptive network to model a specific preferential attachment principle, but actually covers the much more general setting as provided in the current paper. In [4] also analysis of limit behaviour was addressed for a different class of networks.

In future work also for adaptive networks a similar challenge will be addressed: how do certain properties of adaptation principles lead to certain types of adaptive network behaviour. For example, adaptivity based on a Hebbian learning principle, or based on a homophily principle will be analysed.

References

1. Bosse, T., Duell, R., Memon, Z.A., Treur, J., van der Wal, C.N.: Agent-based modeling of emotion contagion in groups. *Cogn Comput* **7**, 111–136 (2015)
2. Brauer, F., Nohel, J.A.: Qualitative theory of ordinary differential equations. Benjamin (1969)
3. Castellano, C., Fortunato, S., Loreto, V.: Statistical physics of social dynamics. *Rev. Mod. Phys.* **81**, 591–646 (2009)
4. Hendrickx, J.M., Tsitsiklis, J.N.: Convergence of type-symmetric and cut-balanced consensus seeking systems. *IEEE Trans. Automat. Control* **58**(1), 214–218 (2013). Extended version: [arXiv:1102.2361v2](https://arxiv.org/abs/1102.2361v2) [cs.SY]
5. Lotka, A.J. (1956): Elements of Physical Biology. Williams and Wilkins Co. (1924), Dover Publications, 2nd ed.
6. Pearl, J.: Causality. Cambridge University Press (2000)
7. Picard, E.: *Traité d'Analyse*, vol. 1 (1891), vol. 2 (1893)
8. Poincaré, H.: Mémoire sur les courbes définie par une équation différentielle (On curves defined by differential equations) (1881–1882)
9. Treur, J.: Verification of temporal-causal network models by mathematical analysis. *Vietnam J Comput Sci* **3**, 207–221 (2016)
10. Treur, J.: Network-Oriented Modeling: Addressing Complexity of Cognitive, Affective and Social Interactions. Springer Publishers (2016)
11. Treur, J.: Modelling and Analysis of the Dynamics of Adaptive Temporal-Causal Network Models for Evolving Social Interactions. *Computational Social Networks* **4**, article 4, pp. 1–20 (2017)
12. Treur, J.: The Ins and Outs of Network-Oriented Modeling: From Biological Networks and Mental Networks to Social Networks and Beyond. Overview paper based on the Keynote Lecture at the 10th International Conference on Computational Collective Intelligence, ICCCI'18. *Transactions in Computational Collective Intelligence* (2018)



Multilevel Network Reification: Representing Higher Order Adaptivity in a Network

Jan Treur^(✉)

Behavioural Informatics Group, Vrije Universiteit Amsterdam, Amsterdam
The Netherlands
j.treur@vu.nl

Abstract. Network reification occurs when a base network is extended by adding explicit states representing the characteristics defining the structure of the base network. This can be used to explicitly represent network adaptation principles within a network. The adaptation principles may change as well, based on second-order adaptation principles of the network. By reification of the reified network, also such second-order adaptation principles can be explicitly represented. This multilevel network reification construction is introduced and illustrated in the current paper. The illustration focuses on an adaptive adaptation principle from Social Science for bonding based on homophily; here connections are changing by a first-order adaptation principle which itself changes over time by a second-order adaptation principle.

1 Introduction

Reification literally means representing something abstract as a material or concrete thing (Merriam-Webster dictionary), or making something abstract more concrete or real (Oxford dictionaries). It is a notion that is known from different scientific areas; in [13] it has been shown how it can be used to explicitly represent adaptation principles in networks. Reification can also be applied again on reified structures, thus obtaining repeated or multilevel reification. In the current paper such a construction of multilevel reification is applied to networks, illustrated for a Network-Oriented Modeling approach based on temporal-causal networks [11, 12]. By reifying the parameters of the base network structure (connection weights, speed factors, and combination functions) by adding them as states in the extended network, and defining proper temporal-causal relations for themselves and with the other states, a reified network explicitly represents the characteristics of the base network, and how this base network evolves over time based on adaptation principles that change the causal relations. This was illustrated in [13] for a variety of adaptation principles known from Cognitive Neuroscience and Social Science.

Adaptation principles may be adaptive themselves too, according to certain second-order adaptation principles. To model such second-order adaptivity, the reified network can be reified itself, thus providing multilevel reification. This multilevel reification

construction is introduced here; it can be used to model higher order adaptivity of any level. Higher order adaptivity can occur in many forms and applications. Some examples are the following:

- In an adaptive mental network based on Hebbian learning the learning speed or persistence factor may change over time, for example, due to age or due to experiences or medicine. Such second-order adaptation has been discussed for plasticity of the brain and in evolutionary context, for example, see [1, 3, 8].
- In an adaptive social network based on an adaptation principle for bonding based on homophily [7] the similarity measure determining how similar two persons are may change over time, for example, due to age or other varying circumstances.

The homophily context will be used as illustration in the current paper.

In the paper, first in Sect. 2 the Network-Oriented Modeling approach based on temporal-causal networks is briefly summarized. Next, in Sect. 3 the idea of reifying the network structure is briefly introduced. Section 4 introduces a multilevel network reification construction and shows how it can model second-order network adaptivity. This is illustrated by a second-order adaptive network based on a first-order adaptation principle for bonding based on homophily, and a second-order adaptation principle for the characteristics of this first-order adaptation principle. In Sect. 5 example simulations within a developed software environment for this multilevel network reification are presented. Section 7 is a discussion.

2 Temporal-Causal Networks: Structure and Dynamics

A conceptual representation of the network structure of a temporal-causal network model is described by a graph with nodes and directed connections which also includes a number of labels for such a graph: connection weights $\omega_{X,Y}$, speed factors η_Y of states Y , and combination functions for states Y . These three notions form the defining network structure of a temporal-causal network model; see Table 1, upper part; see Fig. 1 for an example of a basic fragment of a network with states X_1, X_2 and Y , and labels $\omega_{X_1,Y}, \omega_{X_2,Y}$ for connection weights, $c_Y(\cdot)$ for combination function, and η_Y for speed factor. To provide sufficient flexibility, a library with a number of standard combination functions are available as options, but also own-defined functions can be added.

In Table 1, lower part it is shown how a conceptual representation describing a network structure defines a numerical representation of the network's dynamics; see also [11], Chap. 2. Here X_1, \dots, X_k are the states with outgoing connections to state Y . This defines the detailed dynamic semantics of a temporal-causal network.

The difference equations in the last row in Table 2 can be used for simulation and mathematical analysis, and can also be written in differential equation format: $dY(t)/dt = \eta_Y [c_Y(\omega_{X_1,Y}X_1(t), \dots, \omega_{X_k,Y}X_k(t)) - Y(t)]$. Examples of combination functions often used for aggregation are the *identity* $\text{id}(\cdot)$ for states with impact from only one other state, the *scaled sum* $\text{ssum}_\lambda(\cdot)$ with scaling factor λ , the minimum function $\text{min}(\cdot)$, and the *advanced logistic sum* combination function $\text{alogistic}_{\sigma,\tau}(\cdot)$ with steepness σ and threshold τ ; see also [11], Chap. 2, Table 10: $\text{id}(V) = V$, $\text{ssum}_\lambda(V_1, \dots, V_k) =$

Table 1. Conceptual and numerical representation of a temporal-causal network structure

Concepts	Notation	Explanation
States and connections	$X, Y, X \rightarrow Y$	Describes the nodes and links of a network structure (e.g., in graphical or matrix format)
Connection weight	$\omega_{X,Y}$	The connection weight $\omega_{X,Y} \in [-1, 1]$ represents the strength of the causal impact of state X on state Y through connection $X \rightarrow Y$
Aggregating multiple impacts	$\mathbf{c}_Y(\cdot)$	For each state Y (a reference to) a combination function $\mathbf{c}_Y(\cdot)$ is chosen to combine the causal impacts of other states on state Y
Timing of the causal effect	η_Y	For each state Y a speed factor $\eta_Y \geq 0$ is used to represent how fast a state is changing upon causal impact
Concepts	Numerical representation	Explanation
State values over time t	$Y(t)$	At each time point t each state Y in the model has a real number value in $[0, 1]$
Single causal impact	$\mathbf{impact}_{X,Y}(t)$ $= \omega_{X,Y} X(t)$	At t state X with connection to state Y has an impact on Y , using weight $\omega_{X,Y}$
Aggregating multiple impacts	$\mathbf{agimpact}_Y(t)$ $= \mathbf{c}_Y(\mathbf{impact}_{X_1,Y}(t), \dots, \mathbf{impact}_{X_k,Y}(t))$ $= \mathbf{c}_Y(\omega_{X_1,Y} X_1(t), \dots, \omega_{X_k,Y} X_k(t))$	The aggregated impact of multiple states X_i on Y at t , is determined using combination function $\mathbf{c}_Y(\cdot)$
Timing of the causal effect	$Y(t + \Delta t) = Y(t) + \eta_Y [\mathbf{agimpact}_Y(t) - Y(t)] \Delta t$ $Y(t) \Delta t = Y(t) + \eta_Y [\mathbf{c}_Y(\omega_{X_1,Y} X_1(t), \dots, \omega_{X_k,Y} X_k(t)) - Y(t)] \Delta t$	The causal impact on Y is exerted over time gradually, using speed factor η_Y

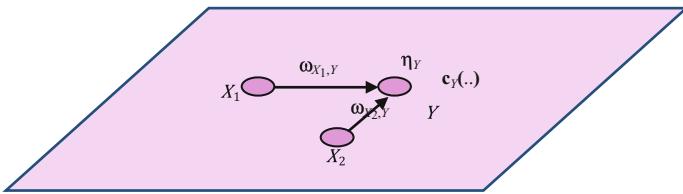


Fig. 1. Fragment of a temporal-causal network structure in a labeled graph representation

$$(V_1, \dots, V_k)/\lambda, \min(V_1, \dots, V_k) = \text{minimum of } V_i, \text{ and } \text{alogistic}_{\sigma, \tau}(V_1, \dots, V_k) = [1/(1 + e^{-\sigma(V_1 + \dots + V_{k-\tau})}) - 1/(1 + e^{\sigma \tau})] (1 + e^{-\sigma \tau}).$$

3 Network Reification

Network reification provides a way to extend a base network by extra states that represent the parameters describing the network structure. The new additional states representing the parameter values for the network structure are what are called *reification states* for the parameters, the parameters are reified by these states. What will be reified in temporal-causal networks in particular are the parameters used to define their network structure: the labels for *connection weights*, *combination functions*, and *speed factors* (see Table 1). For connection weights $\omega_{X_i,Y}$ and speed factors η_Y , their reification states $\Omega_{X_i,Y}$ and H_Y represent the value of them. These reification states are depicted in the upper plane in Fig. 2, whereas the states of the base network are displayed in the lower plane. By having explicit states for the reified characteristics of a network within the reified network, causal relations can be added affecting them, and causal connections from them to related base network states, thus explicitly representing adaptation principles within the (reified) network, as shown in [13]. To this end it is defined how the reification states contribute to an aggregated impact on the related base network state (see the downward arrows in Fig. 2).

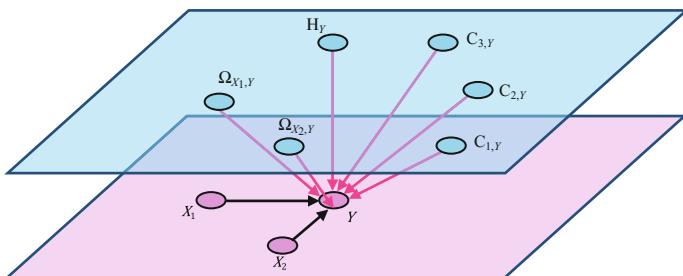


Fig. 2. Network reification for temporal-causal networks: downward connections from reification states to base network states

These downward causal relations and the combination functions will be defined in a generic manner, related to how a specific parameter functions in the overall dynamics as part of the (intended) semantics of a temporal-causal network. More specifically, the general pattern is that each of the reification states $\Omega_{X_i,Y}, H_Y$ and C_Y for connection weights, speed factors and combination functions has a causal connection to state Y in the base network, as they all affect Y . These are the downward arrows from the reification plane to the base plane in Fig. 2. All depicted (downward and horizontal) connections get weight 1. In the extended network the speed factors of the base states are set at 1 too. As the base states have more incoming connections now, new combination functions for them are needed. The different components $C_{1,Y}, C_{2,Y}, \dots$ for C_Y are explained as follows. A countable number of basic combination functions $bc_1(\dots), bc_2(\dots), \dots$ is assumed. From this sequence of basic combination functions $bc_1(\dots), bc_2(\dots), \dots$ for any arbitrary m the finite subsequence up to m can be chosen to be used in a specific application. For example, $bc_1(\dots) = \text{sum}(\dots)$, $bc_2(\dots) = \text{ssum}_\lambda(\dots)$ and $bc_3(\dots) = \text{allogistic}_{\sigma,\tau}(\dots)$. In the base network for each Y combination function weights are assumed: numbers $cw_{1,Y}, cw_{2,Y}, \dots \geq 0$ that may change over time such that the combination function $c_Y(\dots)$ is expressed by:

$$c_Y(t, V_1, \dots, V_k) = [cw_{1,Y}(t)bc_1(V_1, \dots, V_k) + \dots + cw_{m,Y}(t)bc_m(V_1, \dots, V_k)] / [cw_{1,Y}(t) + \dots + cw_{m,Y}(t)]$$

In this way it can be expressed that for Y a weighted average of basic combination functions is used, if more than one of $cw_{i,Y}(t)$ has a nonzero value, or just one basic combination function is selected for $c_Y(\dots)$, if exactly one of the $cw_{i,Y}(t)$ is nonzero. This approach makes it possible, for example, to smoothly switch to another combination function over time by decreasing the value of $cw_{i,Y}(t)$ for the earlier chosen basic combination function and increasing the value of $cw_{i,Y}(t)$ for the new choice of combination function. For each basic combination function weight $cw_{i,Y}(\dots)$ a different reification state $C_{i,Y}$ is added. The value of that state represents the extent to which that basic combination function $bc_i(\dots)$ is applied for state Y . In Treur [12] it has been found that the combination function for base state Y in the reified network is as follows:

$$\begin{aligned} c * Y(S, C_1, \dots, C_m, V_1, \dots, V_k, W_1, \dots, W_k, W) \\ = S(C_1 bc_1(W_1 V_1, \dots, W_k V_k) + \dots + C_m bc_m(W_1 V_1, \dots, W_k V_k)) / (C_1 + \dots + C_m) + (1 - S)W \end{aligned}$$

where S is used for the speed factor reification $H_Y(t)$, C_i for the combination function weight reification $C_{i,Y}(t)$, V_i for the state value $X_i(t)$ of base state X_i , W_i for the connection weight reification $\Omega_{X_i,Y}(t)$, and W for the state value $Y(t)$ of base state Y . This combination function $c * Y(\dots)$ makes that the dynamics of Y within the reified network is described by the following difference equation:

$$\begin{aligned}
Y(t + \Delta t) &= Y(t) \\
&+ [\mathbf{c} *_{\mathbf{Y}} (\mathbf{H}_Y(t), \mathbf{C}_{1,Y}(t), \dots, \mathbf{C}_{m,Y}(t), X_1(t), \dots, X_k(t), \boldsymbol{\Omega}_{X_1,Y}(t), \dots, \boldsymbol{\Omega}_{X_k,Y}(t), Y(t)) - Y(t)] \Delta t \\
&= Y(t) + [\mathbf{H}_Y(t)(\mathbf{C}_{1,Y}(t)\text{bc}_1(\boldsymbol{\Omega}_{X_1,Y}(t)X_1(t), \dots, \boldsymbol{\Omega}_{X_k,Y}(t)X_k(t)) + \dots + \mathbf{C}_{m,Y}(t)\text{bc}_m(\boldsymbol{\Omega}_{X_1,Y}(t) \\
&X_1(t), \dots, \boldsymbol{\Omega}_{X_k,Y}(t)X_k(t))) / (\mathbf{C}_{1,Y}(t) + \dots + \mathbf{C}_{m,Y}(t)) + (1 - \mathbf{H}_Y(t))Y(t) - Y(t)] \Delta t
\end{aligned}$$

Note that in the reification process structures are added which are not reified themselves. A next step is to explore whether and how the structure of the reified network also can be reified within a second-order reification. Structures in the first-order reified network not part the base network are, for example, those used to model specific adaptation principles for evolving networks. In a second-order reified network these adaptation principles can be explicitly represented by states at the second reification level and considered to be adaptive themselves too. In next section it is explored how such second-order reification can be useful to model such adaptive adaptation principles.

4 Multilevel Network Reification

In this section it will be shown how network reification can be applied in a repeated manner, and thus provide multilevel reification. It will be shown how multilevel reification can be useful in modeling evolving networks based on adaptation principles that change over time themselves: adaptive adaptation principles. So, the idea is that the base network evolves through an adaptation principle (called *first-order adaptation principle*), and that adaptation principle changes itself based on another adaptation principle (called *second-order adaptation principle*). The first-order adaptation principle is represented at the first reification level, and the second-order adaptation principle at the second reification level.

The approach will be illustrated through an example inspired by Social Science, in particular concerning the way in which connections between two persons are adapted over time based on similarity between the persons (the homophily adaptation principle as a first-order adaptation principle). One of the notions that plays an important role in the homophily adaptation principle is the homophily *similarity tipping point* or *threshold* τ . This indicates the value such that when the difference between two persons is less than this value, their connection will become stronger, and when the difference is more, their connection will become weaker. Such tipping points are often considered constant, but it may be more realistic when they are considered adaptive over time. This is indeed what is done in the current section. An adaptive form of the homophily adaptation principle is considered where the tipping points change over time by a second-order adaptation principle, and the same applies to the speed factors for different persons of the connection weight adaptation based on the homophily adaptation principle.

In Fig. 3 the architecture of the (second-order) multilevel reified network is shown. The middle plane shows how the connection weights have been reified (first-order reification), and the network relations of the reification states $\Omega_{Xi,Y}$ to the states X_i and Y in the base network (lower plane) are shown. These network relations (including their labels, such as combination functions; see below) define the first-order adaptation

principle based on homophily and its semantics. Note that speed factor reification states H_Y and combination function reification states $C_{i,Y}$ for the base network states Y have been left out of the middle plane here as for this example focusing on the homophily adaptation principle they are considered constant.

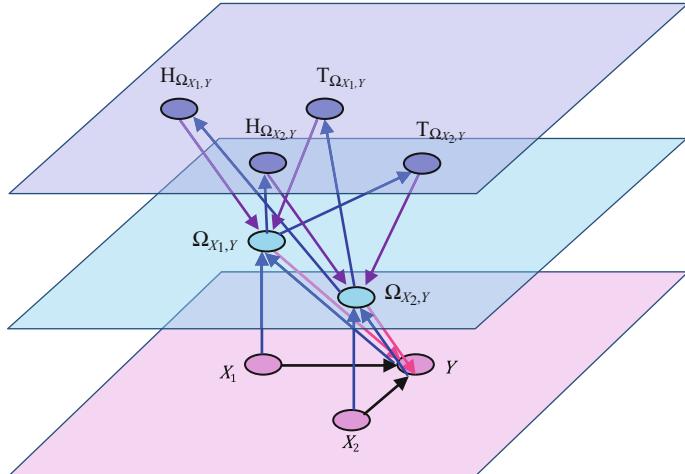


Fig. 3. Multilevel reified conceptual modeling of homophily in social networks with reified tipping point states and connection weight adaptation speed factors at the second reification level (third, upper plane)

On top of this first-order reified network another reification level has been added (the upper plane), in order to get a second-order reified network. Here reification states $T_{\Omega_{X_i,Y}}$ are added for the similarity tipping point characteristic of the homophily adaptation principle, and reification states $H_{\Omega_{X_i,Y}}$ for the speed factor characteristic of the homophily adaptation principle. Also for these reification states (upward and downward) connections have been added that (together with the relevant labels; see below) define the second-order adaptation principle based on them. After having defined the basic architecture, next the labels for the new network are defined.

Base level The speed factors and connection weights in this new network are kept simple: all of them are set at 1. What remains is to define the combination functions; for the base level, this is done by

$$\begin{aligned} \mathbf{c} *_Y (V_1, \dots, V_k, W_1, \dots, W_k) &= \mathbf{c}_Y(W_1 V_1, \dots, W_k V_k) \\ &= [\text{cw}_{1,Y}(t) \text{bc}_1(W_1 V_1, \dots, W_k V_k) + \dots + \text{cw}_{m,Y}(t) \text{bc}_m(W_1 V_1, \dots, W_k V_k)] / \\ &\quad [\text{cw}_{1,Y}(t) + \dots + \text{cw}_{m,Y}(t)] \end{aligned}$$

where V_i stands for $X_i(t)$, and W_i stands for $\Omega_{X_i,Y}(t)$; see also Treur [12]. Here the coefficients $\text{cw}_{i,Y}(t)$ are assumed constant. For example, if $\mathbf{c}_Y(\dots)$ is chosen as the advanced logistic sum function $\text{alogistic}_{\sigma,\tau}(\dots)$, which is the third in the row $\text{bc}_1(\dots), \text{bc}_2(\dots), \dots$, then $\text{cw}_{1,Y} = 0, \text{cw}_{2,Y} = 0$, and $\text{cw}_{3,Y} = 1$, and this obtains:

$$\begin{aligned}\mathbf{c} *_{\mathcal{Y}} (V_1, \dots, V_k, W_1, \dots, W_k) &= \text{bc}_3(W_1 V_1, \dots, W_k V_k) = \mathbf{alogistic}_{\sigma, \tau}(W_1 V_1, \dots, W_k V_k) \\ &= [1/(1 + e^{-\sigma(W_1 V_1 + \dots + W_k V_k - \tau)}) - 1/(1 + e^{\sigma \tau})] (1 + e^{-\sigma \tau})\end{aligned}$$

First reification level Next, consider the combination function defining the homophily adaptation principle at the first reification level (the middle plane); see [7, 11], Chap. 11, Sect. 11.7:

$$\mathbf{c} *_{\Omega_{X_i, Y}} (S, V_1, V_2, T, W) = S (W + \alpha_{\Omega_{X_i, Y}} W (1 - W) (T - |V_1 - V_2|)) + (1 - S) W$$

where S is used for the $\Omega_{X_i, Y}$ speed factor reification $H_{\Omega_{X_i, Y}}(t)$, V_1 for $X_i(t)$, V_2 for $Y(t)$, T for $\Omega_{X_i, Y}$ homophily tipping point reification $T_{\Omega_{X_i, Y}}(t)$, W for connection weight reification $\Omega_{X_i, Y}(t)$, and $\alpha_{\Omega_{X_i, Y}}$ is a homophily modulation factor. This combination function (together with connection weights and speed factor 1) defines the following difference equation for $\Omega_{X_i, Y}$ (see Sect. 2, Table 1):

$$\begin{aligned}\Omega_{X_i, Y}(t + \Delta t) &= \Omega_{X_i, Y}(t) + [H_{\Omega_{X_i, Y}}(t)(\Omega_{X_i, Y} + \alpha \Omega_{X_i, Y}(1 - \Omega_{X_i, Y})(T_{\Omega_{X_i, Y}}(t) - |X_i(t) - Y(t)|)) \\ &\quad + (1 - H_{\Omega_{X_i, Y}}(t))\Omega_{X_i, Y} - \Omega_{X_i, Y}(t)]\Delta t\end{aligned}$$

This indeed makes that an increase in connection weight will take place (in a linear fashion) when the difference in states $|X_i(t) - Y(t)|$ of two persons is less than the tipping point $T_{\Omega_{X_i, Y}}(t)$ and decrease will take place when this difference is more.

Second reification level Finally, consider how the tipping points should be adapted to circumstances. Here the idea is that the tipping point of a person will become higher if the person lacks strong connections (the person becomes less strict) and will become lower if the person already has strong connections (the person becomes more strict). This is modeled using an *average norm weight* v for connections. This can be considered to relate to the amount of time or energy available for social contacts. If the connections of a person are on average stronger than v , downward regulation takes place: the tipping point will become lower and when the connections of this person are on average weaker than v upward regulation takes place: the tipping point will become higher. This is described by the following combination function for the second reification level (the upper plane):

$$\mathbf{c} *_{T_{\Omega_{Y, X_i}}} (W, T) = T + \alpha_{T_{\Omega_{Y, X_i}}} T (1 - T) (v_{T_{\Omega_{Y, X_i}}} - (W_1 + \dots + W_k)/k)$$

where T is used homophily tipping point reification value $T_{\Omega_{Y, X_i}}(t)$ for Ω_{Y, X_i} and W_j for connection weight reification value $\Omega_{Y, X_j}(t)$; $\alpha_{T_{\Omega_{Y, X_i}}}$ is a modulation factor, and $v_{T_{\Omega_{Y, X_i}}}$ is a norm for Y for average connection weight for Ω_{Y, X_i} to Ω_{Y, X_k} . This combination function (together with connection weights and speed factor 1) defines the following difference equation for $T_{\Omega_{Y, X_i}}(t)$ (see Sect. 2, Table 2).

$$\begin{aligned} T_{\Omega_{Y,X_i}}(t + \Delta t) &= T_{\Omega_{Y,X_i}}(t) \\ &+ [T_{\Omega_{Y,X_i}}(t) + \alpha_{T_{\Omega_{Y,X_i}}} T_{\Omega_{Y,X_i}}(t)(1 - T_{\Omega_{Y,X_i}}(t))(v_{T_{\Omega_{Y,X_i}}} - (\Omega_{Y,X_1}(t) + \dots + \Omega_{Y,X_k}(t))/k) - T_{\Omega_{Y,X_i}}(t)]\Delta t \end{aligned}$$

Note that as a slightly different variant the division by k can be left out. Then the norm does not concern the average but the cumulative connection weights. For the opposite connections similarly the following combination function can be used:

$$\mathbf{c} *_{T_{\Omega_{X_i,Y}}} (W, T) = T + \alpha_{T_{\Omega_{X_i,Y}}} T(1 - T)(v_{T_{\Omega_{X_i,Y}}} - (W_1 + \dots + W_k)/k)$$

where T is used for $\Omega_{X_i,Y}$ homophily tipping point reification value $T_{\Omega_{X_i,Y}}(t)$ and W_j for connection weight reification value $\Omega_{X_j,Y}(t)$

$\alpha_{T_{\Omega_{X_i,Y}}}$ is a modulation factor for $T_{\Omega_{X_i,Y}}$

$v_{T_{\Omega_{X_i,Y}}}$ is a norm for average (incoming) connection weights for Y .

For the adaptive connection adaptation speed factor the following combination function can be considered making use of a similar mechanism using a norm for connection weights.

$$\mathbf{c} *_{H_{\Omega_{Y,X_i}}} (W, S) = S + \beta_S S(1 - S)(v_{H_{\Omega_{Y,X_i}}} - (W_1 + \dots + W_k)/k)$$

where S is used for Ω_{Y,X_i} speed factor reification value $H_{\Omega_{Y,X_i}}(t)$, W_j for connection weight reification value $\Omega_{Y,X_i}(t)$, $\beta_{H_{\Omega_{Y,X_i}}}$ is a modulation factor for $H_{\Omega_{Y,X_i}}$, and $v_{H_{\Omega_{Y,X_i}}}$ is a norm for average of (outgoing) connection weights for Y . This combination function defines the following differential equation for $H_{\Omega_{Y,X_i}}$:

$$\begin{aligned} H_{\Omega_{Y,X_i}}(t + \Delta t) &= H_{\Omega_{Y,X_i}}(t) \\ &+ [H_{\Omega_{Y,X_i}}(t) + \beta_{H_{\Omega_{Y,X_i}}} H_{\Omega_{Y,X_i}}(t)(1 - H_{\Omega_{Y,X_i}}(t))(v_{H_{\Omega_{Y,X_i}}} - (\Omega_{Y,X_1}(t) + \dots + \Omega_{Y,X_k}(t))/k) - H_{\Omega_{Y,X_i}}(t)]\Delta t \end{aligned}$$

Also here an opposite variant is possible:

$$\mathbf{c} *_{H_{\Omega_{X_i,Y}}} (W, S) = S + \beta_{H_{\Omega_{X_i,Y}}} S(1 - S)(v_{H_{\Omega_{X_i,Y}}} - (W_1 + \dots + W_k)/k)$$

where S is used for $\Omega_{X_i,Y}$ speed factor reification value $H_{\Omega_{X_i,Y}}(t)$, W_j for connection weight reification value $\Omega_{X_j,Y}(t)$, $\beta_{H_{\Omega_{X_i,Y}}}$ is a modulation factor for $H_{\Omega_{X_i,Y}}$, and $v_{H_{\Omega_{X_i,Y}}}$ is a norm for average of (incoming) connection weights for Y .

5 Simulation Scenarios

The scenarios concern 10 persons X_1 to X_{10} . For the first two scenarios only the outgoing connections of X_1 have been modelled in an adaptive manner, the other connection weights were kept constant. For all simulations $\Delta t = 1$ was used, and the focus in all three scenarios was on the homophily adaptation with constant connection

weight speed factor $H_{\Omega_{X_j, X_i}} = \eta_{\Omega_{X_j, X_i}} = 1$. Moreover, in Scenarios 1 and 2 the focus is only on the adaptive connections from X_1 , and the other connections were kept constant. Scenarios 1 and 2 can be found in the Appendix¹. In Table 2 the main parameter values for Scenario 3 can be found.

Scenario 3: All connections adaptive

For the third scenario all connections were adaptive with main parameters shown in Table 2 and initial connection weight values shown in Table 3. Note that the norm for average connection weight is 0.4 this time.

Table 2. Scenario 3: Main parameter values

Base level		First reification level		Second reification level	
Contagion allogistic steepness σ_{X_i} for X_i	0.8	Homophily modulation factor $\alpha_{\Omega_{X_j, X_i}}$ for Ω_{X_j, X_i}	1	Tipping point speed factor $\eta T_{\Omega_{X_j, X_i}}$ for $T_{\Omega_{X_j, X_i}}$	0.5
Contagion allogistic threshold τ_{X_i} for X_i	0.15	Connection weight speed factor $\eta_{\Omega_{X_j, X_i}}$ for Ω_{X_j, X_i}	1	Tipping point modulation factor $\alpha T_{\Omega_{X_j, X_i}}$ for $T_{\Omega_{X_j, X_i}}$	0.4
Speed factor η_{X_i} for base state X_i	0.5			Tipping point connection norm $\sqrt{T_{\Omega_{X_j, X_i}}}$ for $T_{\Omega_{X_j, X_i}}$	0.4

Table 3. Scenario 3: Initial connection weights

Connections	X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8	X_9	X_{10}
X_1		0.5	0.3	0.1	0.2	0.6	0.5	0.2	0.3	0.4
X_2	0.5		0.6	0.3	0.4	0.7	0.7	0.9	0.5	
X_3	0.3	0.6		0.7	0.7	0.4	0.4		0.6	0.8
X_4	0.6	0.4	0.6		0.4	0.6	0.7	0.8		0.9
X_5	0.2	0.5		0.7		0.4		0.4	0.9	0.4
X_6	0.6	0.6	0.7	0.5			0.7	0.7	0.5	0.7
X_7	0.2	0.8	0.6	0.7	0.6	0.7		0.7		
X_8	0.6	0.5		0.4		0.6	0.5		0.4	0.5
X_9	0.6		0.6	0.7	0.4		0.7			0.6
X_{10}	0.6	0.7		0.7	0.4	0.6		0.8		

In Figs. 4, 5, 6 7 the simulation outcomes are shown. As can be seen in Fig. 7 eventually all connection weights converge to 0 or 1. Figure 4 shows in particular the values of the connection weights from X_1 , and their average, and Fig. 5 shows the corresponding tipping points.

¹ <http://www.few.vu.nl/~treur/AppendixSimulationScenarios.pdf>.

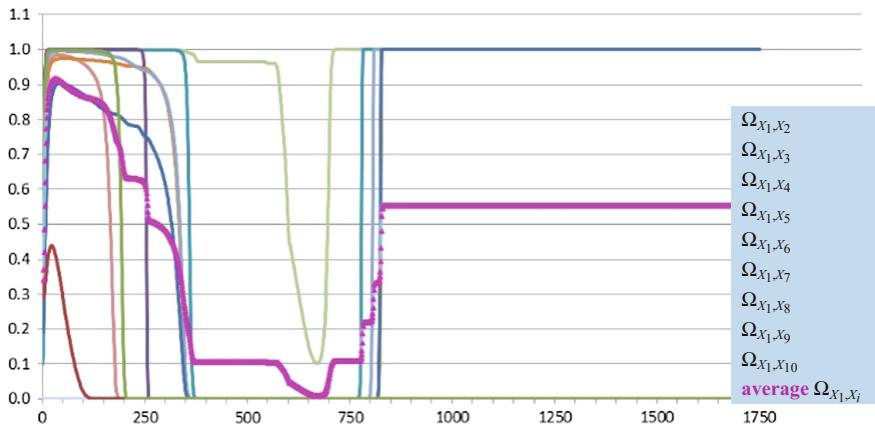


Fig. 4. Scenario 3: Adaptive weights of outgoing connections from X_1 over time, with the thick pink line showing the average weight for X_1

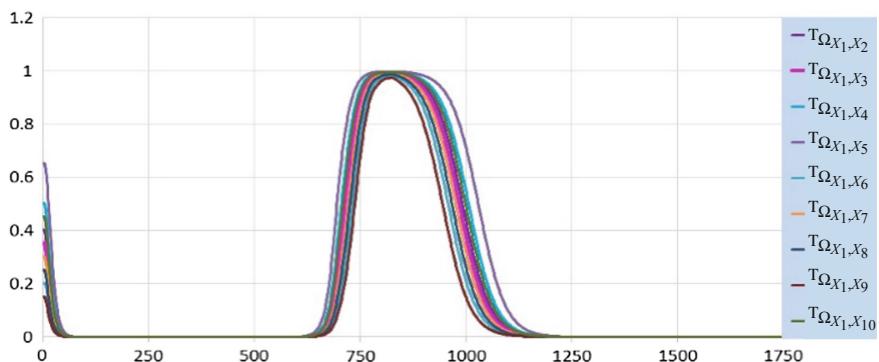


Fig. 5. Scenario 3: Adaptive tipping points $T_{\Omega_{X_1, X_j}}$ over time

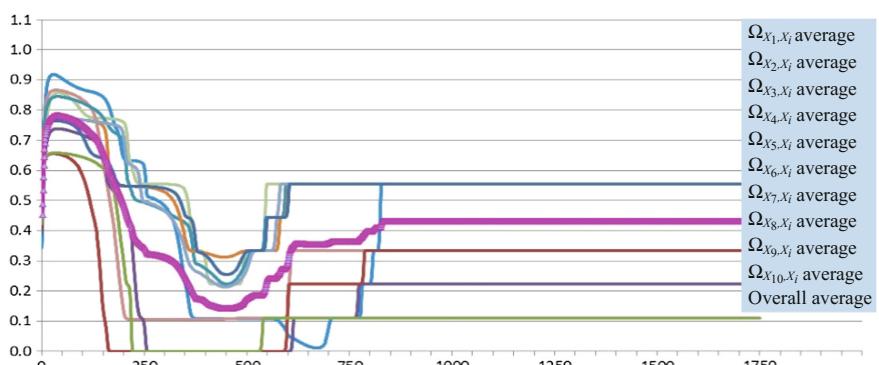


Fig. 6. Scenario 3: Average connection weights for each of X_1 – X_{10} and of all connections over time

Note that Fig. 6 shows that in the emerging process eventually the average connection weights per person stick in some seemingly mysterious manner to a discrete set of values: $0.111111(X_{10})$, $0.222222(X_5)$, $0.333333(X_3, X_9)$, and $0.555555(X_1, X_2, X_4, X_6, X_7, X_8)$, all multiples of 0.111111 ; the overall average ends up in 0.433333 (recall that the norm $v_{T\Omega_{X_i, X_j}}$ for average connection weight for each person was 0.4). Also in other simulations this discrete set of multiples of 0.111111 emerges. In Sect. 6 it will be analysed where these values come from.

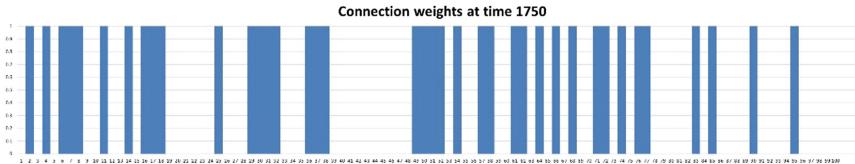


Fig. 7. Scenario 3: All connection weights are 0 or 1 at time 1750

In Fig. 7 it is shown that all connection weights converge to 0 or 1. This will also be analysed in Sect. 6. For the tipping points, for all outgoing connections of X_1 they converge to 0 (see also Fig. 6), and for all outgoing connections of the other persons they converge to 1.

6 Analysis of Equilibrium Values

In this section it is analysed what are the possible values to which certain states in the second-order reified network may converge. This is done by an analysis of equilibrium states. A state Y has a *stationary point* at t if $\mathbf{d}Y(t)/dt = 0$. The network is in *equilibrium* at t if every state Y of the model has a stationary point at t . Considering the differential equation for a temporal-causal network model (and assuming a nonzero speed factor) a more specific criterion can be found:

Criterion for a stationary point in a temporal-causal network

Let Y be a state with speed factor $\eta_Y > 0$ and X_1, \dots, X_k the states with outgoing connections to state Y . Then Y has a stationary point at t iff $\mathbf{c}_Y(\omega_{X_1, Y} X_1(t), \dots, \omega_{X_k, Y} X_k(t)) = Y(t)$.

Note that this can be applied to the states at all levels in the second-order reified network, in particular to the first and second reification level. First, in an equilibrium state, the above criterion applied to tipping point reification states at the second reification level is as follows:

$$\mathbf{c} *_{T\Omega_{X_i, X_j}} (W, T\Omega_{X_i, X_j}(t)) = T\Omega_{X_i, X_j}(t)$$

This provides the following equation

$$T_{\Omega_{X_i, X_j}}(t) + \alpha_{T_{\Omega_{X_i, X_j}}} T_{\Omega_{X_i, X_j}}(t)(1 - T_{\Omega_{X_i, X_j}}(t))(v_{T_{\Omega_{X_i, X_j}}} - (W_1 + \dots + W_k)/k) = T_{\Omega_{X_i, X_j}}(t)$$

which can be rewritten as follows

$$\alpha_{T_{\Omega_{X_i, X_j}}} T_{\Omega_{X_i, X_j}}(t)(1 - T_{\Omega_{X_i, X_j}}(t))(v_{T_{\Omega_{X_i, X_j}}} - (\Omega_{X_i, X_1}(t) + \dots + \Omega_{X_i, X_k}(t))/k) = 0$$

Assuming $\alpha_{T_{\Omega_{X_i, X_j}}}$ nonzero, this equation has three solutions:

$$T_{\Omega_{X_i, X_j}}(t) = 0 \quad \text{or} \quad T_{\Omega_{X_i, X_j}}(t) = 1 \quad \text{or} \quad (\Omega_{X_i, X_1}(t) + \dots + \Omega_{X_i, X_k}(t))/k = v_{T_{\Omega_{X_i, X_j}}}$$

Similarly, the criterion can be applied to connection weights at the first reification level:

$$\mathbf{c} *_{\Omega_{X_i, X_j}} (H_{\Omega_{X_i, X_j}}(t), X_i(t), Y(t), T_{\Omega_{X_i, X_j}}(t), \Omega_{X_i, X_j}(t)) = \Omega_{X_i, X_j}(t)$$

This provides the following equation

$$\begin{aligned} & H_{\Omega_{X_i, X_j}}(t)(\Omega_{X_i, X_j}(t) \\ & + \alpha_{\Omega_{X_i, X_j}} \Omega_{X_i, X_j}(t)(1 - \Omega_{X_i, X_j}(t))(T_{\Omega_{X_i, X_j}}(t) - |X_i(t) - Y(t)|)) + (1 - H_{\Omega_{X_i, X_j}}(t))\Omega_{X_i, X_j}(t) \\ & = \Omega_{X_i, X_j}(t) \end{aligned}$$

which can be rewritten as follows

$$\begin{aligned} & H_{\Omega_{X_i, X_j}}(t)\alpha_{\Omega_{X_i, X_j}}\Omega_{X_i, X_j}(t)(1 - \Omega_{X_i, X_j}(t))(T_{\Omega_{X_i, X_j}}(t) - |X_i(t) - Y(t)|) + \Omega_{X_i, X_j}(t) = \Omega_{X_i, X_j}(t) \\ & H_{\Omega_{X_i, X_j}}(t)\alpha_{\Omega_{X_i, X_j}}\Omega_{X_i, X_j}(t)(1 - \Omega_{X_i, X_j}(t))(T_{\Omega_{X_i, X_j}}(t) - |X_i(t) - Y(t)|) = 0 \end{aligned}$$

Assuming $\alpha_{T_{\Omega_{X_i, X_j}}}$ nonzero, this equation has four solutions:

$$H_{\Omega_{X_i, X_j}}(t) = 0 \quad \text{or} \quad \Omega_{X_i, X_j}(t) = 0 \quad \text{or} \quad \Omega_{X_i, X_j}(t) = 1 \quad \text{or} \quad |X_i(t) - X_j(t)| = T_{\Omega_{X_i, X_j}}(t)$$

Combined with the three solutions for $T_{\Omega_{X_i, X_j}}(t)$ the matrix in Table 4 can be found.

Table 4. Overview of the different solutions of the equilibrium equations for Ω_{X_i, X_j} and $T_{\Omega_{X_i, X_j}}$

$T_{\Omega_{X_i, X_j}}(t) = 0$	$H_{\Omega_{X_i, X_j}}(t) = 0$	$\Omega_{X_i, X_j} = 0$	$\Omega_{X_i, X_j} = 1$	$ X_i - X_j = T_{\Omega_{X_i, X_j}}(t)$
$T_{\Omega_{X_i, X_j}}(t) = 0$	$H_{\Omega_{X_i, X_j}}(t) = 0$	$\Omega_{X_i, X_j} = 0$	$\Omega_{X_i, X_j} = 1$	$X_i = X_j$
	$T_{\Omega_{X_i, X_j}}(t) = 0$	$T_{\Omega_{X_i, X_j}}(t) = 0$	$T_{\Omega_{X_i, X_j}}(t) = 0$	$T_{\Omega_{X_i, X_j}}(t) = 0$
$T_{\Omega_{X_i, X_j}}(t) = 1$	$H_{\Omega_{X_i, X_j}}(t) = 0$	$\Omega_{X_i, X_j} = 0$	$\Omega_{X_i, X_j} = 1$	$X_i = 0 \text{ and } X_j = 1 \text{ or } X_i = 1 \text{ and } X_j = 0$
	$T_{\Omega_{X_i, X_j}}(t) = 1$	$T_{\Omega_{X_i, X_j}}(t) = 1$	$T_{\Omega_{X_i, X_j}}(t) = 1$	$T_{\Omega_{X_i, X_j}}(t) = 1$
$\Sigma_m \Omega_{X_i, X_m} / k = v_{T_{\Omega_{X_i, X_j}}}$	$H_{\Omega_{X_i, X_j}}(t) = 0$	$\Sigma_m \Omega_{X_i, X_m} / k = v_{T_{\Omega_{X_i, X_j}}}$	$\Sigma_m \Omega_{X_i, X_m} / k = v_{T_{\Omega_{X_i, X_j}}}$	$ X_i - X_j = T_{\Omega_{X_i, X_j}}(t)$

In cases that not $|X_i - X_j| = T_{\Omega_{X_i, X_j}}(t)$, and $H_{\Omega_{X_i, X_j}}(t)$ is nonzero (which was the case for the simulations displayed in Sect. 5), all Ω_{X_i, X_j} are 0 or 1, so with $k = 9$ the average $(\Omega_{X_1, X_1} + \dots + \Omega_{X_9, X_9})/k$ is a multiple of 1/9, which in the simulation case the norm $v_{T_{\Omega_{X_i, X_j}}} = 0.4$ is not. Therefore the cases $(\Omega_{X_1, X_1}(t) + \dots + \Omega_{X_9, X_9}(t))/k = v_{T_{\Omega_{X_i, X_j}}}$ cannot actually occur, so then $T_{\Omega_{X_i, X_j}}(t) = 0$ or $T_{\Omega_{X_i, X_j}}(t) = 1$ and $\Omega_{X_i, X_j}(t) = 0$ or $\Omega_{X_i, X_j}(t) = 1$ are the only solutions, which is also shown in the simulations (e.g., see Fig. 7), and indeed all averages are multiples of $1/9 = 0.111111$, as observed above (see Fig. 6). This explains the specific discrete set of numbers $0.111111, 0.222222, 0.333333, \dots$ observed in the simulations. Note that although in general the reified speed factor $H_{\Omega_{X_i, X_j}}(t)$ may be assumed nonzero, but there are also specific processes in which it converges to 0, for example, the temperature in simulated annealing.

7 Discussion

The construction of network reification can provide advantages similar to those found for reification in modeling and programming languages in other areas of AI and Computer Science; e.g., [2, 4–6, 9, 10, 14]. A reified network including an explicit representation of the network structure enables to model dynamics of the original network by dynamics within the reified network. In this way an adaptive network can be represented by a non-adaptive network. In [13] it is shown how network reification provides a unified manner of modelling adaptation principles, and allows comparison of such principles across different domains.

In the current paper it was shown how a multilevel reified network can be obtained by reifying an already reified network. It was shown how first-order adaptation principles which are explicitly represented at the first reification level can be made adaptive themselves by adding explicit representations of their characteristics at the second reification level, and specifying a second-order adaptation principle at that level. Thus a network architecture is obtained for adaptive adaptation principles: (second-order) adaptation principles applied to (first-order) adaptation principles. This was illustrated in particular for a first-order adaptation principle based on homophily from Social Science [7] represented at the first reification level, and a second-order adaptation principle describing change of characteristics of this first-order adaptation principle. This second order adaptation was applied to the characteristics similarity tipping point and speed factor for the first-order adaptation principle based on homophily.

Network reification will increase complexity, but for one reification step this will at most be quadratic in the number of nodes N ; see [13]. If this is applied in an iterative manner for second-order network reification, then the increase in complexity is still polynomial: at most in the fourth power of the number of nodes: $(N^2)^2 = N^4$. Can this iteration still be continued further, thus obtaining n th order reification for any n ? Yes, theoretically there is no end in this. But also practically, for example, in the case used as illustration in the current paper, the parameter $v_{T_{\Omega_{X_i, X_j}}}$ for the norm of the average connection weight for the tipping point adaptation used as characteristic at the second reification level still could be made adaptive (e.g., related to how busy someone is) and

reified at a third reification level. For third-order reification the increase in complexity is still polynomial: at most in the order of $((N^2)^2)^2 = N^8$. If n reification levels are added, then it is in the order of $N^{(2n)}$, which is still polynomial in N , but double exponential in n . The latter may suggest to limit the number of reification levels in practical applications to just a few, or in each reification step add only a few new reification states: for each step reification can be done in a partial manner as well. For example, if only speed factors are reified, the number of states will only increase in a linear way: one extra state for each existing state.

Note that in an n th order reified network still there will be network structures introduced in the last step from $n - 1$ to n that have no reification within the n th order reified network. From a theoretical perspective it can also be considered to repeat the construction infinitely many times, for all natural numbers n ; this can be called ω -order reification, where ω is the ordinal for the natural numbers. Then an infinite network is obtained, which is theoretically well-defined as a mathematical structure. All network structures in this ω -order reified network are reified within the network itself, so it is closed under reification. But it may not be clear whether such an ω -order construction has a useful application in practice, or be used to explore theoretical research questions. This may be a subject for future research.

References

1. Arnold, S., Suzuki, R., Arita, T.: Selection for representation in higher-order adaptation. *Mind. Mach.* **25**(1), 73–95 (2015)
2. Bowen, K.A., Kowalski, R.: Amalgamating language and meta-language in logic programming. In: Clark, K., Tarnlund, S. (eds.) *Logic Programming*, pp. 153–172. Academic Press, New York (1982)
3. Daimon, K., Arnold, S., Suzuki, R., Arita, T.: The emergence of executive functions by the evolution of second-order learning. *Artif. Life Robot.* **22**, 483–489 (2017)
4. Demers, F.N., Malenfant, J.: Reflection in logic, functional and object-oriented programming: a short comparative study. In: IJCAI'95 Workshop on Reflection and Meta-Level Architecture and their Application in AI, pp. 29–38 (1995)
5. Galton, A.: Operators versus arguments: the ins and outs of reification. *Synthese* **150**, 415–441 (2006)
6. Hofstadter, D.R.: Gödel, Escher, Bach. Basic Books, New York (1979)
7. McPherson, M., Smith-Lovin, L., Cook, J.M.: Birds of a feather: homophily in social networks. *Annu. Rev. Sociol.* **27**, 415–444 (2001)
8. Robinson, B.L., Harper, N.S., McAlpine, D.: Meta-adaptation in the auditory midbrain under cortical influence. *Nat. Commun.* **7**, 13442 (2016)
9. Sterling, L., Shapiro, E.: The Art of Prolog. MIT Press (Ch 17, pp. 319–356) (1996)
10. Sterling, L., Beer, R.: Meta interpreters for expert system construction. *J. Logic Program.* **6**, 163–178 (1989)
11. Treur, J.: Network-Oriented Modeling: Addressing Complexity of Cognitive. Springer Publishers, Affective and Social Interactions (2016)
12. Treur, J.: The Ins and outs of network-oriented modeling: from biological networks and mental networks to social networks and beyond. In: *Transactions on Computational*

- Collective Intelligence. Lecture Notes in Computer Science, Springer Publishers. Contents of Keynote Lecture at ICCCI'18 (2018)
- 13. Treur, J.: Network reification as a unified approach to represent network adaptation principles within a network. In: Proceedings of the 7th International Conference on Natural Computing. Lecture Notes in Computer Science. Springer Publishers (2018)
 - 14. Weyhrauch, R.W.: Prolegomena to a theory of mechanized formal reasoning. *Artif. Intell.* **13**, 133–170 (1980)



An Exploration of the Network Installation and Recovery Problem with Blackstart Nodes

Kayla S. Cummings, Janie L. Neal, Andi Chen, and Tzu-Yi Chen^(✉)

Computer Science Department, Pomona College, Claremont, CA 91711, USA
`{ksc72014, jlnn2015, ac002014}@mymail.pomona.edu, tzuyi.chen@pomona.edu`

Abstract. The Neighbor Aided Network Installation Problem asks how best to install the nodes in a network under the assumption that the cost of installing a node depends solely on the number of its neighbors which have been previously installed. We study a version which incorporates blackstart nodes into the model. Under the assumption of a decreasing convex cost function and a single blackstart node, we solve the problem for subclasses of almost-complete and augmented-tree networks. We also describe two heuristics and present experimental results on real world networks.

Keywords: Network models · Network recovery · Blackstart nodes

1 Introduction

The electric power transmission grid in the contiguous United States consists of approximately 120,000 miles of transmission lines operated by approximately 500 companies. This complex interconnected network, regulated by the North American Electric Reliability Corporation, delivers power from generators to individual consumers [10].

Ideally this network runs smoothly at all times. In practice the flow of electricity from generators to consumers can be interrupted by power outages, also known as blackouts. Because of the network structure, outages can cascade, leading to blackouts that affect large geographic areas. Once a blackout has occurred, the goal is to restore power as efficiently as possible. This can require not only repairing physical damage by, for example, clearing fallen tree branches, but also restarting power generators in an appropriate order. Approaches for determining how to restart the power grid after an outage include incorporating a subset of the variables into a complex mixed integer linear programming problem [9], applying heuristics motivated by complex network theory [7], agent-based approaches [6, 8].

In contrast to this heuristic work, a more theoretical approach is taken in [4, 5]. Here a graph problem which models a simplified version of the power recovery problem is described and studied.

In this paper we extend the model proposed in [4] by incorporating blackstart nodes, which represent nodes in the network that must be started first because they contain the small diesel generators that some power stations have to facilitate a restart after a wide-area power outage. After such a power outage, these nodes are the only ones that can be restarted without any of their neighbors being restarted first. The question then becomes: given a network and a set of blackstart nodes, what is the most cost-efficient way to restart the entire network? We answer this question for a restricted set of networks. We also describe two heuristics and discuss preliminary experimental results on real world networks.

2 Background

An instance of the *Neighbor Aided Network Installation Problem* (NANIP) as defined in [4] consists of a network $G = (V, E)$ with an assigned cost function $f : \mathbb{N}_0 \rightarrow \mathbb{R}_+ \cup \{0\}$. If the highest-degree node in V has degree $\deg(V)$, then the domain of f is defined by $\mathbb{N}_0 = \{n \in \mathbb{Z} : 0 \leq n \leq \deg(V)\}$. The cost of installing a node after k of its neighbors have already been installed is $f(k)$. Given a permutation σ of the nodes in V , the cost of σ is denoted $C_\sigma(G)$ and is defined as the sum of the costs of installing each node in G in that order. The goal is to find a minimum-cost permutation of the nodes.

In [4] they show the general NANIP problem is NP-hard and then focus on the case where the cost function f is decreasing convex. A function $f : \mathbb{N}_0 \rightarrow \mathbb{R}_+ \cup \{0\}$ is *decreasing convex* if $f(i) - f(i+1) \geq f(j) - f(j+1)$ for all $i, j \in \mathbb{N}_0$ with $j \geq i$. This assumption models an essential aspect of network installation. As the number of previously installed nodes increases, the installation cost of one extra node decreases due to the extra neighbor aid. Convexity captures diminishing returns: the benefit of extra neighbor aid decreases as the number of neighbors increases. In other words, the benefit of adding a fifth neighbor that is already installed is more significant than the benefit of adding a hundredth neighbor.

In [4] they prove a lower bound on the cost of any permutation for the case where f is decreasing convex. As a corollary, if the graph G is a tree graph, they prove the optimal permutation σ attains a cost of:

$$C_\sigma(G) = f(0) + (n - 1)f(1) \quad (1)$$

Then, in [5], it is shown that the NANIP problem remains NP-hard even for decreasing convex cost functions.

3 Blackstart Nodes

The NANIP-Blackstart problem extends the model in [4] by adding blackstart nodes. As previously described, blackstart nodes represent the only nodes that can be restarted before any of their neighbors. In this situation, all solutions σ

must be *valid* permutations, which means every node that is not a blackstart node must have at least one neighbor appear earlier in the permutation.

For now, assume there is a single blackstart node v_b and that the cost of installing v_b is $f(0)$.

Proposition 1. *NANIP-Blackstart with a single blackstart node is NP-hard*

Proof. Consider a decision version of NANIP-Blackstart in which a yes instance means there exists an ordering with cost less than c . An instance of NANIP $\{G = (V, E), f, c\}$ can be reduced to an instance of NANIP-Blackstart as follows. Let $G' = (V', E')$, where $V' = V \cup v_0$ and $E' = E \cup F$ where $F = \{(v, v_0) | v \in V\}$. Let v_0 be the single blackstart node in B . Define f' so $f'(k) = f(k - 1) + 1$ for all $k \geq 1$ and $f'(0) = 0$. Let $c' = c$.

A valid ordering for the instance of NANIP-Blackstart is a minimum cost ordering for the instance of NANIP without the blackstart node. The blackstart version means the installed nodes must always form a connected component. However, since every node in V is connected to v_0 by an edge that costs nothing, this does not restrict the valid orderings in the instance of NANIP-Blackstart. \square

We now turn to proving exact solutions for restricted subclasses of graphs before presenting the results of experiments using two heuristics.

3.1 Augmented Trees

Because many real world electrical networks have low treewidth [2], it is useful to study the NANIP-Blackstart problem on trees and on augmented tree graphs.

Proposition 2. *Let T be a tree with n nodes and let σ be a valid permutation of the nodes. Then*

$$C_\sigma(T) = f(0) + (n - 1)f(1). \quad (2)$$

Proof. The constraint added by including a blackstart node means the installed portion of the tree must always be connected. This gives the minimum cost solution for a tree without a blackstart node [4]. \square

Trees Augmented with Distinct Cycles

Definition 1. *A set of k cycles are distinct if no two cycles share an edge. In Fig. 1, the two cycles (A, B, C) and (B, D, E) are distinct. The two cycles (A, B, C) and (B, C, F, E) are not distinct.*

Proposition 3. *Let T be a tree graph with n vertices. Let T_m be a graph with m distinct cycles that is created by adding m edges to T . The cost of any valid permutation σ is given by the formula:*

$$C_\sigma(T_m) = f(0) + (n - m - 1)f(1) + mf(2). \quad (3)$$

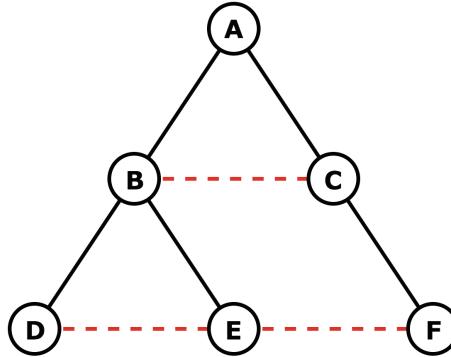


Fig. 1. A tree augmented with three edges (drawn as dashed lines).

Proof. Partition V into 3 sets: V_B , V_C , and V_A where nodes in V_C are nodes that are the last node installed in a cycle, V_B contains only the blackstart node, and $V_A = V \setminus \{V_B \cup V_C\}$. The cost of installing the node in V_B is $f(0)$ by definition. The cost of installing each node in V_C is $f(2)$, while the cost of each node in V_A is $f(1)$. \square

Wheel Graphs

Definition 2. Let $C = (V, E)$ be a cycle graph and let $|V| = n$. Let $V' = V \cup v'$ and let $E' = E \cup \{(v', v) | v \in V\}$. We define $W_m = (V', E')$ as a wheel graph on $m = n + 1$ vertices (Fig. 2).

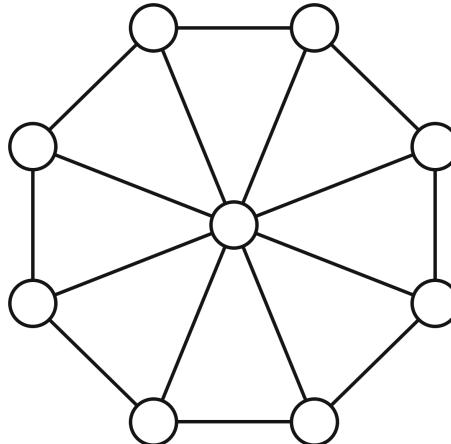


Fig. 2. The wheel graph W_9 .

Proposition 4. Let W_n be a wheel graph and let the central vertex be the blackstart node. The optimal permutation σ has cost

$$C_\sigma(W_n) = f(0) + f(1) + (n - 3)f(2) + f(3). \quad (4)$$

Proof. The second node in σ incurs a cost of $f(1)$. If σ goes around the cycle taking the vertices in order, Eq. 4 gives the overall cost. If we change this ordering so that a non-contiguous vertex is chosen from the cycle at any point, this would convert two of the $f(2)$ costs to an $f(1) + f(3)$. The latter is more expensive since f is a decreasing convex function. \square

Proposition 5. Let W_n be a wheel graph such that $n \geq 3$. Let v_c be the central node and let one of the nodes in the cycle be the blackstart node. Depending on the function f , the optimal permutation σ has cost either

$$C_\sigma(W_n) = f(0) + f(1) + (n - 3)f(2) + f(3) \quad (5)$$

or

$$C_\sigma(W_n) = f(0) + (n - 3)f(1) + f(3) + f(n - 2). \quad (6)$$

Proof. Let ρ be any valid permutation. Partition the vertices of W_n into 3 sets V_1 , $\{v_c\}$, and V_2 where V_1 is the set of nodes installed before v_c and V_2 is the set of nodes installed after v_c . Let $y = |V_1|$.

There are three cases to analyze. Case 1 is when $1 \leq y < n - 2$, case 2 is when $y = n - 2$, and case 3 is when $y = n - 1$.

Case 1: If $1 \leq y < n - 2$, then the cost of installing the blackstart node is $f(0)$ and the cost to install all the other nodes in V_1 is $(y - 1)f(1)$. The cost of installing v_c is $f(y)$. Following the same reasoning as in the previous proof, installing the nodes in V_2 is best done by taking them in order for a cost of $(n - y - 2)f(2) + f(3)$.

Summing these costs gives:

$$f(0) + (y - 1)f(1) + (n - y - 2)f(2) + f(3) + f(y) \quad (7)$$

Substituting $y = k + 1$ into Eq. 7 gives:

$$f(0) + (k)f(1) + (n - k - 3)f(2) + f(3) + f(k + 1) \quad (8)$$

Subtracting Eq. 7 with $y = k$ from Eq. 8 gives $f(1) - f(2) + f(k + 1) - f(k)$, which is positive since f is a decreasing convex function. Hence Eq. 7 is minimized when $y = 1$ and Eq. 5 gives the minimum cost.

Case 2: If $y = n - 2$, then the cost of installing the blackstart node is $f(0)$, the cost to install all the other nodes in V_1 is $(n - 3)f(1)$, the cost of installing v_c is $f(n - 2)$, and the cost of installing the single node in V_2 is $f(3)$. The total cost is given in Eq. 6.

Case 3: If $y = n - 1$, then v_c is the last node installed. The cost of installing the blackstart node is $f(0)$ and the cost to install all the other nodes in V_1 is $(n - 3)f(1) + f(2)$. The cost of installing v_c is $f(n - 1)$.

Summing these costs gives:

$$f(0) + (n - 3)f(1) + f(2) + f(n - 1) \quad (9)$$

Subtracting Eq. 6 from Eq. 9 gives $f(2) - f(3) + f(n - 1) - f(n - 2)$, which is non-negative for any $n \geq 3$ since f is a decreasing convex cost function. As a result, Case 2 is always less expensive than Case 3 if $n \geq 3$. \square

3.2 Almost-Complete Networks

The low treewidth of many real-world networks leads to certain structural vulnerabilities [2]. Because it can be less expensive to add edges that are near each other geographically, power networks can evolve to have dense local substructures. As a result, we also considered the NANIP-Blackstart problem on complete and almost-complete networks.

Proposition 6. *Let $K_n = (V, E)$ be a complete graph with $|V| = n$. The cost of installing K_n for any σ is*

$$C_\sigma(K_n) = \sum_{i=0}^{n-1} f(i). \quad (10)$$

Proof. By definition, v_1 is the blackstart node and has cost $f(0)$. Consider node v_{i+1} in the permutation σ . Because K_n is complete, v_{i+1} will be adjacent to each of the i vertices before it in the permutation. Therefore, the cost of installing v_{i+1} is $f(i)$. By induction on i , the cost of any σ is Eq. 10 (Fig. 3). \square

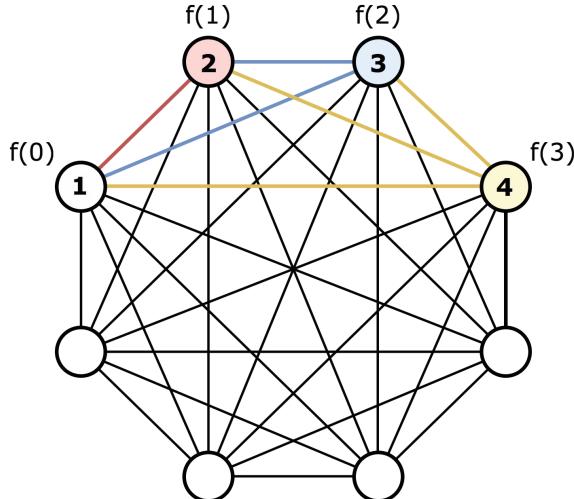


Fig. 3. A K_8 mid-installation to illustrate Proposition 6. Nodes are enumerated according to installation order and labeled by installation cost.

Proposition 7. Let $S_n = (V, E)$ be a graph with $|V| = n$ and $|E| = \frac{n(n-1)}{2} - 1$. In other words, S_n is a complete graph with one missing edge. Let σ be the minimum cost permutation for installing S_n . The cost of σ is given by:

$$C_\sigma(S_n) = \left(\sum_{i=0}^{n-2} f(i) \right) + f(n-2). \quad (11)$$

In addition, one of the two nodes with degree $n-2$ must be installed last in σ .

Proof. Let u_1 and u_2 be the two vertices of degree $n-2$ in S_n . Without loss of generality assume u_1 appears before u_2 in the minimum cost ordering σ and let u_2 be the k^{th} node in σ .

We know $k \geq 2$ because u_2 must be preceded in σ by u_1 . In addition, $k \geq 3$ because even if $v_b = u_1$, u_2 cannot be installed immediately after u_1 since such an ordering would not be valid. Thus, $3 \leq k \leq n$.

The cost of installation from the first node through node $k-1$ is $\sum_{i=0}^{k-2} f(i)$ for the reasons given in Eq. 10. This is because every node including u_1 will be adjacent to every node that has been previously installed.

However, u_2 is not adjacent to u_1 , which has already been installed, so u_2 has cost $f(k-2)$. Thus,

$$C_\sigma(S_n) = \left(\sum_{i=0}^{n-1} f(i) \right) + f(k-2) - f(k-1). \quad (12)$$

Because f is a decreasing convex cost function, $f(j-2) - f(j-1) \geq f(j-1) - f(j)$ for all j . Repeated substitution shows that the above equation is minimized when $k = n$, giving us Eq. 11. \square

Proposition 8. Let S_{nk} be a complete graph with k removed edges. Furthermore, assume the k removed edges are vertex-disjoint so that the minimum degree of a node in S_{nk} is $n-2$. Let σ be a minimum cost valid permutation for installing S_{nk} . The cost of σ is given by:

$$C_\sigma(S_{nk}) = \sum_{i=0}^{(n-1)-k} f(i) + \sum_{j=1}^k f(n-j). \quad (13)$$

In addition, one vertex from each of the k removed edges must be among the last k nodes in σ (Fig. 4).

Proof. Let (u_i, v_i) be the pair of vertices that define the i^{th} removed edge in S_{nk} . Without loss of generality, always install u_i before v_i , and always install v_i before v_{i+1} . Note that u_i is adjacent to all nodes in $V \setminus \{v_i\}$, and vice versa.

Let v_1, \dots, v_k be respectively installed as the $m_1^{\text{th}}, \dots, m_k^{\text{th}}$ nodes σ , with $m_1 < \dots < m_k$. If u_i is the p^{th} installed node, then the cost of installing u_i is $f(p-1)$ because u_i is adjacent to all $p-1$ previously installed nodes. However, if

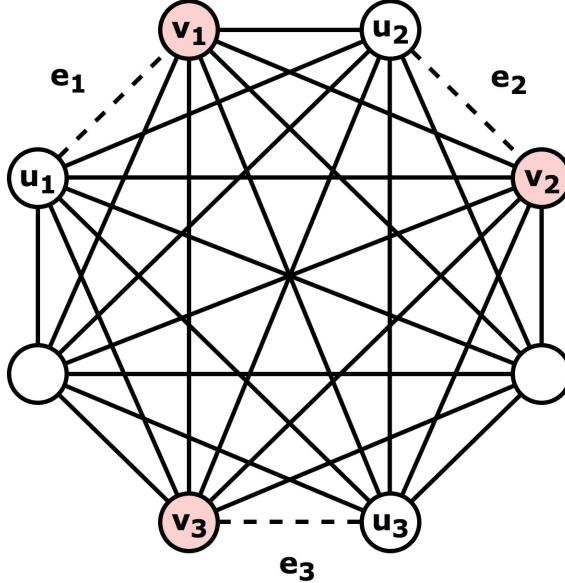


Fig. 4. A K_8 with three vertex-distinct removed edges e_1, e_2, e_3 to illustrate Proposition 8. Without loss of generality, the permutation (v_1, v_2, v_3) defines the last three nodes of all valid, minimum cost permutations for this K_8 network.

v_i is the q^{th} installed node, then the cost of v_i is $f(q - 2)$ because v_i is adjacent to all previously installed nodes except for u_i . Then the cost of installing S_n under σ is

$$C_\sigma(S_{nk}) = \sum_{i=0}^{(n-1)} f(i) + \sum_{j=0}^k \left(f(m_j - 2) - f(m_j - 1) \right).$$

The idea is to begin with the cost of installing a complete graph on n nodes, then adjust the costs of each v_j . Now it remains to determine the values of m_j that minimize $C_\sigma(S_{nk})$.

Because f is a decreasing convex cost function, $f(n - 2) - f(n - 1) \leq f(m_k - 2) - f(m_k - 1)$ for all possible values of m_k . Thus, $m_k = n$, which means that v_k should be installed last.

As the cost of each $v_{k-(i-1)}$ is fixed, consider the cost of v_{k-i} for $i \in \{1, \dots, k-1\}$. Given that $m_{k-j} = n - j$ for $0 \leq j \leq i-1$, observe that

$$f((n - i) - 2) - f((n - i) - 1) \leq f(m_{k-i} - 2) - f(m_{k-i} - 1),$$

which supports the conclusion that $m_{k-i} = n - i$. Using strong induction on i , we show that $C_\sigma(S_{nk})$ is minimized when the last k nodes in σ are v_1, \dots, v_k . Equation 13 follows. \square

4 Experimental Heuristics

We implemented two greedy heuristics to estimate solutions to the NANIP-Blackstart problem.

- The **MaxNeighbors** heuristic repeatedly chooses the uninstalled node that has the largest number of installed neighbors. Since we are using decreasing convex cost functions, this heuristic repeatedly installs the cheapest node next.
- The **MaxPercent** heuristic repeatedly chooses the uninstalled node that has the highest percentage of installed neighbors. This heuristic is motivated by the observation that there may be local subnetworks that would benefit from being installed as a coherent unit.

The codes take a network and a blackstart node as input; the output is an ordering on the vertices and a list of coefficients a_0, a_1, \dots, a_n representing the cost $\sum_{i=0}^{n-1} a_i f(i)$.

The implementation uses a max-priority queue to store the nodes that can be installed; each node in the queue has a priority that is the number of neighboring nodes installed for **MaxNeighbors** or the percentage of neighboring nodes installed for **MaxPercent**. Initially only the blackstart node is in the queue. In each step the code installs the maximum priority node, updates the priorities of its neighbors that are already in the queue, and adds to the queue any neighbors that have not yet been enqueued.

4.1 Experimental Framework

We implemented these two heuristics in Python and tested them on a set of networks from the SuiteSparse Matrix Collection [3]. By way of contrast, previous heuristics described in [5] for the original NANIP problem are tested only on much smaller, randomly generated graphs.

Table 1 details a subset of our test suite.

Table 1. Description of 6 test networks

Name	n	Application area
LeGresley_2508	2508	Power flow analysis of an electrical grid
LeGresley_4908	4908	Power flow analysis of an electrical grid
Powersim	15838	Power simulation matrix
S20PI_n	1182	Power system model
S40PI_n	2182	Power system model
S80PI_n	4182	Power system model

For each network we tested three different blackstart nodes, chosen at random. We then ran the two heuristics on each network and blackstart node pair.

The results were written as a sum of installation costs. For example, in one run on the LeGresley_4908 network, **MaxNeighbors** had a cost of:

$$f(0) + 21f(1) + 2236f(2) + 2227f(3) + 212f(4) + 211f(5) \quad (14)$$

whereas **MaxPercent** had a cost of:

$$f(0) + 18f(1) + 725f(2) + 696f(3) + 637f(4) + 411f(5) + 14f(6) + 3f(7). \quad (15)$$

4.2 Results

Looking just at the equations returned by our heuristics suggests a trend where **MaxPercent** has a longer tail than **MaxNeighbors** in the sense that the equations include $f(k)$ terms for larger k . Because the cost function is a decreasing function, this might seem to suggest that **MaxPercent** should typically outperform **MaxNeighbors**.

However, when we substituted in decreasing convex functions such as $f(k) = 1/k$ or $f(k) = 1/\sqrt{k}$, we found that in all cases **MaxNeighbors** did at least as well as **MaxPercent**. One possibility is that the convexity of the cost function means the advantage of having $f(k)$ terms with larger k decreases as k increases. As a result, it is the terms with smaller k that have more of an effect.

That said, the differences seen were minimal as there were no instances where **MaxNeighbors** outperformed **MaxPercent** by more than 9%. The differences might be greater using other cost functions or on other networks with different characteristics.

5 Conclusion

In this paper we describe an extension to the model for the network installation problem described in [4]. We study a variant with blackstart nodes, which represent locations which can be self-started after a widespread power outage. We show the overall problem remains NP-hard, then prove exact results for certain subclasses of graphs. We also describe two heuristics and analyze their behaviors on a set of real-world networks.

Looking ahead, we are interested in studying what happens as the model better approximates real world power networks. For example, dividing a network into subsystems is often advantageous for restoration since the subsystems can be restored in parallel [1]. Introducing multiple blackstart nodes models exactly this situation. In addition, we are interested in analyzing the inverse question where the network is given and the goal is to identify the nodes at which k blackstart nodes should be located in order to minimize the restart cost.

Acknowledgements. We are grateful to Pomona College for helping fund this work.

References

1. Adibi, M.M., Fink, L.H.: Power system restoration planning. *IEEE Trans. Power Syst.* **9**(1), 22–28 (1994)
2. Atkins, K., Chen, J., Kumar, V.A., Marathe, A.: The structure of electrical networks: a graph theory-based analysis. *Int. J. Crit. Infrastruct.* **5**(3), 265–284 (2009)
3. Davis, T.A., Hu, Y.: The University of Florida Sparse Matrix Collection. *ACM Trans. Math. Softw.* **38**(1), 1:1–1:25 (2011). <https://doi.org/10.1145/2049662.2049663>, <http://doi.acm.org/10.1145/2049662.2049663>
4. Gutfraind, A., Bradonjić, M., Novikoff, T.: Modelling the neighbour aid phenomenon for installing costly complex networks. *J. Complex Netw.* (2014). <https://doi.org/10.1093/comnet/cnu033>
5. Gutfraind, A., Kun, J., Lelkes, Á.D., Reyzin, L.: Network installation and recovery: approximation lower bounds and faster exact formulations. *CoRR* abs/1411.3640 (2014). <http://arxiv.org/abs/1411.3640>
6. Ketabi, A., Feuillet, R.: Ant colony search algorithm for optimal generators startup during power system restoration. *Math. Probl. Eng.* **2010**, 1–11 (2010). <https://doi.org/10.1155/2010/906935>
7. Lin, Z.Z., Wen, F.S., Chung, C.Y., Wong, K.P., Zhou, H.: Division algorithm and interconnection strategy of restoration subsystems based on complex network theory. *IET Gener. Trans. Distrib.* **5**(6), 674–683 (2011). <https://doi.org/10.1049/iet-gtd.2010.0586>
8. Rokrok, E., Shafie-khah, M., Siano, P., Catalão, J.P.S.: A decentralized multi-agent-based approach for low voltage microgrid restoration. *Energies* **10**(10) (2017). <https://doi.org/10.3390/en10101491>, <http://www.mdpi.com/1996-1073/10/10/1491>
9. Van Hentenryck, P., Coffrin, C.: Transmission system repair and restoration. *Math. Programm.* **151**(1), 347–373 (2015). <https://doi.org/10.1007/s10107-015-0887-0>
10. Wikipedia: North American Electric Reliability Corporation. https://en.wikipedia.org/wiki/North_American_Electric_Reliability_Corporation. Accessed 27 July 2018



Mathematical Analysis of a Network's Asymptotic Behaviour Based on Its Strongly Connected Components

Jan Treur^(✉)

Behavioural Informatics Group, Vrije Universiteit Amsterdam, Amsterdam, The Netherlands
j.treur@vu.nl

Abstract. In this paper a general theorem is presented that relates asymptotic behaviour of a network to the network's characteristics concerning the network's strongly connected components and their mutual connections. The theorem generalises existing theorems for specific cases such as acyclic networks, fully and strongly connected networks, and theorems addressing only linear functions.

1 Introduction

This paper analyses the relation between network structure and emerging asymptotic diffusion behaviour of a network. In many cases this relation is only studied by performing simulation experiments. However, as shown in this paper, within a certain context it is also possible to analyse mathematically how certain asymptotic behaviours relate to certain properties of the network structure. In [12] this question was only addressed for two special cases: the case of an acyclic network, and the case of a strongly connected network; the general case remained open. The current paper develops a mathematical analysis for the general case. To achieve that, tools were adopted from the area of Graph Theory, in particular the manner to identify the connectivity structure within a graph by decomposition of the graph according to its (maximal) strongly connected components and the resulting (acyclic) condensation graph [6], Chap. 3, and the notion of stratification of an acyclic directed graph; e.g., [3].

In addition to the connectivity structure, the main theorem also takes into account the combination functions by which the impacts from multiple incoming connections are aggregated. It applies not to just one type (for example, linear or scaled sum functions), but to a whole class of functions: those that are characterised as being monotonic, scalar-free and normalised. This class does include the often used linear functions, but also, for example, n th order Euclidean combination functions involving squares or higher powers of values in the aggregation. Moreover, it explains which exactly are the relevant characteristics that make that these functions contribute to certain asymptotic behaviour. It will be shown how using the above mentioned tools from Graph Theory, together with the characteristics of combination functions mentioned, enable to address the general case and obtain a main theorem about it.

To apply this main theorem to a given network, first the decomposition of the network into its strongly connected components is determined. A variety of efficient

algorithms are available to determine these strongly connected components; for example, see [1, 4, 5, 8, 9, 13]. Next, the connections between these components are identified, as shown in an acyclic condensation graph, and a stratification of it. Based on this acyclic and stratified structure added to the original network, the main theorem will indicate whether and which states within the network will end up in a common equilibrium value, and determine bounds for the equilibrium values of the states.

In the paper, first in Sect. 2 the basic definition of network used as a vehicle is briefly discussed. Next, in Sect. 3 asymptotic behaviour is discussed, illustrated for an example network with an example simulation. In Sect. 4 the definitions of the Graph Theory tools on connectivity are discussed; in Sect. 5 the identified characteristics of combination functions are defined. In Sect. 6 the main theorem is formulated and it is shown how it was proved. Finally, Sect. 7 is a discussion.

2 Temporal-Causal Networks

A temporal-causal network model is based on three notions, connection weight, combination function, and speed factor, which define the network structure; see Table 1, upper part. Note that the word temporal in temporal-causal refers to the causality, not to the network. To provide sufficient flexibility, a number of standard combination functions are available as options, but also own-defined functions can be used. In Table 1, lower part it is shown how a conceptual representation describing a network structure defines a numerical representation of the network's dynamics; see also [10], Chap. 2, or [11]. Here X_1, \dots, X_k are the states with outgoing connections to state Y . This defines the detailed dynamic semantics of a temporal-causal network.

Table 1. Conceptual and numerical representations of a temporal-causal network

Concepts	Notation	Explanation
States and connections	$X, Y, X \rightarrow Y$	Describes the nodes and links of a network structure (e.g., in graphical or matrix format)
Connection weight	$\omega_{X,Y}$	Connection weight $\omega_{X,Y} \in [-1, 1]$ represents the strength of the impact of state X on state Y through connection $X \rightarrow Y$
Aggregating multiple impacts	$c_Y(\cdot)$	For each state Y a combination function $c_Y(\cdot)$ is chosen to combine the causal impacts of other states on state Y
Timing of the causal effect	η_Y	For each state Y a speed factor $\eta_Y \geq 0$ is used to represent how fast a state is changing upon causal impact
Concepts	Numerical representation	Explanation
State values over time t	$Y(t)$	At each time point t each state Y has a real number value in $[0, 1]$
Single causal impact	$\text{impact}_{X,Y}(t)$ $= \omega_{X,Y} X(t)$	At t state X with connection to state Y has an impact on Y , using weight $\omega_{X,Y}$
Aggregating multiple impacts	$\text{aggimpact}_Y(t)$ $= c_Y(\text{impact}_{X_1,Y}(t), \dots, \text{impact}_{X_k,Y}(t))$ $= c_Y(\omega_{X_1,Y} X_1(t), \dots, \omega_{X_k,Y} X_k(t))$	The aggregated impact of multiple states X_i on Y at t , is determined using combination function $c_Y(\cdot)$
Timing of the causal effect	$Y(t + \Delta t) = Y(t) + \eta_Y [\text{aggimpact}_Y(t) - Y(t)] \Delta t$ $= Y(t) + \eta_Y [c_Y(\omega_{X_1,Y} X_1(t), \dots, \omega_{X_k,Y} X_k(t)) - Y(t)] \Delta t$	The impact on Y is exerted over time gradually, using speed factor η_Y

The difference equations in the last row in Table 2 can be used for simulation and mathematical analysis, and can also be written in differential equation format: $\mathbf{d}Y(t)/\mathbf{dt} = \eta_Y[\mathbf{c}_Y(\omega_{X_1,Y}X_1(t), \dots, \omega_{X_k,Y}X_k(t)) - Y(t)]$. Note that combination functions are functions on the 0 – 1 interval within the real numbers: $[0, 1]^k \rightarrow [0, 1]$. Examples of combination functions often used (see also [10], Chap. 2, Table 2.10) are the *identity* **id**(.) for states with only one impact, the minimum function **min**(..), the *advanced logistic sum* combination function **alogistic** $_{\sigma,\tau}(.)$ with steepness σ and threshold τ , defined by

$$\text{alogistic}_{\sigma,\tau}(V_1, \dots, V_k) = [1/(1 + e^{-\sigma(V_1 + \dots + V_{k-\tau})}) - 1/(1 + e^{\sigma\tau})](1 + e^{-\sigma\tau}),$$

and the Euclidean combination function of n th order with scaling factor λ (generalising the scaled sum **ssum** $_{\lambda}(..)$ for $n = 1$) defined by

$$\text{eucl}_{n,\lambda}(V_1, \dots, V_k) = ((V_1^n + \dots + V_k^n)/\lambda)^{1/n}$$

3 Asymptotic Network Behaviour

The asymptotic behaviour will be explored by analysing the possible equilibria. First a definition of stationary points and equilibria.

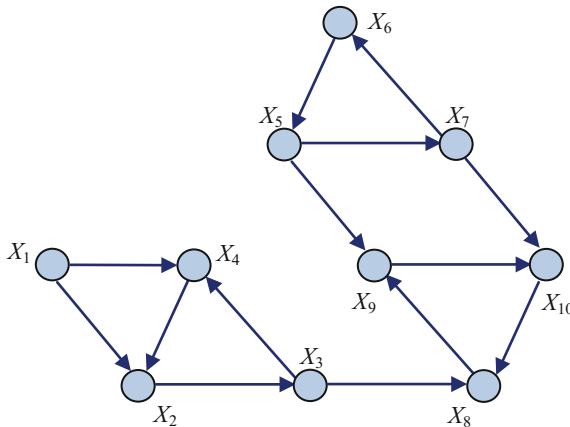
Definition 1 (stationary point and equilibrium). A state Y has a *stationary point* at t if $\mathbf{d}Y(t)/\mathbf{dt} = 0$. The network is in *equilibrium* at t if every state Y of the model has a stationary point at t .

Considering the differential equation format for a temporal-causal network model, and assuming a nonzero speed factor a more specific criterion can be found.

Lemma 1 (Criterion for a stationary point in a temporal-causal network). Let Y be a state and X_1, \dots, X_k the states with outgoing connections to state Y . Then Y has a stationary point at t if and only if $\mathbf{c}_Y(\omega_{X_1,Y}X_1(t), \dots, \omega_{X_k,Y}X_k(t)) = Y(t)$.

The example network shown in Fig. 1 is used as illustration. For the connection weights shown in Table 2 the simulation outcome (for $\Delta t = 0.5$) shown in Fig. 2 was obtained.

In this simulation state X_1 has initial value 0.9 and this stays constant due to having speed factor 0. The other states have initial value 0, except X_5 which has 0.9 as initial value. The speed factor of states X_2 to X_{10} is 0.5, and their combination function is a normalised scaled sum function. The simulation outcome can be seen in Fig. 2. It turns out that states X_1 to X_4 all end up at value 0.9, states X_5 to X_7 all at value 0.3 and states X_8 to X_{10} at different individual values 0.681, 0.490, and 0.389, respectively. So, there is some clustering, but also some states end up in their own unique value, and it can be observed that these unique values are in between the cluster values.

**Fig. 1.** Example network**Table 2.** Connection weights for the example simulation

	X ₁	X ₂	X ₃	X ₄	X ₅	X ₆	X ₇	X ₈	X ₉	X ₁₀
X ₁	0.8		0.5							
X ₂		1								
X ₃			0.2				0.8			
X ₄	0.6									
X ₅						0.6		0.8		
X ₆				0.7						
X ₇					0.8				0.8	
X ₈								0.8		
X ₉									0.7	
X ₁₀						0.6				

How can such an emerging pattern be explained? This question will be addressed in the next sections. It will be found out how the pattern depends on the network's characteristics, and in particular on the connectivity within the network and the characteristics of the combination functions. Each of these factors will be discussed first in different sections, after which it will be analysed how they relate to the emerging pattern.

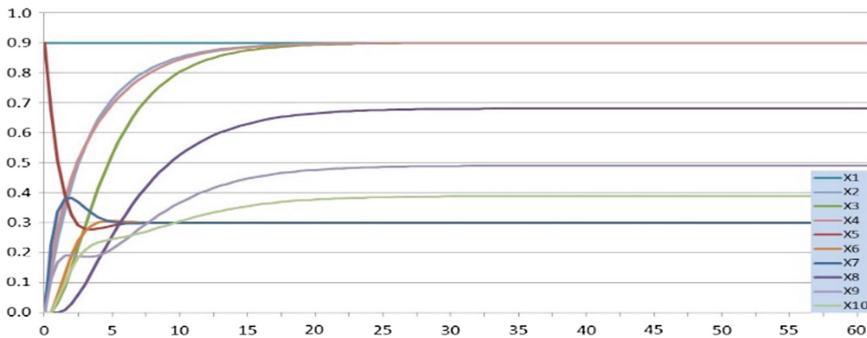


Fig. 2. Example simulation

4 Connectivity and Strongly Connected Components

To analyse connectivity within Graph Theory the notion of strongly connected component has been identified. The main parts of the following definitions can be found, for example, in [6], Chap. 3, or [7], Sect. 6. Note that in the current paper only nonnegative connection weights are considered.

Definition 2 (reachability and strongly connected components).

- State Y is *forward reachable* from state X if there is a directed path from X to Y with nonzero connection weights and speed factors.
- A network N is *strongly connected* if every two states are mutually forward reachable within N .
- A state is called *independent* if it is not forward reachable by any other state.
- A *subnetwork* of a network N is a network whose states and connections are states and connections of N .
- A *strongly connected component* C of a network N is a strongly connected subnetwork of N such that no larger strongly connected subnetwork of N contains it as a subnetwork.

Strongly connected components C can be identified by choosing any node X of N and adding all nodes that are on any cycle through X . Note also that when a node X is not on any cycle, then it will form a singleton strongly connected component C by itself; this applies in particular to all nodes of N with indegree or outdegree zero. There are efficient algorithms available to determine the strongly connected components of a network or graph; for example, see [1, 4, 5, 8, 9, 13]. The strongly connected components of the example network shown in Fig. 1 are depicted in Fig. 3.

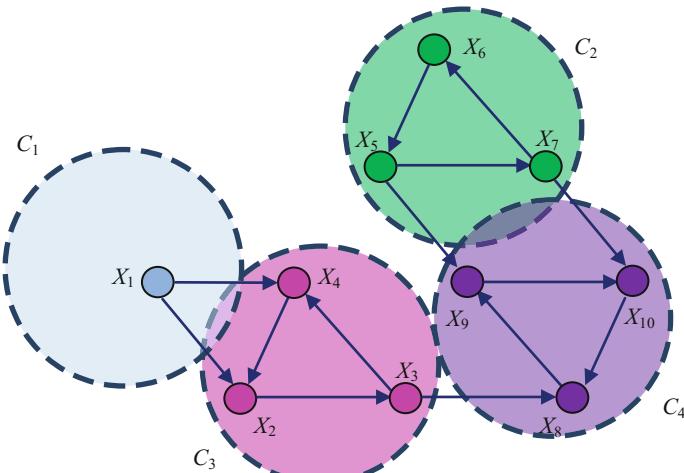


Fig. 3. The strongly connected components within the example network

Having identified the strongly connected components, allows to obtain a kind of abstracted picture of the network, called the condensation graph; see Fig. 4.

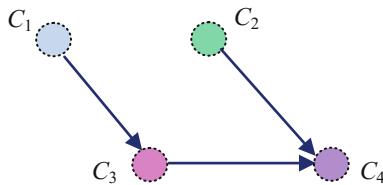


Fig. 4. Condensation of the example network by its strongly connected components: the directed acyclic condensation graph $C(N)$

Definition 3 (condensation graph). The *condensation* $C(N)$ of a network N with respect to its strongly connected components is a graph whose nodes are the strongly connected components of N and whose connections are determined as follows: there is a connection from node C_i to node C_j in $C(N)$ if and only if in N there is at least one connection from a node in the strongly connected component C_i to a node in the strongly connected component C_j .

An important result is that a condensation graph $C(N)$ is always an acyclic graph. The following theorem summarizes this; see also [6], Chap. 3, Theorems 3.6 and 3.8, or [7], Sect. 6.

Theorem 1 (acyclic condensation graph).

- (a) For any network N its condensation graph $C(N)$ is acyclic, and has at least one state of outdegree zero and at least one state of indegree zero.

- (b) The network N is acyclic itself if and only if it is graph-isomorphic to $C(N)$. In this case the nodes in $C(N)$ all are singleton sets $\{X\}$ containing one state X from N .
- (c) The network N is strongly connected itself if and only if $C(N)$ only has one node; this node is the set of all states of N .

As a next step, for any acyclic directed graph a stratification structure is defined; for example, see [3]. Here a similar construction is applied in particular to the condensation graph $C(N)$ thus obtaining a stratified condensation graph $SC(N)$; see Fig. 5.

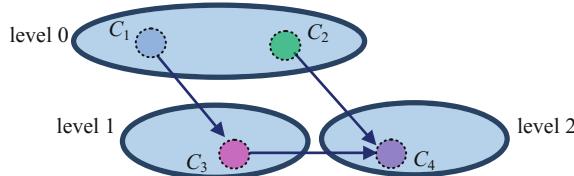


Fig. 5. Stratified condensation graph $SC(N)$ for the example network

Definition 4 (stratified condensation graph). The *stratified condensation graph* for network N , denoted by $SC(N)$, is the condensation graph $C(N)$ together with a leveled partition S_0, \dots, S_{h-1} in strata S_i such that $S_0 \cup \dots \cup S_{h-1}$ is the set of all nodes of $C(N)$ and the S_i are mutually disjoint, which is defined inductively as follows. Here h is the height of $C(N)$, i.e., the length of the longest path in $C(N)$.

- (i) The stratum S_0 is the set of nodes in $C(N)$ without incoming connections in $C(N)$
- (ii) For each $i > 0$ the stratum S_i is the set of nodes in $C(N)$ for which all incoming connections in $C(N)$ come only from nodes in S_0, \dots, S_{i-1} .

If node X is in stratum S_i , its *level* is i .

5 Characteristics of Combination Functions

It has been found out (see Sect. 6) how the following characteristics of combination functions relate to asymptotic behaviour as discussed in Sect. 3. Note that for combination functions it is (silently) assumed that $c(V_1, \dots, V_k) = 0$ iff $V_i = 0$ for all i .

Definition 5 (monotonic, scalar-free, and additive for a combination function).

- (a) A function $c(\cdot)$ is called *monotonically increasing* if for all values U_i, V_i it holds

$$U_i \leq V_i \text{ for all } i \Rightarrow c(U_1, \dots, U_k) \leq c(V_1, \dots, V_k)$$

- (b) A function $c(\cdot)$ is called *strictly monotonically increasing* if

$$U_i \leq V_i \text{ for all } i, \text{ and } U_j < V_j \text{ for at least one } j \Rightarrow c(U_1, \dots, U_k) < c(V_1, \dots, V_k)$$

- (c) A function $c(\cdot)$ is called *scalar-free* if for all $\alpha > 0$ and all V_1, \dots, V_k it holds

$$\mathbf{c}(\alpha V_1, \dots, \alpha V_k) = \alpha \mathbf{c}(V_1, \dots, V_k)$$

(d) A function $\mathbf{c}(\cdot)$ is called *additive* if for all U_1, \dots, U_k and V_1, \dots, V_k it holds

$$\mathbf{c}(U_1 + V_1, \dots, U_k + V_k) = \mathbf{c}(U_1, \dots, U_k) + \mathbf{c}(V_1, \dots, V_k)$$

(e) A function $\mathbf{c}(\cdot)$ is called *linear* if it is both scalar-free and additive.

Note that (n th order) Euclidean combination functions satisfy (a), (b), and (c), and the first order ones (= scaled sum functions) satisfy also (d) and (e).

Definition 6 (normalised). A network is *normalised* if for each state Y it holds $\mathbf{c}_Y(\omega_{X1,Y}, \dots, \omega_{Xk,Y}) = 1$, where X_1, \dots, X_k are the states with outgoing connections to Y .

Note that for a Euclidean combination function of n th order the scaling parameter choice $\lambda_Y = \omega_{X1,Y}^n + \dots + \omega_{Xk,Y}^n$ will provide a normalised network. This can be done in general:

(1) normalising a combination function

If any combination function $\mathbf{c}_Y(\cdot)$ is replaced by $\mathbf{c}'_Y(\cdot)$ defined as

$$\mathbf{c}'_Y(V_1, \dots, V_k) = \mathbf{c}_Y(V_1, \dots, V_k) / \mathbf{c}_Y(\omega_{X1,Y}, \dots, \omega_{Xk,Y})$$

(note $\mathbf{c}_Y(\omega_{X1,Y}, \dots, \omega_{Xk,Y}) > 0$ since $\omega_{Xi,Y} > 0$), then the network becomes normalised.

(2) normalising the connection weights (for scalar-free combination functions)

For scalar-free combination functions also normalisation is possible by adapting the connection weights; define $\omega'_{Xi,Y} = \omega_{Xi,Y} / \mathbf{c}_Y(\omega_{X1,Y}, \dots, \omega_{Xk,Y})$, then indeed it holds:

$$\mathbf{c}_Y(\omega'_{X1,Y}, \dots, \omega'_{Xk,Y}) = \mathbf{c}(\omega_{X1,Y} / \mathbf{c}_Y(\omega_{X1,Y}, \dots, \omega_{Xk,Y}), \dots, \omega_{Xk,Y} / \mathbf{c}_Y(\omega_{X1,Y}, \dots, \omega_{Xk,Y})) = 1$$

The following proposition illustrates some of the implications of the above-defined characteristics. This will be used in Sect. 6.

Proposition 1. Suppose the network is normalised.

- (a) If the combination functions are scalar-free and X_1, \dots, X_k are the states with outgoing connections to Y , and $X_1(t) = \dots = X_k(t) = V$ for some common value V , then also $\mathbf{c}_Y(\omega_{X1,Y} X_1(t), \dots, \omega_{Xk,Y} X_k(t)) = V$.
- (b) If the combination functions are scalar-free and X_1, \dots, X_k are the states with outgoing connections to Y , and for $U_1, \dots, U_k, V_1, \dots, V_k$ and $\alpha \geq 0$ it holds $V_i = \alpha U_i$, then $\mathbf{c}_Y(\omega_{X1,Y} V_1, \dots, \omega_{Xk,Y} V_k) = \alpha \mathbf{c}_Y(\omega_{X1,Y} U_1, \dots, \omega_{Xk,Y} U_k)$

If in this situation in two different simulations, state values $X_i(t)$ and $X'_i(t)$ are generated then $X'_i(t) = \alpha X_i(t) \Rightarrow X'_i(t + \Delta t) = \alpha X_i(t + \Delta t)$.

- (c) If the combination functions are additive and X_1, \dots, X_k are the states with outgoing connections to Y , then for values $U_1, \dots, U_k, V_1, \dots, V_k$ it holds

$$\begin{aligned} \mathbf{c}_Y(\omega_{X1,Y}(U_1 + V_1), \dots, \omega_{Xk,Y}(U_k + V_k)) &= \mathbf{c}_Y(\omega_{X1,Y}U_1, \dots, \omega_{Xk,Y}U_k) \\ &\quad + \mathbf{c}_Y(\omega_{X1,Y}V_1, \dots, \omega_{Xk,Y}V_k) \end{aligned}$$

If in this situation in three different simulations, state values $X_i(t)$, $X'_i(t)$ and $X''_i(t)$ are generated then

$$X''_i(t) = X_i(t) + X'_i(t) \Rightarrow X''_i(t + \Delta t) = X_i(t + \Delta t) + X'_i(t + \Delta t)$$

- (d) If the combination functions are scalar-free and monotonically increasing, and X_1, \dots, X_k are the states with outgoing connections to Y , and $V_1 \leq X_1(t), \dots, X_k(t) \leq V_2$ for some values V_1 and V_2 , then also

$$V_1 \leq \mathbf{c}_Y(\omega_{X1,Y}X_1(t), \dots, \omega_{Xk,Y}X_k(t)) \leq V_2$$

and if $\eta_Y \Delta t \leq 1$ and $V_1 \leq Y(t) \leq V_2$ then $V_1 \leq Y(t + \Delta t) \leq V_2$.

Proof. (a) This follows from

$$\begin{aligned} \mathbf{c}_Y(\omega_{X1,Y}X_1(t), \dots, \omega_{Xk,Y}X_k(t)) &= \mathbf{c}_Y(\omega_{X1,Y}V, \dots, \omega_{Xk,Y}V) \\ &= V \mathbf{c}_Y(\omega_{X1,Y}, \dots, \omega_{Xk,Y}) = V \end{aligned}$$

(b) and (c) can be verified easily.

(d) This follows from

$$\begin{aligned} V_1 &= V_1 \mathbf{c}_Y(\omega_{X1,Y}, \dots, \omega_{Xk,Y}) = \mathbf{c}_Y(\omega_{X1,Y}V_1, \dots, \omega_{Xk,Y}V_1) \\ &\leq \mathbf{c}_Y(\omega_{X1,Y}X_1(t), \dots, \omega_{Xk,Y}X_k(t)) \leq \mathbf{c}_Y(\omega_{X1,Y}V_2, \dots, \omega_{Xk,Y}V_2) \\ &= V_2 \mathbf{c}_Y(\omega_{X1,Y}, \dots, \omega_{Xk,Y}) = V_2 \end{aligned}$$

and the second part from

$$\begin{aligned} Y(t) + \eta_Y [\mathbf{c}_Y(\omega_{X1,Y}X_1(t), \dots, \omega_{Xk,Y}X_k(t)) - Y(t)] \Delta t \\ = \mathbf{c}_Y(\omega_{X1,Y}X_1(t), \dots, \omega_{Xk,Y}X_k(t)) \eta_Y \Delta t + Y(t)(1 - \eta_Y \Delta t) \leq V_2 \eta_Y \Delta t + V_2(1 - \eta_Y \Delta t) = V_2 \end{aligned}$$

and similarly for V_1

$$\mathbf{c}_Y(\omega_{X1,Y}X_1(t), \dots, \omega_{Xk,Y}X_k(t)) \eta_Y \Delta t + Y(t)(1 - \eta_Y \Delta t) \geq V_1 \eta_Y \Delta t + V_1(1 - \eta_Y \Delta t) = V_1$$

6 Asymptotic Network Behaviour and Network Characteristics

In this section it is shown how the network structure characteristics concerning connectivity and combination function characteristics as discussed in Sects. 4 and 5 relate to emerging network behaviour. As a first case, consider a network without cycles, for example, a hierarchical population following a leader. Then the following theorem has been proven using Lemma 1 from Sect. 3 and Proposition 1; see [12].

Theorem 2 (common state values provide equilibria). Suppose a network with nonnegative connections is based on normalised and scalar-free combination functions. Then the following hold.

- (a) Whenever all states have the same value V , the network is in an equilibrium state.
- (b) If for every state for its initial value V it holds $V_1 \leq V \leq V_2$, then for all t for every state Y it holds $V_1 \leq Y(t) \leq V_2$. In an achieved equilibrium for every state for its equilibrium value V it holds $V_1 \leq V \leq V_2$.

Theorem 3 (Common equilibrium state values; acyclic case). Suppose an acyclic network with nonnegative connections is based on normalised and scalar-free combination functions.

- (a) If in an equilibrium state the independent states all have the same value V , then all states have the same value V .
- (b) If, moreover, the combination functions are monotonically increasing, and in an equilibrium state the independent states all have values V with $V_1 \leq V \leq V_2$, then all states have values V with $V_1 \leq V \leq V_2$.

Next, a basic lemma for dynamics of normalised networks with combination functions that are (strictly) monotonically increasing and scalar-free; see [12].

Lemma 2. Let a normalised network with nonnegative connections be given and its combination functions are monotonically increasing and scalar-free; then the following hold:

- (a) (i) If for some node Y at time t for all nodes X with $\omega_{X,Y} > 0$ it holds $X(t) \leq Y(t)$, then $Y(t)$ is decreasing at t : $dY(t)/dt \leq 0$.
(ii) If the combination functions are strictly increasing and a node X exists with $X(t) < Y(t)$ and $\omega_{X,Y} > 0$, and the speed factor of Y is nonzero, then $Y(t)$ is strictly decreasing at t : $dY(t)/dt < 0$.
- (b) (i) If for some node Y at time t for all nodes X with $\omega_{X,Y} > 0$ it holds $X(t) \geq Y(t)$, then $Y(t)$ is increasing at t : $dY(t)/dt \geq 0$.
(ii) If, the combination function is strictly increasing and a node X exists with $X(t) > Y(t)$ and $\omega_{X,Y} > 0$, and the speed factor of Y is nonzero, then $Y(t)$ is strictly increasing at t : $dY(t)/dt > 0$.

Using Lemmas 1 and 2 the following proposition has been proven for strongly connected networks with cycles; see [12].

Theorem 4 (Common equilibrium state values; strongly connected cyclic case). Suppose the combination functions of the normalised network N are scalar-free and strictly monotonically increasing. Then the following hold.

- (a) If the network is strongly connected itself, then in an equilibrium state all states have the same value.
- (b) Suppose the network has one or more independent states and the subnetwork without these independent states is strongly connected. If in an equilibrium state all independent states have values V with $V_1 \leq V \leq V_2$, then all states have values V with $V_1 \leq V \leq V_2$. In particular, when all independent states have the same value V , then all states have this same value V .

Next, the main, general theorem is formulated, consisting of two Theorems 5 and 6.

Theorem 5 (main theorem on equilibrium state values, part I). Suppose the network N is normalised and its combination functions are scalar-free and strictly monotonic. Let $\text{SC}(N)$ be the stratified condensation graph of N . Then in an equilibrium state of N the following hold.

- (a) For each strongly connected component $C \in \text{SC}(N)$ of N of level 0 the following hold:
 - (i) All states in N belonging to C have the same equilibrium value V .
 - (ii) If for the initial values V of all states in N belonging to C it holds $V_1 \leq V \leq V_2$, then also for the equilibrium values V of all states in C it holds $V_1 \leq V \leq V_2$.
 - (iii) In particular, when all initial values of states in N belonging to C are equal to one value V , then the equilibrium value of all states in C is also V .
- (b) Let $C \in \text{SC}(N)$ be a strongly connected component of N of level $i > 0$. Let $C_1, \dots, C_k \in \text{SC}(N)$ be the strongly connected components of N with an outgoing connection to C within the condensation graph $\text{SC}(N)$. Then the following hold:
 - (i) If for the equilibrium values V of all states in N belonging to $C_1 \cup \dots \cup C_k$ it holds $V_1 \leq V \leq V_2$, then for all states in N belonging to C for their equilibrium value V it holds $V_1 \leq V \leq V_2$.
 - (ii) In particular, when all equilibrium values of all states in N belonging to $C_1 \cup \dots \cup C_k$ are equal to one value V , then also the equilibrium values of all states in N belonging to C are equal to the same V .

Proof.

- (a) (i) This follows from Theorem 3(a).
- (ii) This follows from Proposition 1(b).
- (iii) This follows from (ii) with $V_1 = V_2 = V$.

- (b) (i) This follows from Theorem 3(b) applied to C augmented with (as independent states) the states in $C_1 \cup \dots \cup C_k$ with outgoing connections to states in C , with their values and these connections.
(ii) This follows from (i) with $V_1 = V_2 = V$.

Theorem 6 (main theorem on equilibrium state values, part II). Suppose the network N is normalised and its combination functions are scalar-free and strictly monotonic. Let $\text{SC}(N)$ be the stratified condensation graph of N . Then in an equilibrium state of N the following hold.

- (a) If the equilibrium values of all states in every strongly connected component of level 0 in $\text{SC}(N)$ are equal to one value V , then the equilibrium state values of all states in N are equal to the same value V .
- (b) If for the equilibrium values V of all states in every strongly connected component of level 0 in $\text{SC}(N)$ it holds $V_1 \leq V \leq V_2$, then for the equilibrium state values V of all states in N it holds $V_1 \leq V \leq V_2$.
- (c) If the initial values of all states in every strongly connected component of level 0 in $\text{SC}(N)$ are equal to one value V , then for the equilibrium state values of all states in N are equal to the same value V .
- (d) If for the initial values V of all states in every strongly connected component of level 0 in $\text{SC}(N)$ it holds $V_1 \leq V \leq V_2$, then for the equilibrium state values V of all states in N it holds $V_1 \leq V \leq V_2$.

Proof. Use induction over the number of strata in $\text{SC}(N)$ and apply Theorem 4(a) for the level 0 stratum and Theorem 4(b) for the induction step from the strata of level $j < i$ to the stratum of level $i > 0$.

These theorems are in accordance with the example simulation shown in Fig. 2 and other simulations that were conducted. As an illustration, the strongly connected components of level 0 for the example are the subnetworks based on $\{X_1\}$ and $\{X_5, X_6, X_7\}$ (see Figs. 3 and 5). The initial values of X_1 and X_5 are 0.9, and all other initial values are 0. From Theorems 5(a)(i) and 4(a)(ii), it follows that the equilibrium value of X_1 is 0.9, which indeed is the case, and those of X_5, X_6, X_7 are the same and ≤ 0.9 ; this is indeed the case in Fig. 2, as these three equilibrium values of X_5, X_6, X_7 are all 0.3. This specific value 0.3 in principle depends on the initial values of the states and the connection weights, which are not taken into account in the theorems. However, see also Theorem 7.

Next, consider the level 1 component C_3 , based on $\{X_2, X_3, X_4\}$. The only incoming connection here is from X_1 , which has equilibrium value 0.9 (implied by Theorem 5(a)(ii)). Now by Theorem 5(b)(ii) it follows that all of X_2, X_3, X_4 also have the same equilibrium value 0.9; this is indeed the case in Fig. 2. Finally, consider level 2 component C_4 , based on $\{X_8, X_9, X_{10}\}$. It has two incoming connections, one from X_3 in and one from X_5 in C_2 . Now their equilibrium values are not equal: they are 0.9 and 0.3, respectively. Therefore it is not implied by the above theorems that the equilibrium values of X_8, X_9, X_{10} are the same; and indeed in Fig. 2 they are different: 0.681, 0.490, and 0.389, respectively. Yet an implication from Theorem 5(b)(i) is that these equilibrium values should be ≥ 0.3 and ≤ 0.9 . This is indeed the case in Fig. 2. This

illustrates how these theorems can be applied. Note that the specific equilibrium values 0.681, 0.490, and 0.389 cannot be predicted in this way. They also depend on the connection weights for the states X_8, X_9, X_{10} within component C_4 , and these are not taken into account in the general theorem. However see also below, in the last paragraph of this section.

As a variation, if the initial value of X_1 is set at 0.3 instead of 0.9, then all equilibrium values turn out to become the same 0.3; see Fig. 6. In this case the values of all states in the level 0 components C_1 and C_2 have the same value 0.3. Similar to the first case above, also the states in C_3 have the equilibrium value 0.3 because they are only affected by X_1 which has value 0.3. But now the equilibrium values of both X_3 in C_3 and X_5 in C_2 are the same 0.3, so Theorem 5(b)(ii) can be applied to derive that all states in C_4 also have that same equilibrium value 0.3.

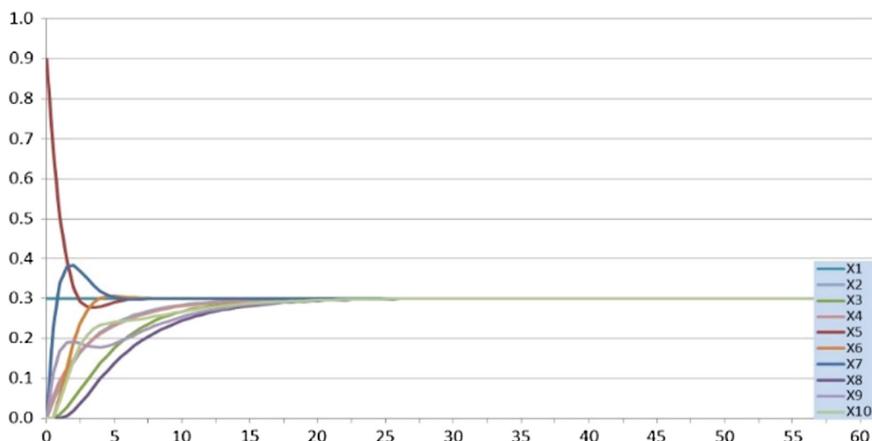


Fig. 6. Variation of the example simulation for initial value 0.3 of X_1

This proves that all states of the network have to have value 0.3 in the equilibrium. Alternatively, just apply Theorem 6(a) for this case. Then from the equal equilibrium values in the level 0 components C_1 and C_2 it immediately follows that all states in the network have that same equilibrium value.

In the main theorem the level 0 components play an important role, as initial nodes in the stratified condensation graph $SC(N)$. Therefore it can be useful to know more about their state values in an equilibrium. A result on this is the following.

Theorem 7 (equilibrium state values for level 0 components). Suppose the network N with states X_1, \dots, X_n is normalised and strongly connected. Then the following hold.

- (a) If the combination functions of the network N are scalar-free, then for given connection weights and speed factors, for any value $V \in [0, 1]$ there are initial values such that V is the common state value in an equilibrium achieved from these initial values.

- (b) For given connection weights and speed factors, let $\text{eq}: [0, 1]^n \rightarrow [0, 1]$ be the function such that $\text{eq}(V_1, \dots, V_n)$ is the common state value for an equilibrium achieved from initial values $X_i(0) = V_i$ for all i . Then $\text{eq}(0, \dots, 0) = 0$, $\text{eq}(1, \dots, 1) = 1$, and the following hold:
 - (i) If the combination functions of the network are scalar-free, then eq is scalar-free
 - (ii) If the combination functions of the network are additive, then eq is additive
- (c) Suppose the combination functions of the network N are linear. For given connection weights and speed factors for each i let e_i be the achieved common equilibrium value for initial values $X_i(0) = 1$ and $X_j(0) = 0$ for all $j \neq i$, i.e., $e_i = \text{eq}(0, \dots, 0, 1, 0, \dots, 0)$ with 1 as i th argument. Then the sum of the e_i is 1, i.e., $e_1 + \dots + e_n = 1$ and in the general case for these given connection weights and speed factors, the common equilibrium value $\text{eq}(\dots)$ is a linear, monotonically increasing, continuous and differentiable function of the initial values V_1, \dots, V_n satisfying the following linear relation: $\text{eq}(V_1, \dots, V_n) = e_1 V_1 + \dots + e_n V_n$. If the combination functions of N are strictly increasing, then $e_i > 0$ for all i , and eq is also strictly increasing.

Proof. (a) This follows from Proposition 1(a) or (d) with $V_1 = V_2 = V$.

(b and c) This follows from Proposition 1(b) and (c), and Lemma 2.

Based on this theorem, in particular for the case of linear combination functions, for level 0 components after each base equilibrium value e_i is determined, any equilibrium value can be predicted from the initial values by the identified linear expression.

Note that in particular for the case of linear combination functions the equilibrium equations are linear and could be solved algebraically. However, this does not provide additional information for level 0 components. They have an infinite number of solutions as every common value V is a solution; apparently the linear equations always have a mutual dependency in this case. But for components of level $i > 0$ solving the linear equations can provide specific values, due to the specific input values they get from one or more lower level components. In this way the specific equilibrium values of the states X_8, X_9 and X_{10} in C_4 can be determined algebraically from the values of the states X_3, X_5 , and X_7 in the lower level components C_2 and C_3 (repetitive digits in italics):

$$X_8 = 0.680952380952381, X_9 = 0.4904761904761905, X_{10} = 0.3888888888888889$$

which indeed is in accordance with the values found in the simulation.

Finally a similar theorem that is applicable for components of level $i > 0$.

Theorem 8 (equilibrium state values for components of level $i > 0$). Suppose the network is normalised, and consists of a strongly connected component plus a number of independent states A_1, \dots, A_p with outgoing connections to this strongly connected component. Then the following hold

- (a) Suppose the combination functions are scalar-free and X_1, \dots, X_k are the states with outgoing connections to Y . If for $U_1, \dots, U_k, V_1, \dots, V_k$ and $\alpha \geq 0$ it holds $V_i = \alpha U_i$ for all i , then $\mathbf{c}_Y(\omega_{X_1,Y}V_1, \dots, \omega_{X_k,Y}V_k) = \alpha\mathbf{c}_Y(\omega_{X_1,Y}U_1, \dots, \omega_{X_k,Y}U_k)$
- (b) Suppose the combination functions are additive and X_1, \dots, X_k are the states with outgoing connections to Y . Then if for values $U_1, \dots, U_k, V_1, \dots, V_k, W_1, \dots, W_k$ it holds $W_i = U_i + V_i$ for all i , then $\mathbf{c}_Y(\omega_{X_1,Y}W_1, \dots, \omega_{X_k,Y}W_k) = \mathbf{c}_Y(\omega_{X_1,Y}U_1, \dots, \omega_{X_k,Y}U_k) + \mathbf{c}_Y(\omega_{X_1,Y}V_1, \dots, \omega_{X_k,Y}V_k)$
- (c) Suppose all combination functions of the network N are linear. Then for given connection weights and speed factors, for each state Y the achieved equilibrium value for Y only depends on the equilibrium values V_1, \dots, V_p of states A_1, \dots, A_p ; the function $\text{eq}_Y(V_1, \dots, V_p)$ denotes this achieved equilibrium value for Y .
- (d) Suppose the combination functions of the network N are linear. For the given connection weights and speed factors for each i let $d_{i,Y}$ be the achieved equilibrium value for state Y in a situation with equilibrium values $A_i = 1$ and $A_j = 0$ for all $j \neq i$, i.e., $d_{i,Y} = \text{eq}_Y(0, \dots, 0, 1, 0, \dots, 0)$ with 1 as i th argument. Then in the general case for these given connection weights and speed factors, for each Y in the strongly connected component its equilibrium value is a linear, monotonically increasing, continuous and differentiable function $\text{eq}_Y(\dots)$ of the equilibrium values V_1, \dots, V_p of A_1, \dots, A_p satisfying the following linear relation: $\text{eq}_Y(V_1, \dots, V_p) = d_{1,Y}V_1 + \dots + d_{p,Y}V_p$. Here the sum of the $d_{i,Y}$ is 1: $d_{1,Y} + \dots + d_{p,Y} = 1$. In particular, the equilibrium values are independent of the initial values for all states Y different from A_1, \dots, A_p . If the combination functions of N are strictly increasing, then $d_{i,Y} > 0$ for all i , and $\text{eq}_Y(\dots)$ is also strictly increasing.

Proof. (a and b) follow from Proposition 1

(c) From (a) and (b) it follows that the equilibrium value of Y is a linear function of the initial values of all states of N . Therefore the function is a linear combination of $e_i = \text{eq}_Y(0, \dots, 0, 1, 0, \dots, 0)$ where only one state has initial value 1 and all other 0. However, when all independent states have (constant) value 0, from Theorem 5(b)(ii) it follows that all states will have equilibrium value 0. In particular, this holds for cases that only one of the states that are not independent have initial value 1 and all other states have initial value 0. This shows that from the linear combination the coefficient e_i of these terms are 0. Therefore $\text{eq}_Y(\dots)$ is a function of V_1, \dots, V_p only. From (a) and (b) it follows that $\text{eq}_Y(V_1, \dots, V_p)$ is linear, as indicated above. Therefore

$$\begin{aligned}\text{eq}_Y(V_1, \dots, V_p) &= \text{eq}_Y(V_1, 0, \dots, 0) + \dots + \text{eq}_Y(0, \dots, 0, V_i, 0, \dots, 0) \\ &\quad + \dots + \text{eq}_Y(0, \dots, V_p) \\ &= \text{eq}_Y(1, 0, \dots, 0)V_1 + \dots + \text{eq}_Y(0, \dots, 0, 1, 0, \dots, 0)V_i \\ &\quad + \dots + \text{eq}_Y(0, \dots, 1)V_p \\ &= d_{1,Y}V_1 + \dots + d_{i,Y}V_i + \dots + d_{p,Y}V_p\end{aligned}$$

This theorem can be applied by only determining the effect of the independent states on the equilibrium values; this is easily applicable especially in cases that there are just a

few of them.¹ Note that by using in the above proof Theorem 3 instead of Theorem 5(b) (ii), a similar theorem is obtained for the case of an acyclic network: then the equilibrium values of all states are linear combinations of the values of the initial states.

7 Discussion

To analyse and predict what asymptotic behaviour a given network will show is in general a challenging issue. In this paper a general theorem was presented that relates asymptotic network behaviour in terms of an equilibrium state to the network characteristics: in two parts described by Theorems 5 and 6 in Sect. 6. The relevant network characteristics concern on the one hand connectivity in terms of the network's strongly connected components and their mutual connections as shown in the network's condensation graph, and on the other hand characteristics of the combination functions used to aggregate the effects of multiple incoming connections (in particular, monotonicity, scalar freeness and normalisation). The theorem subsumes and generalises existing theorems for specific cases such as similar theorems for acyclic networks, fully connected networks and strongly connected networks (e.g., Theorems 3 and 4 in Sect. 6), and theorems addressing only scaled sum combination functions as one fixed type of combination function (e.g., Theorem 3 at p. 120 of [2]).

The main theorem can be applied to predict behaviour of a given network, or to set initial values in order to get some expected behaviour. It also can be used as a form of verification to check correctness of the implementation of a network. If simulation outcomes contradict the theorem, then this suggests that some debugging of the implementation may be needed.

References

1. Bloem, R., Gabow, H.N., Somenzi, F.: An algorithm for strongly connected component analysis in $n \log n$ symbolic steps. *Form. Meth. Syst. Des.* **28**, 37–56 (2006)
2. Bosse, T., Duell, R., Memon, Z.A., Treur, J., van der Wal, C.N.: Agent-based modelling of emotion contagion in groups. *Cogn. Comput.* **7**(1), 111–136 (2015)
3. Chen, Y.: General spanning trees and reachability query evaluation. In: Desai, B.C. (ed.) *Proceedings of the 2nd Canadian Conference on Computer Science and Software Engineering*, C3S2E'09, pp. 243–252. ACM Press (2009)
4. Fleischer, L.K., Hendrickson, B., Pinar, A.: On identifying strongly connected components in parallel. In: Rolim J. (ed) *Parallel and Distributed Processing. IPDPS 2000. Lecture Notes in Computer Science*, vol. 1800, pp. 505–511. Springer (2000)
5. Gentilini, R., Piazza, C., Policriti, A.: Computing strongly connected components in a linear number of symbolic steps. In: *Proceedings of the SODA'03*, pp. 573–582 (2003)
6. Harary, F., Norman, R.Z., Cartwright, D.: *Structural models: an introduction to the theory of directed graphs*. Wiley, New York (1965)
7. Kuich, W.: On the entropy of context-free languages. *Inf. Control* **16**, 173–200 (1970)

¹ At <http://www.few.vu.nl/~treur/linearsolvingv04.pdf> further analysis of the example network illustrates Theorem 8, and also for another example network model.

8. Li, G., Zhu, Z., Cong, Z., Yang, F.: Efficient decomposition of strongly connected components on GPUs. *J. Syst. Architect.* **60**(1), 1–10 (2014)
9. Tarjan, R.: Depth-first search and linear graph algorithms. *SIAM J. Comput.* **1**(2), 146–160 (1972)
10. Treur, J.: Network-Oriented Modeling: Addressing the Complexity of Cognitive, Affective and Social Interactions. Springer Publishers (2016)
11. Treur, J.: The Ins and Outs of Network-Oriented Modeling: from Biological Networks and Mental Networks to Social Networks and Beyond. Paper for Keynote Lecture at ICCCI'18. Springer Publishers (2018)
12. Treur, J.: Relating emerging network behaviour to network structure. In: Proceedings of the 7th International Conference on Complex Networks and their Applications, ComplexNetworks'18. Studies in Computational Intelligence, Springer (2018)
13. Wijs, A., Katoen, J.P., Bošnacki, D.: Efficient GPU algorithms for parallel decomposition of graphs into strongly connected and maximal end components. *Form. Methods Syst. Des.* **48**, 274–300 (2016)



Random Graph Generators for Hyperbolic Community Structures

Saskia Metzler¹(✉) and Pauli Miettinen²

¹ Max Planck Institute for Informatics, Saarland Informatics Campus, Saarbrücken, Germany

{sметzлер, saskia.metzler}@mpi-inf.mpg.de

² University of Eastern Finland, Kuopio, Finland

pauli.miettinen@uef.fi

Abstract. Proper testing of graph mining algorithms, for example, algorithms for community detection, requires the capability of creating realistic random graphs. As our understanding of real graph communities evolves, so should the random graph generators evolve, too. In this work, we propose a random graph generator called HYGEN that, unlike the existing random graph generators, is designed to preserve the community structure of real networks, especially the commonly observed hyperbolic intra-community connectivity structure. The generated graphs will also preserve the total degree distributions and clustering coefficients of the original graph without introducing too much determinism. In addition, we also propose realistic distributions for the parameters controlling the hyperbolic shape of the communities.

Keywords: Random graphs · Graph generators
Community detection · Hyperbolic community structure

1 Introduction

Real-world networks often not only display a modular composition, but also characteristic structures within the constituents. Previous work has established that structure of communities is described well by a *hyperbolic model* [4, 18]. This model can express the particular core-tail structure which is frequently observed in real-world networks and is suitably general to also represent clique-like connectivity (see Fig. 1b). Especially communities in social networks show a pronounced core-tail structure: a small fraction of the members have strong ties to each other and form the core. The majority of members only have ties to the core and not to each other [2, 19, 21, 23].

Understanding the organization of such networks is a primary goal of social sciences and requires competent algorithms for detecting and describing the structures. The algorithms have to be tested, though, and a thorough testing requires significant amounts of reliably labelled test data – which is often not available. Good random graph generators can be used to alleviate this problem.

There exist a variety of different graph generators: the Watts–Strogatz model [24], the Barabási–Albert model [3], the stochastic block model [10], and so on. They are designed with a focus on different features of (real-world) graphs, such as small-world or scale-free properties. To the best of our knowledge, however, no existing random graph generator is designed to model graphs consisting of hyperbolic communities.

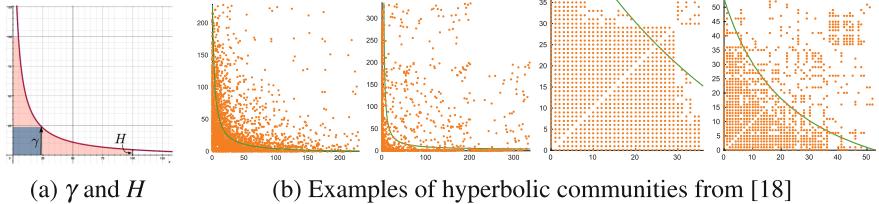


Fig. 1. Visualization of parameters for the hyperbolic model, and real-world examples. Models are shown on the degree-ordered adjacency matrix of each community.

As we believe that many real-world graphs actually follow a hyperbolic model, we introduce a novel random graph generator to fill this gap. HYGEN generates modular networks with realistic intra-community structures using parameter distributions derived from observations on real graphs.

2 Existing Random Graph Generators

The emphasis of our approach is to model the intra-community structure of graphs realistically. Many existing random graph generators do not consider the concept of community structure at all. The classical Erdős–Rényi [8] model is one of them, as are the Barabási–Albert [3] model and the Forest fire model [16]. Approaches for *graph expansion* [22,25] typically focus on the global level of modelling real-world graphs. Likewise do hyperbolic geometric graphs [13] whose construction is based on *hyperbolic geometry*. Despite the related-sounding name, there is no direct resemblance to our model. Kronecker graphs [15] permit community structure, but due to their recursive construction using the Kronecker product, communities consist of self-similar building blocks and do not vary in size.

Graph generators that permit community structure and that we consider most relevant in comparison to our method are the stochastic block model (SBM) [10], in particular the variation called degree-corrected SBM (DC-SBM) [11], the R-MAT model [6], and the Lancichinetti–Fortunato–Radicchi (LFR) benchmark [14]. Notice that neither DC-SBM, nor R-MAT, nor LFR provide solutions to the modelling task we aim to solve. We tested to what extent an ideal hyperbolic structure could be captured by the different graph generators and found that none of them captures it well (we omit the details). Some reasons for this are discussed below.

The stochastic block model (SBM) is a popular random graph model that partitions vertices into blocks. The vertices within each block are stochastically equivalent. But SBMs cannot model uneven degree distributions within communities. Degree-corrected SBMs (DC-SBM) [11] are a variation where an additional degree parameter is incorporated for each vertex such that uneven edge probabilities can be accounted for. A crucial detail of this approach is that the probability of a node forming an edge is a global property. Degree-generated SBMs [26], although they overcome the shortcoming of DC-SBMs to not separate vertices based on degree even when that would be the correct partitioning, are similar in that respect. We in contrast assume different probabilities for a node to form inter- or intra-community edges.

R-MAT [6] is based on the recursive construction of an adjacency matrix. The proposed algorithm recursively subdivides this matrix into four equally sized partitions and distributes the edges within according to partition-specific probabilities. This special construction limits the shape of the communities that can be attained.

The Lancichinetti–Fortunato–Radicchi (LFR) benchmark [14] is a graph generator proposed to test community detection algorithms. It can produce overlapping community structures as well as weighted and directed networks. It extends the Girvan–Newman benchmark [9] with emphasis on features of real-world graphs such as heterogeneous distributions of the overall node degree and the community size. Still we observe non-realistic intra-community structures showing a nearly uniform degree distribution.

3 Hyperbolic Community

Given an undirected graph $G = (V, E)$ with n nodes and m edges, we assign a number from $\{0, \dots, n - 1\}$ to the vertices and use (i, j) to denote both a pair of vertices and the (potential) undirected edge between them. We call a tuple (V_C, π_C, Θ_C) a *community* C . The set $V_C \subseteq V$ contains the nodes of the community, and we write $n_C = |V_C|$. The permutation $\pi_C: V_C \rightarrow \{0, \dots, n_C - 1\}$ orders the nodes, and Θ_C denotes a set of parameters. The hyperbolic community model assumes the nodes to be ordered according to their degrees inside the community. In the model, not every edge between the nodes in V_C is necessarily part of the community – otherwise all communities would be cliques.

Following [18], community models are defined using functions $f: \{0, \dots, n_C - 1\} \times \{0, \dots, n_C - 1\} \times \Theta_C \rightarrow \{0, 1\}$ operating on a set of parameters Θ_C and deciding for any pair of vertices $(i, j) \in \{0, \dots, n_C - 1\} \times \{0, \dots, n_C - 1\}$ if an edge between i and j is part of the community or not. Notice that the function f only gets the indices relative to the subgraph, not to the full graph. Thus, to test a pair $(i, j) \in V_C \times V_C$, we need to compute $f(\pi_C(i), \pi_C(j), \Theta_C)$.

Metzler et al. [18] define multiple equivalent parameter sets Θ . We describe our model using $\Theta = \text{fixed}(\gamma, H)$, which provides an immediate intuition about the shape of the connectivity pattern of the community: γ defines the size of

the core (a clique) and H indicates how thick the tail¹ is (see Fig. 1a). In the subsequent analysis it is sometimes useful to consider $\Theta = \text{hyperbolic}(p, \theta)$ instead. The parameters then have an immediate geometric interpretation: p defines the centre of an hyperbola at $(-p, -p)$, and an edge (i, j) is in considered to be part of the community if $(i + p)(j + p) \leq \theta$ (see also [18]).

Yet alternatively, the model can equivalently be expressed in terms of $\Theta = \text{mixture}(x, \Sigma)$ [18]. With this formulation, the generality of the hyperbolic model is most evident. The mixture parameter $x \in [-1, 1]$ indicates how much the model looks like a line and how much like a hyperbola centered at the origin: $(1 - |x|)(i \cdot j) + x(i + j) \leq \Sigma$. Controlling the boundary condition Σ while fixing $x = 0$ will yield communities that strictly follow a power law connectivity pattern.

Data: distributions D_{size} , D_γ , D_H , densities d_{inside} , $d_{outside}$, number of communities k

Result: random graph G

```

for  $i = 1 : k$  do
    draw size  $s$  from  $D_{size}$ ,  $\gamma$  from  $D_\gamma$ , and  $H$  from  $D_H$ 
    scale  $\gamma$  according to  $s$ , and  $H$  according to  $\gamma$ 
    make model fixed( $\gamma$ ,  $H$ )
    select edges to discard uniformly at random to reach  $d_{inside}$ 
    plant result into  $G$ 
apply noise  $d_{outside}$  to the outside community area of  $G$ 
return  $G$ 
```

Algorithm 1: HYGEN algorithm

4 Our Model

In this section, we propose HYGEN, the random graph generator that accounts for specific intra-community connection patterns that are frequently observed in real-world social networks [4, 18]. We first explain the construction of individual communities and then describe how a graph of multiple such communities is obtained.

To generate a single random hyperbolic community of size n_C , we need to sample its core size γ , and its tail height H from predefined distributions. Based on observations in real-world data sets, we assume that γ (relative to n_C) can be well modelled through a Normal distribution with mean μ and variance σ^2 , and H (as a fraction of γ) follows an Exponential distribution with a decay rate of λ . (More on the distribution functions in Sect. 6.) Assuming μ , σ^2 , and λ are given,

¹ More commonly, intra-communities structures are described as *core* and *periphery* [5]. We use the notion of *core* and *tail* instead since the hyperbolic model allows for more shape variations than the term periphery implies: Tails may get progressively thinner while nodes in the periphery are assumed to be evenly connected to the core.

the resulting community C of size n_C then perfectly follows the model defined by the sampled parameter set $\text{fixed}(\gamma, H)$. Real communities however are typically inexact. We apply a uniform noise model with separate parameters d_{inside} and $d_{outside}$ for edges inside and outside the community. This means that every edge from inside C is retained with probability d_{inside} , and with probability $d_{outside}$, edges are introduced outside the community.

With individual hyperbolic communities as building blocks HYGEN generates graphs of k communities, obtained sampling their sizes from the distribution of community sizes D_{size} and drawing the parameters γ and H for every community (see Algorithm 1). While our experiments suggest to draw from a Generalized extreme value distribution, a power law function as used in [14] is a considerable alternative.

Notice that we make the following assumptions: Noise is not only constant within each community but also the same among all communities. We model the area outside communities with a uniform density. We assume the size and shape of the communities to be uncorrelated. Communities are assumed to be non-overlapping.

The time complexity of Algorithm 1 is as follows. For a community C , let E_C^p be the edges a “perfect” community (i.e. one with no noise) would have and let E^p be that for the full graph. Let E^n be the set of edges noise adds to the graph (inter and intra community). Then, drawing the parameters and adjusting them is $O(1)$ operation, making the model takes $O(n_C)$, and sampling the edges to discard from the community can be done in $O(|E_C^p|)$ [12, p. 137]. Repeated k times this becomes $O(k(n_c + |E_C^p|)) = O(|V| + |E^p|)$. This leaves the part to add the noise; to that end, we need to do sampling without replacement over a population of $O(|V|^2 - |E_p|)$ edges, taking essentially linear time. When $|E_p|$ and $|E^n|$ are small compared to $|V|^2$ (i.e. the graph is sparse), we can sample with replacement to obtain practically the same result, taking $O(|E^n|)$ time. In total, the full running time of Algorithm 1 is $O(|V| + |E^p| + |E^n|)$ which is only slightly more than $O(|V| + |E|)$.

Finally, it is worth noticing that our model can also be generalized as a *graphon* [20]. The division to communities works the same way as when modelling the stochastic block models as graphons; only the edge probability inside the communities is not uniform but rather depends on the model for the community. The modelling as a graphon facilitates the analysis of infinitely large random graphs and the convergence and concentration features of our model, but these are beyond the scope of this manuscript.

5 Properties of HYGEN Random Graphs

HYGEN preserves important measures of network connectivity. The degree distribution and the clustering coefficient are frequently used to describe the connectivity patterns of networks [1]. In this section, we show that these measures are retained when generating a new graph from the parameters observed in an existing network.

For a graph G , consisting of disjoint communities C that perfectly follow the hyperbolic models $\text{hyperbolic}(p, \theta)$ we have:

Lemma 1. *The degree distribution $d: V_C \rightarrow \{1, \dots, n_c\}$ of C is determined through the parameters p and θ .*

Proof. The model defines that an edge $(i, j) \in \text{hyperbolic}(p, \theta)$ if $(i+p)(j+p) \leq \theta$. This inequality can be reformulated such that

$$j \leq \theta/(i+p) - p . \quad (1)$$

For every $i \in V_C$, the highest integer j that fulfils (1) is equal to the degree of node i , and hence $d(i) = \max \{j \in N_C : j \leq \theta/(i+p) - p\}$ defines the degree distribution. \square

The more intuitive integer parameters H and γ that indicate the size of the core and the tail can alternatively be used in this derivation:

Corollary 1. *The degree distribution of C is determined through H and γ .*

This follows directly from Eqs. (7) and (8) in [18]. We show that the same also holds for the entire graph G .

Lemma 2. *The degree distribution of G is determined by the parameters of its hyperbolic communities.*

Proof. Lemma 1 applies for every community in G . As the communities are disjoint, we obtain the overall degree distribution choosing p and θ according to the respective community and evaluating the inequality for index $\pi_C(i)$ referencing the node within that community (see Sect. 3),

$$d(i) = \sum_{C \in G} \max \{\pi_C(j) \in V_C : \pi_C(j) \leq \theta_C / (\pi_C(i) + p_C) - p_C\} . \quad \square \quad (2)$$

In real-world data sets, the assumption of perfect hyperbolic models we made for the above result is typically not met. Although the hyperbolic model covers a variety of connectivity patterns, such as the classic power law-like structure, as well as the extremes of a star and a clique, real-world data is often noisy and the hyperbolic models only approximately describe the data. We now assess how much the error in the modelling affects the resulting degree distribution.

Suppose $q \in [0, 1]$ is the average noise of graph G . That is, a fraction of q edges are missing from inside the communities of G and the outside-community area has a fraction of q surplus edges, i.e. $d_{\text{outside}} = q$, $d_{\text{inside}} = 1 - q$. Then, for node i with a unperturbed degree of $d(i)$, the expected degree $\bar{d}(i)$ is

$$\bar{d}(i) = d(i) - qd(i) + q(n - d(i)) = d(i) + q(n - 2d(i)) . \quad (3)$$

The relative degree of a node i is the fraction $\alpha(i)$ of all nodes of G to which i is connected. Hence, the relative expected degree is $\bar{d}(i)/n = \alpha(i) + q(1 - 2\alpha(i))$.

Noise has the strongest effect on a node if the degree $d(i)$ is near its limits, i.e. if $\alpha \approx 1$ or $\alpha \approx 0$. For a node with $\alpha = 0.5$, the presence of noise on expectation will not affect the degree at all. The star pattern is an example of a graph where the degree distribution is heavily impaired when the noise factor q is high.

The clustering coefficient is also determined through the parameters of the hyperbolic models that constitute a graph G . Two well-known variations of the clustering coefficient exist: the global and the local [1]. While the former is the overall ratio of triangles to wedges in a graph, the latter is computed per node and denotes the fraction of triangles around a node.

Suppose C is a perfect hyperbolic community $\text{hyperbolic}(p, \theta)$.

Lemma 3. *The local clustering coefficient CC_i of the nodes of C is determined through p and θ .*

Proof. The local clustering coefficient for node i of C is given by

$$CC_i = \frac{2 |\{(j, h) : j, h \in \Gamma(i), (j, h) \in E\}|}{d(i)(d(i) - 1)}, \quad (4)$$

where $\Gamma(i)$ denotes the set of nodes directly connected to i . The nominator counts twice the edges between nodes j and h that are both connected to i . As C perfectly follows $\text{hyperbolic}(p, \theta)$ we may use (1) to express $\Gamma(i)$ as $\Gamma(i) = \{h \in V_C : h \leq \theta/(i + p) - p\}$. The set of edges, E , is $E = \{i, j \in V_C : j \leq \theta/(i + p) - p\}$. For the degree $d(i)$, used in the denominator to indicate the size of the neighbourhood, Lemma 1 applies. \square

To make a similar statement about the global clustering coefficient CC , we first need some definitions. Define the indicator function $\chi(i, j)$ as

$$\chi(i, j) = \begin{cases} 1 & j \leq \theta/(i + p) - p \\ 0 & \text{otherwise.} \end{cases} \quad (5)$$

This function returns for every pair of nodes i and j of community C whether there exists an edge between them according to $\text{hyperbolic}(p, \theta)$.

To count the number of triangles in C , we define $T(i, j, h) : \mathbb{N}_{\geq 0}^3 \rightarrow \{0, 1\}$ as

$$T(i, j, h) = \chi(i, j) \cdot \chi(i, h) \cdot \chi(j, h). \quad (6)$$

$T(i, j, h)$ decides on the basis of the hyperbolic model whether a triangle exists between i , j , and h by checking the presence of the three possible edges.

For the existence of wedges in C , we define $W(i, j, h) : \mathbb{N}_{\geq 0}^3 \rightarrow \{0, 1\}$ as

$$\begin{aligned} W(i, j, h) = & (1 - \chi(i, j)) \cdot \chi(i, h) \cdot \chi(j, h) + \chi(i, j) \cdot (1 - \chi(i, h)) \cdot \chi(j, h) \\ & + \chi(i, j) \cdot \chi(i, h) \cdot (1 - \chi(j, h)). \end{aligned} \quad (7)$$

This function checks the presence of exactly two out of the three possible edges between i , j , and h within C . With these functions defined, we now show the dependency between the clustering coefficient and the hyperbolic model.

Lemma 4. *The global clustering coefficient CC of a community C is determined through p and θ .*

Proof. The global clustering coefficient of a graph is three times the number of triangles divided by the number of wedges. If C perfectly follows the hyperbolic model $\text{hyperbolic}(p, \theta)$, we can use (6) to check for the presence of a triangle for every node triple (i, j, h) and (7) to check for the presence of a wedge. Thus, we compute

$$CC = \frac{3 \sum_{i=0}^{n-1} \sum_{j=i+1}^{n-1} \sum_{h=j+1}^{n-1} T(i, j, h)}{\sum_{i=0}^{n-1} \sum_{j=i+1}^{n-1} \sum_{h=j+1}^{n-1} W(i, j, h)}. \quad \square \quad (8)$$

We could compute the number of triangles with only a single sweep over the nodes: there are $\binom{\gamma+1}{3}$ triangles in the core, and every i with $d(i) \geq 2$ adds $\binom{d(i)}{2}$ more triangles because it only has connections to the core. To the best of our knowledge, there is no similar expression to determine the number of wedges.

Lemma 5. *The local and global clustering coefficient of G are determined through the parameters of the hyperbolic communities of G .*

Proof. For a graph G consisting of multiple disjoint hyperbolic communities, the clustering coefficients behave analogously. $T(i, j, h)$ and $W(i, j, h)$ need to be evaluated with respect to the community of the nodes. They must yield 0 in case i , j , and h belong to different communities and use the community specific parameters p_C and θ_C to evaluate 1_C otherwise. \square

It is not trivial to analyse the effects of noise to the clustering coefficient. Triangles or wedges from the inside-community area disappear, and new ones get introduced involving the outside-community area. Given the overall density of a graph, the expected number of triangles or wedges is derivable, but integrating the specific intra-community structure into this expectation remains an open problem.

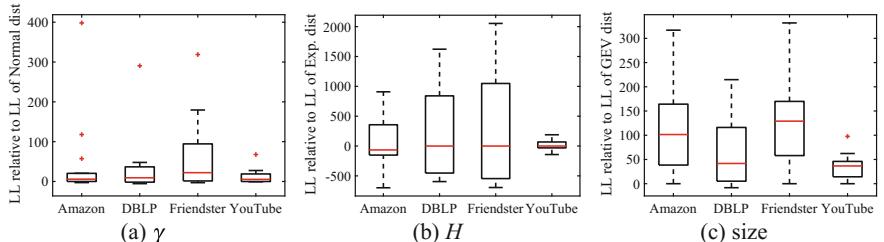


Fig. 2. Fitting quality of the tested distribution functions compared to that of the chosen distribution for each observed parameter in each dataset.

6 Empirical Parameter Distributions for HYGEN

In this section we detail how we obtained the suggested parameter distributions.

We use four networks with community information from the Stanford Large Network Dataset collection [17]: **Amazon**, **DBLP**, **Friendster**, and **YouTube**. We draw a sample of 500 communities of size between 100 and 1000 nodes and compute the hyperbolic models for each community individually. This yields empirical distributions of γ , H , and the (truncated) community size n_C . Note that the **YouTube** network has only 129 communities of the respective size.

For each of the empirical distributions for γ , H , and n_C , we fit a variety of distribution functions (Generalized extreme value, Inverse Gaussian, Birnbaum–Saunders, Exponential, log-Normal, log-Logistic, Gamma, Rayleigh, Weibull, Nakagami, Rician, Normal, Logistic, Extreme value, t -location-scale). Not every distribution is applicable for each of the parameters. While the observed γ s look normally distributed, H and the community size show an exponential behaviour. Nevertheless, we evaluate the fitting quality in terms of negative log-likelihood (LL) for each possible distribution.

As we optimize the parameters of the distribution functions to explain the empirical observations, we notice that many of the compared distribution functions yield a similar shape. We compare the fit quality of the distribution we choose to that of every other possible distribution function. To do so, we subtract the negative LL of the chosen distribution from the negative LL of the remaining distributions. The results are summarized as boxplots in Fig. 2 for each data set. A value of 0 indicates similar quality. Higher values indicate that the chosen distribution function is a better fit, and lower values that another distribution fits the observations better.

For the empirical distributions of γ , we decide to propose the Normal distribution as a good description across the studied datasets. In terms of fit quality compared to the other distribution functions, this seems to be a favourable choice for the four datasets. The empirical distributions of the parameter H show more variation in their shape, their clear commonality is that thin tails are much more frequent than thick ones. Deriving the exact shape of the parameter distributions given these samples appears challenging. As Fig. 2b indicates, the distribution functions fit the data with varying quality. We choose to represent H by an Exponential function, but dependent on the data set there might be better options. For the community size, the Generalized extreme value distribution (GEV) yields the best fit in terms of LL for every dataset (see Fig. 2c).

7 HYGEN Graphs from Known Distributions

While graphs modelled after empirical observations are potentially more realistic, HYGEN can also be used to construct random graphs from pre-defined parameter distributions.

Suppose we aim to generate random graphs where the communities show a power law connectivity. This task might not seem not easily expressible in

terms of γ and H , in particular because they are modelled independently. Recall that the hyperbolic model can be expressed with several equivalent formulations (see Sect. 3). If we express the model in terms of `mixture`(x, Σ) an immediate solution is obvious: x becomes fixed to constantly 0 and only the distribution of the boundary condition Σ needs to be supplied, for instance in form of a Normal distribution.

Generating clique-like connectivity within the communities is as easy as setting γ and H constantly to their maximum, i.e. 100 % of the size of the community. Notice that the same result could however be achieved with less modelling effort.

8 Stability of the Graph Generation

With our analysis on real-world data, we demonstrate two contrasting aspects: here, we show how well HYGEN-generated graphs adapt to the characteristic distributions of these networks; in Sect. 9, we show that the graph generation procedure introduces enough randomness such that the resulting graph differs from its template.

To test HYGEN on real-world data sets², we fit distributions to the observed parameter distributions from every data set described in Sect. 6. With the obtained distribution functions we use HYGEN to generate new random graphs. On these new graphs, we compute the best hyperbolic model of each community. For comparison, we also evaluate the performance of LFR and DC-SBM on the task of generating graphs with hyperbolic communities. While LFR-generated graphs, like graphs from HYGEN, are equipped with ground-truth community information, the DC-SBM implementation³ only derives the information about communities in the model fitting step. Hence, for this comparison we assume that DC-SBM correctly recovers communities in the graphs it has generated.

Figure 3 shows boxplots of the empirical distributions of γ , H , and the community size in comparison to the distributions obtained after computing hyperbolic models on the results of HYGEN, LFR, and DC-SBM. Notice that 100 times as many random communities were drawn than what was in the input data, i.e. 12900 communities for YouTube and 50000 for the other datasets. For performance reasons, DC-SBM generated only one graph with 100 communities for every data set.

We observe a good resemblance of the median γ s between the original and the HYGEN-generated graphs. For H , we observe mostly similar-looking distributions for the original communities and the HYGEN-generated ones. HYGEN however shows a tendency to produce communities with exceptionally thick tails. For the community size, Fig. 3c indicates, that although the medians of the generated graphs match the original, we have slightly less of the larger communities in the HYGEN-generated graphs. This is likely a result of the way we interpret the empirically observed distributions.

² Our code: <http://cs.uef.fi/~pauli/hybobo/rgg>.

³ <https://github.com/ntamas/blockmodel>.

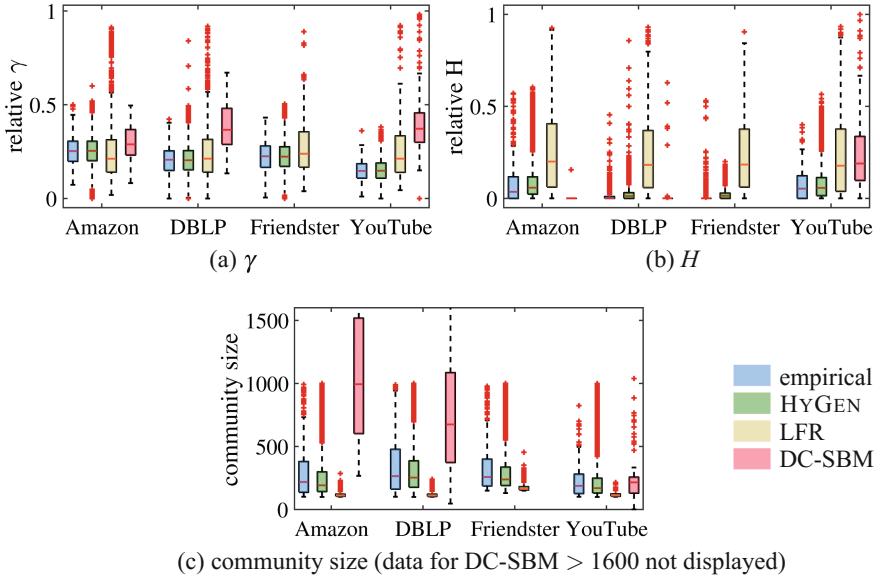


Fig. 3. Distributions of parameters in generated graphs compared to original data. H and γ are obtained after fitting hyperbolic models.

In the median, the shape of the communities LFR generates is quite accurate for the core size γ , however more communities with exceptionally large cores can be observed than in the original data. The height H of the tails is often overestimated. Community sizes are much smaller than in the empirical observation.

DC-SBM produces communities with thicker cores than originally observed, while the tail heights are underestimated, except in YouTube. Conversely, apart from YouTube, communities get much larger than originally observed, indicating that our assumption that DC-SBM correctly recovers communities in the graphs it generated may not hold. (Results for Friendster did not compute within a week.)

9 Randomness of the Generated Graphs

From the comparison of the parameter distributions of the original graphs and the generated graphs, we can conclude that their hyperbolic structure is similar. This however would also hold for an identical copy of the original graph. To assess whether the structurally similar graphs are also reasonably different to the original, we study their conditional entropy.

The conditional entropy $\mathcal{H}(Y|X)$ quantifies the amount of information needed to describe the outcome of a discrete random variable Y provided the discrete random variable X is known. The result is $\mathcal{H}(Y|X) = 0$ if Y is completely determined by X , and $\mathcal{H}(Y|X) = \mathcal{H}(Y)$ if and only if Y and X are independent [7]. As we are interested in the relative amount of dependency of Y

given X , we scale $\mathcal{H}(Y|X)$ by $\mathcal{H}(Y)$ to obtain a value within $[0, 1]$. We define the relative conditional entropy as $\mathcal{H}_{\text{rel}}(Y|X) = \mathcal{H}(Y|X)/\mathcal{H}(Y)$. Since the graphs to compare are binary, $\mathcal{X} = \mathcal{Y} = \{0, 1\}$. To interpret them as random variables X and Y , we vectorize the upper triangular of the adjacency matrices after ordering the nodes by their degree.

To compute $\mathcal{H}_{\text{rel}}(Y|X)$, we generate 100 random graphs from the observed parameters of each dataset keeping the distribution of community sizes identical to the original graph to ensure that the resulting sample is of the same size. We compute $\mathcal{H}_{\text{rel}}(Y|X)$ per community rather than for each entire generated graph for two reasons: First, the sizes of the communities are fixed for this experiment, which introduces an artificial amount of determinism that could bias the measurement. Second, from the original graphs, we sampled communities but their context is not maintained (in particular because we assume non-overlapping communities). Thus we would compare the inter-community structure of the original graph to uniformly at random distributed edges in the generated graphs. As our modelling focuses entirely on the intra-community structure, this comparison seems futile.

We obtain the following average relative conditional entropies over all the communities and all the samples: **Amazon** 0.996, **DBLP** 0.996, **Friendster** 0.990, **YouTube** 0.963 (standard deviations are within the displayed precision). This result indicates that the amount of determinism of the generated communities given the original is very small. Thus, we conclude that HYGEN generates structurally similar random graphs that still exhibit non-determinism compared to their template.

10 Conclusions

We have introduced the random graph generator HYGEN. Our results indicate that this generator is able to produce realistic intra-community structure, unlike the state-of-the-art methods. Despite simplifying assumptions, such as non-overlapping communities and a uniform noise model, HYGEN is a step towards more realistic random graph generators.

To further improve the modelling, important future work encompasses to handle overlapping communities as well as to offer more realistic noise models. While our current noise model assumes the average of the observed noise evenly for all communities, the examined data sets show evidence that noise is correlated with community size. Hence modelling noise in a more adjusted way could result in even more accurate models. In addition, while we estimate core sizes of communities precisely, covering in particular thin tails of them with higher accuracy could yield further improvement.

HYGEN has its obvious use for testing community detection algorithms. It can generate realistic graphs equipped with reliable labelling of the communities. Besides this, HYGEN might serve as an anonymization tool to study the structure of social networks without revealing the participants identities.

References

1. Aggarwal, C.C., Wang, H. (eds.): *Managing and Mining Graph Data*. Springer, New York (2010)
2. Alba, R.D., Moore, G.: Elite social circles. *Sociol. Methods Res.* **7**(2), 167–188 (1978)
3. Albert, R., Barabási, A.L.: Statistical mechanics of complex networks. *Rev. Mod. Phys.* **74**, 47–97 (2002)
4. Araujo, M., Günemann, S., Mateos, G., Faloutsos, C.: Beyond blocks: hyperbolic community detection. In: ECMLPKDD '14, pp. 50–65 (2014)
5. Borgatti, S.P., Everett, M.G.: Models of core/periphery structures. *Soc. Netw.* **21**, 375–395 (1999)
6. Chakrabarti, D., Zhan, Y., Faloutsos, C.: R-MAT: a recursive model for graph mining. In: SDM '04, pp. 442–446 (2004)
7. Cover, T.M., Thomas, J.A.: *Elements of Information Theory*. Wiley, Hoboken, NJ (2006)
8. Erdős, P., Rényi, A.: On random graphs I. *Publ. Math. Debrecen* **6**, 290 (1959)
9. Girvan, M., Newman, M.E.J.: Community structure in social and biological networks. *Proc. Natl. Acad. Sci.* **99**(12), 7821–7826 (2002)
10. Holland, P., Laskey, K., Leinhardt, S.: Stochastic blockmodels: first steps. *Soc. Netw.* **5**(2), 109–137 (1983)
11. Karrer, B., Newman, M.E.J.: Stochastic blockmodels and community structure in networks. *Phys. Rev. E* **83**, 016,107 (2011)
12. Knuth, D.E.: *The Art of Computer Programming. Seminumerical Algorithms*, vol. 2, 2nd edn. Addison-Wesley, Reading, MA (1981)
13. Krioukov, D., Papadopoulos, F., Kitsak, M., Vahdat, A., Boguñá, M.: Hyperbolic geometry of complex networks. *Phys. Rev. E* **82**, 036,106 (2010)
14. Lancichinetti, A., Fortunato, S., Radicchi, F.: Benchmark graphs for testing community detection algorithms. *Phys. Rev. E* **78**(4), 046,110 (2008)
15. Leskovec, J., Chakrabarti, D., Kleinberg, J., Faloutsos, C., Ghahramani, Z.: Kronecker graphs: an approach to modeling networks. *J. Mach. Learn. Res.* **11**, 985–1042 (2010)
16. Leskovec, J., Kleinberg, J., Faloutsos, C.: Graphs over time: densification laws, shrinking diameters and possible explanations. In: KDD '05, pp. 177–187 (2005)
17. Leskovec, J., Krevl, A.: SNAP datasets: Stanford large network dataset collection (2014). <http://snap.stanford.edu/data>. Accessed 11 Feb 2016
18. Metzler, S., Günemann, S., Miettinen, P.: Hyperbolae are no hyperbole: modelling communities that are not cliques. In: ICDM '16, pp. 330–339 (2016)
19. Morgan, D.L., Neal, M.B., Carder, P.: The stability of core and peripheral networks over time. *Soc. Netw.* **19**(1), 9–25 (1997)
20. Orbanz, P., Roy, D.M.: Bayesian models of graphs, arrays and other exchangeable random structures. *IEEE Trans. Pattern. Anal.* **37**(2), 437–461 (2015)
21. Panzarasa, P., Opsahl, T., Carley, K.M.: Patterns and dynamics of users' behavior and interaction: network analysis of an online community. *J. Am. Soc. Inf. Sci. Technol.* **60**(5), 911–932 (2009)
22. Park, H., Kim, M.S.: EvoGraph: an effective and efficient graph upscaling method for preserving graph properties. In: KDD '18, pp. 2051–2059 (2018)
23. Reed, P.B., Selbee, L.K.: The civil core in disproportionality in charitable giving, volunteering, civic participation. *Nonprofit Volunt. Sect. Q.* **30**(4), 761–780 (2001)

24. Watts, D.J., Strogatz, S.H.: Collective dynamics of ‘small-world’ networks. *Nature* **393**(6684), 440–442 (1998)
25. Zhang, J.W., Tay, Y.C.: GSCALER: synthetically scaling a given graph. In: EDBT’16, pp. 53–64 (2016)
26. Zhu, Y., Yan, X., Moore, C.: Oriented and degree-generated block models: generating and inferring communities with inhomogeneous degree distributions. *J. Complex Netw.* **2**(1), 1–18 (2014)



Estimating Personal Network Size with Non-random Mixing via Latent Kernels

Swupnil Sahai¹, Timothy Jones², Sarah K. Cowan³, and Tian Zheng^{2(✉)}

¹ Mangalytics, San Francisco, CA, USA

swupnil@gmail.com

² Department of Statistics, Columbia University, 1255 Amsterdam Avenue, New York, NY 10027, USA

{timothy.jones,tian.zheng}@columbia.edu

³ Sociology Department, New York University, 295 Lafayette Street, New York, NY 10012, USA

sarahkcowan@nyu.edu

Abstract. A major problem in the study of social networks is estimating the number of people an individual knows. However, there is no general method to account for barrier effects, a major source of bias in common estimation procedures. The literature describes approaches that model barrier effects, or *non-random mixing*, but they suffer from unstable estimates and fail to give results that agree with specialists' knowledge. In this paper we introduce a model that builds off existing methods, imposes more structure, requires significantly fewer parameters, and yet allows for greater interpretability. We apply our model on responses gathered from a survey we designed and show that our conclusions better match what sociologists find in practice. We expect that this approach will provide more accurate estimates of personal network sizes and hence remove a significant hurdle in sociological research.

Keywords: Personal network size estimation · Non-random mixing
Barrier effects · Kernel-based models

1 Introduction

Social networks have become crucial for understanding and explaining social phenomena. A primary vehicle for collection of social network data are sample surveys that collect aggregated relational data (ARD) via questions of the form (“How many X ’s do you know?”) [5, 8]. However, understanding the composition of social networks through such surveys have traditionally been difficult. Recently, a number of more sophisticated models using ARD have been created to accurately estimate properties about an individual’s social network. In particular, McCormick et al. [5] developed a method to estimate both individual network size and degree distribution in a population using a battery of questions that can be easily embedded into existing surveys. In this paper, we extend their

work and propose a more powerful, yet statistically robust framework for estimating these quantities.

Although our method provides estimates of network sizes, we are mainly interested in estimating non-random social mixing or barrier effects. Barrier effects occur whenever some individuals systematically know more (or fewer) members of a specific subpopulation than would be expected under random mixing. Modeling the impact of barrier effects is essential for accurate estimation of network sizes. McCormick et al. proposed a latent non-random mixing matrix to capture these barrier effects [5]. This paper extends the discrete mixing matrix to a continuous, structured framework by deriving a latent kernel representation of social mixing patterns.

This paper is organized as follows. We begin with a review of previous attempts to measure personal network size and social mixing patterns. In Sect. 3, we derive a continuous latent mixing kernel framework which resolves these problems, and as a byproduct, enables estimation of the mixing kernel bandwidth as a function of age. In Sect. 4 we outline the priors and likelihood of our model, and describe our model fitting algorithm. We discuss in Sect. 5 the results of fitting the model to 1,190 survey responses from an online survey we designed, asking respondents how many people they know with certain first names. We draw on insights developed during the statistical modeling to offer guidelines for future improvements in Sect. 6.

2 Previous Research

Modeling social mixing arose as a byproduct of trying to eliminate bias in estimating personal network size estimates via the scale-up method [3]. Consider a population of size N . We can store the information about the social network connecting the population in an adjacency matrix $\Delta = [\delta_{ij}]_{N \times N}$ such that $\delta_{ij} = 1$ if person i knows person j . Throughout this paper we will assume you *know* someone if you know each other's name and would stop and talk at least for a moment if you ran into each other on the street or in a shopping mall. The personal network size is then $d_i = \sum_j d_{ij}$.

Naively asking survey respondents whether they know every single individual in the population is unrealistic. The scale-up method instead asks respondents about an entire set of people at once, for example, asking “how many women do you know who gave birth in the last 12 months?” as opposed to asking if she knows 3.6 million distinct people. More generally, the scale-up method uses responses to questions of this form (“How many X 's do you know?”) to estimate personal network size.

Suppose in this example you report knowing three women who gave birth which represents about one-millionth of all women who gave birth within the last year. Assuming that your personal network's demographics are similar to that of the whole country, we can then use this information to estimate that you know about one-millionth of all Americans. That is

$$\frac{3}{3.6 \text{ million}} \cdot (300 \text{ million}) \approx 250 \text{ people.}$$

However, this method of estimation suffers bias due to barrier effects. For example, since people tend to know others of similar age and gender, a 25-year old woman probably knows more women who have recently had an abortion than would be predicted just based on her personal network size and the number of women who have had an abortion [6]. Consequently, estimating personal network size by asking only “how many women do you know who have recently had an abortion?” will tend to overestimate the degree of 25-year old women.

McCormick et al. [5] accounted for nonrandom mixing by explicitly modeling the propensity for a respondent in ego group e to know members in alter group a . The ego and alter groups are defined by gender and a discrete age category. Age is binned into one of following $\{0-20, 21-40, 41-60, 61+\}$ and gender is allowed to take one of male or female. Let $M_{[(\mathcal{A}_i, g_i), (\mathcal{A}_j, g_j)]}$ be the probability that alter j is in age category \mathcal{A}_j and gender g_j , given that ego i in age category \mathcal{A}_i and gender g_i knows alter j . (M is called the mixing rate). In the personal network of an ego with gender g_i and age category \mathcal{A}_i , alters with gender g_j and age category \mathcal{A}_j are expected to take a $100 \times M_{[(\mathcal{A}_i, g_i), (\mathcal{A}_j, g_j)]}\%$ share. The model is then

$$y_{ik} \sim \text{Neg-Binom}(\mu_{ike}, \omega_k), \quad \mu_{ike} = d_i \sum_{\mathcal{A}_j, g_j} M_{[(\mathcal{A}_i, g_i), (\mathcal{A}_j, g_j)]} \left(\frac{N_{k, \mathcal{A}_j, g_j}}{N_{\mathcal{A}_j, g_j}} \right), \quad (1)$$

where $N_{k, \mathcal{A}_j, g_j}$ denotes the number of people in subpopulation \mathcal{G}_k and age category \mathcal{A}_j with gender g_j , and $N_{\mathcal{A}_j, g_j}$ denotes the number of people in age category \mathcal{A}_j with gender g_j .

After defining this new framework, McCormick et al. then applied it to survey responses from McCarty et al. [4], estimating not only the individual degrees of the respondents d_i but also the latent non-random mixing matrix M . Figure 1 displays the fitted mixing matrix separated by ego age category and gender. The six plots correspond to the six rows of M (ego ages were binned into three categories), while the eight bars within each plot correspond to the eight columns of M . Most notable is that the alter age distributions for each of the six egos peak at the egos’ age categories (i.e. the social networks of young egos are dominated proportionally by young alters, while the social networks of old egos are dominated proportionally by old alters), a phenomenon known as homophily in sociology. The model’s ability to estimate homophily without requiring priors on the elements of the mixing matrix is a testament to the power of the latent non-random mixing matrix model. However, this model is not without its faults. In particular, Fig. 1 also implies that the all six ego categories know more 0–20 year old females than they do 0–20 year old males, in some cases by several orders of magnitude (e.g., the difference is dramatic for adult egos). This extreme behavior is unexpected, especially given that the national prevalence of males and females is nearly identical amongst 0–20 year olds. While it seems plausible that such a result could be due to the variability of the survey respondents (i.e. perhaps this particular survey’s respondents included many individuals who know a lot of young females), our simulated data experiments point to a different explanation. Figure 2 shows the results of constructing a balanced mixing matrix, simulating

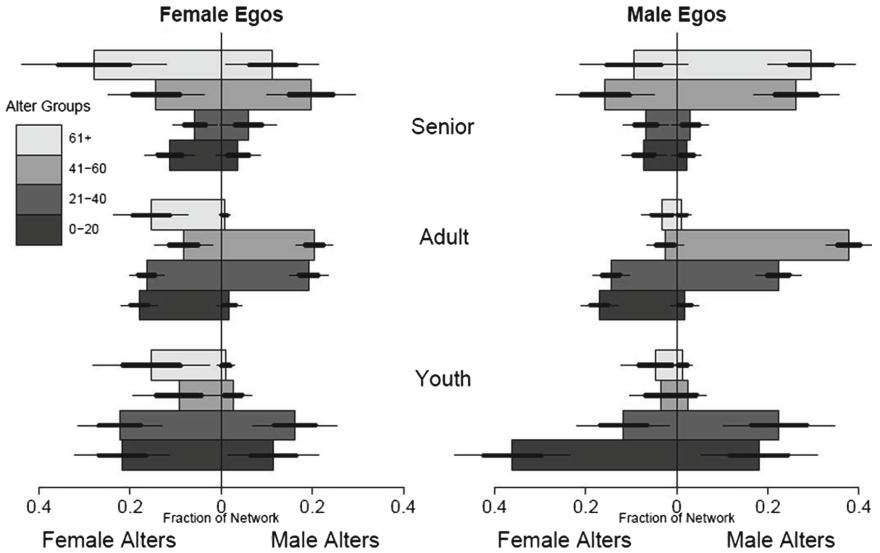


Fig. 1. The latent non-random mixing matrix estimated from survey respondents who were asked “How many X’s do you know?”, where X were 12 different names. Each horizontal bar represents the magnitude of an element of the mixing matrix $M_{6 \times 8}$. Reproduced from Fig. 5 in McCormick et al. [5]

responses to questions about 14 names, and then trying to recover the mixing matrix by fitting the latent non-random mixing matrix model to a subset of the responses (starting with just 4 names and going up to 14 names). The estimated mixing matrices show extreme biases for certain alter groups, even when using all 14 names. Further still, repeated response simulations from the same fixed mixing matrix produced different posterior estimates of the mixing matrix with the same order of bias. The bias and variance patterns appear sporadic between sets of simulations. We believe this is due to the non-identifiability of the mixing proportions, which stems from the lack of structure imposed on M ’s elements.

3 Latent Kernel Representation of Social Mixing

As a way to place constraints on the elements of the mixing matrix, we attempt to model the social mixing rates using a more structured framework. In particular, we replace the discrete histograms in Fig. 1 with smooth continuous curves. In order to do this, we first treat age as continuous rather than binning age into categories. Then, to create a standard functional form that can be consistent across all ego ages, we use a Gaussian kernel as our continuous curve. This allows us to replace the $e \times (a - 1)$ parameters used by the non-random mixing matrix with a nonrandom mixing kernel which imposes structure and requires significantly fewer parameters to be estimated.

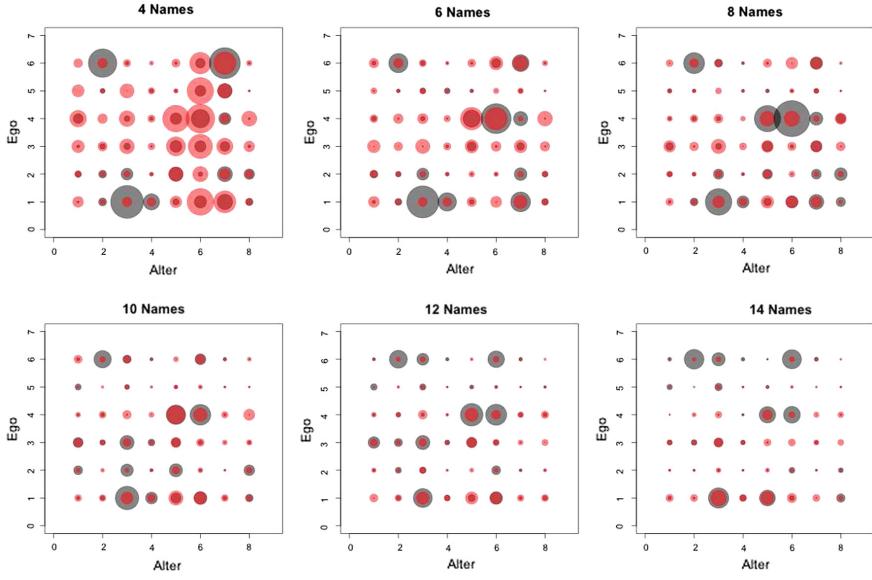


Fig. 2. The bias and standard error of the posterior mean of each element of the mixing matrix, estimated using simulated responses to 14 names and fitting to 4, 6, 8, 10, 12, and 14 names. The size of the black circles corresponds to bias, and the size of red circles correspond to standard error.

3.1 Latent Mixing Kernel

Consider the likelihood of alter j being age a_j , conditioning on alter j being of gender g_j , ego i is connected to alter j , and ego i is of age a_i and gender g_i ,

$$\begin{aligned} p(a_j|g_j, a_i, g_i, i \rightarrow j) &= \frac{1}{\sqrt{2\pi\lambda_{g_i g_j}(a_i)}} \exp\left\{-\frac{(a_i - a_j)^2}{2\lambda_{g_i g_j}(a_i)}\right\} \\ &= \text{Normal}(a_j|a_i, \lambda_{g_i g_j}(a_i)), \end{aligned}$$

where $\lambda_{g_i g_j}(a_i)$ is a latent factor that can be interpreted as the bandwidth of the age mixing kernel as a function of ego age a_i and $\{i \rightarrow j\}$ is the event that i knows j .

In order to keep $\lambda_{g_i g_j}(a_i)$ flexible without drastically increasing model complexity, we parameterize the bandwidth as a function of age using a fourth order spline [1] with the knots set to the deciles of the population age distributions. With this setup, and after adding an intercept term, the expression for the kernel bandwidth becomes

$$\lambda_{g_i g_j}(a_i) = \alpha_0^{g_i g_j} + \sum_{n=1}^{13} \alpha_n^{g_i g_j} B_{n,k}(a_i). \quad (2)$$

As for gender, if we assume that gender mixing does not depend on an ego's age, then we can use a gender-only mixing matrix to model the likelihood of alter j

being gender g_j , given that ego i with gender g_i and age a_i knows alter j . Thus we have

$$\begin{aligned} p(g_j|a_i, g_i, i \rightarrow j) &= p(g_j|g_i, i \rightarrow j) \\ &\triangleq \rho_{g_i g_j}. \end{aligned}$$

With the above quantities defined, the alter demographic distribution of an ego's network is

$$\begin{aligned} p(a_j, g_j|a_i, g_i, i \rightarrow j) &= p(g_j|a_i, g_i, i \rightarrow j)p(a_j|g_j, a_i, g_i, i \rightarrow j) \\ &= \rho_{g_i g_j} \text{Normal}(a_j|a_i, \lambda_{g_i g_j}(a_i)). \end{aligned}$$

3.2 Expectation Derivation

With the kernel framework outlined, we can derive the expected number of alters known in \mathcal{G}_k by ego i as follows:

$$\begin{aligned} \mu_{ik} &= d_i \sum_{g_j} \int_{a_j} p(j \in \mathcal{G}_k, a_j, g_j|a_i, g_i, i \rightarrow j) da_j \\ &= d_i \sum_{g_j} \rho_{g_i g_j} \int_{a_j} p(j \in \mathcal{G}_k|a_j, g_j) \text{Normal}(a_j|a_i, \lambda_{g_i g_j}(a_i)) da_j \\ &= d_i \sum_{g_j} \rho_{g_i g_j} \left(\int_a p(j \in \mathcal{G}_k|a, g_j) da \right) \int_{a_j} \left(\frac{p(j \in \mathcal{G}_k|a_j, g_j)}{\int_a p(j \in \mathcal{G}_k|a, g_j) da} \right) \text{Normal}(a_j|a_i, \lambda_{g_i g_j}(a_i)) da_j \\ &\approx d_i \sum_{g_j} \rho_{g_i g_j} \left(\int_a p(j \in \mathcal{G}_k|a, g_j) da \right) \int_{a_j} \text{Normal}(a_j|\mu_{kg_j}, \sigma_{kg_j}^2) \text{Normal}(a_j|a_i, \lambda_{g_i g_j}(a_i)) da_j \\ &= d_i \sum_{g_j} \rho_{g_i g_j} \left(\int_a p(j \in \mathcal{G}_k|a, g_j) da \right) \frac{e^{-\frac{(a_i - \mu_{kg_j})^2}{2(\lambda_{g_i g_j}(a_i) + \sigma_{kg_j}^2)}}}{\sqrt{2\pi(\lambda_{g_i g_j}(a_i) + \sigma_{kg_j}^2)}}. \end{aligned}$$

Here we take the probability of alter j being in group \mathcal{G}_k given alter's age a_j and gender g_j , normalize it with respect to the alter's age, approximate the normalized quantity with its discrete age counterpart, and approximate the discrete age normalized quantity with a Normal distribution so that:

$$\frac{p(j \in \mathcal{G}_k|a_j, g_j)}{\int_a p(j \in \mathcal{G}_k|a, g_j) da} \approx \frac{N_{k,a_j,g_j}/N_{a_j,g_j}}{\sum_a N_{k,a_j,g_j}/N_{a,g_j}} \approx \text{Normal}(a_j|\mu_{kg_j}, \sigma_{kg_j}^2).$$

The center and scale of the approximating normal, μ_{kg_j} and $\sigma_{kg_j}^2$, are estimated from \mathcal{G}_k 's population distributions $N_{k,a_j,g_j}/N_{a_j,g_j}$. These quantities are analogous, though not exactly equal, to the center and scale of the age distribution of individuals in \mathcal{G}_k with gender g_j .

In practice, the survey respondents' ages are grouped into intervals. In such cases, we can approximate the integral $\int_a p(j \in \mathcal{G}_k|a, g_j) da$ by the summation $\sum_\alpha \mathbb{P}(j \in \mathcal{G}_k|\alpha, g_j)$ over discrete age $\alpha \in \{0, 1, 2, \dots\}$ so that the expression

becomes

$$\begin{aligned}
\mu_{ik} &= d_i \sum_{g_j} \rho_{g_i g_j} \left(\int_a p(j \in \mathcal{G}_k | a, g_j) da \right) \frac{e^{-\frac{(a_i - \mu_{kg_j})^2}{2(\lambda_{g_i g_j}(a_i) + \sigma_{kg_j}^2)}}}{\sqrt{2\pi(\lambda_{g_i g_j}(a_i) + \sigma_{kg_j}^2)}} \\
&\approx d_i \sum_{g_j} \rho_{g_i g_j} \left(\sum_{\alpha} \mathbb{P}(j \in \mathcal{G}_k | \alpha, g_j) \right) \frac{e^{-\frac{(a_i - \mu_{kg_j})^2}{2(\lambda_{g_i g_j}(a_i) + \sigma_{kg_j}^2)}}}{\sqrt{2\pi(\lambda_{g_i g_j}(a_i) + \sigma_{kg_j}^2)}} \\
&= d_i \sum_{g_j} \rho_{g_i g_j} \left(\sum_a \frac{N_{k,a,g_j}}{N_{a,g_j}} \right) \frac{e^{-\frac{(a_i - \mu_{kg_j})^2}{2(\lambda_{g_i g_j}(a_i) + \sigma_{kg_j}^2)}}}{\sqrt{2\pi(\lambda_{g_i g_j}(a_i) + \sigma_{kg_j}^2)}}. \tag{3}
\end{aligned}$$

Compared to the unrestricted elements of the latent mixing matrix in Eq. 1, the latent kernel framework allows us to have more control over the shape of the distributions in Fig. 1. Additionally, this framework allows us to estimate the alter demographic distributions for egos of any arbitrary age, rather than just for egos in six age categories.

This latent mixing kernel framework is also more parsimonious. Indeed, our gender matrix $\rho_{2 \times 2}$ only requires estimation of two parameters and the mixing kernel only requires estimation of four kernel bandwidths $\lambda_{g_i g_j}$. Consequently, this framework requires estimation of only six parameters whereas the latent mixing matrix requires estimation of 42 free parameters.

4 Latent Kernel Model for Aggregated Relational Data (ARD)

Now we introduce the model for the aggregated relational data where μ_{ik} in Eq. 1 is replaced with Eq. 3. More specifically,

$$y_{ik} \sim \text{Neg-Binom}(\mu_{ik}, \omega_k) \tag{4}$$

with μ_{ik} as defined in Eq. 3. Similar to McCormick et al. [5], we fit the proposed model under a Bayesian framework. We introduce priors for the model parameters as follows. The degree parameters d_i are given log normal distributions

$$\log d_i \sim \text{Normal}(\delta_i, \eta).$$

We choose a functional form for δ_i that allows us to place a latent dependence of gender and age on ego degree.

$$\delta_i = \beta_1 + \beta_2 \mathbb{I}\{g_i = \text{Female}\} - e^{\beta_3} \left(\frac{a_i - \bar{a}}{\bar{a}} \right)^2,$$

where β_1, β_2 , and β_3 are latent coefficients and \bar{a} is the average age of the population estimated from the population age distribution. Furthermore, we place

priors on η, β_1, β_2 , and β_3 so that the priors $p(\delta_i)$ match the empirical distribution of degree estimates from McCormick et al. [5]. We put

$$\begin{aligned}\log \eta &\sim \text{Normal}(-0.7, 0.1), \\ \beta_1 &\sim \text{Normal}(6, 1), \quad \beta_2 \sim \text{Normal}(0, 1), \quad \beta_3 \sim \text{Normal}(-3, 1).\end{aligned}$$

We also place an noninformative prior on the gender mixing matrix's rows

$$\rho_{1\cdot} \sim \text{Dirichlet}(1, 1), \quad \rho_{2\cdot} \sim \text{Dirichlet}(1, 1).$$

We place the following priors on the spline coefficients from Eq. 2 for regularization

$$\alpha_0^{g_i g_j} \sim \text{Normal}(0, 2), \quad \alpha_n^{g_i g_j} \sim \text{Normal}(0, \tau), \quad \tau \sim \text{Normal}^+(0, 2).$$

Lastly, we place a prior on the inverse overdispersion $\omega'_k = 1/(1/\omega_k - 1)$ so that

$$\omega'_k \sim \text{Beta}(\alpha = 4.5, \beta = 0.5),$$

where α and β are chosen so that the priors $p(\omega'_k)$ match the empirical distribution of inverse overdispersion estimates from McCormick et al. [5].

Letting θ denote the set of all unknown parameters and letting y denote the survey responses, the posterior is then

$$\begin{aligned}\theta|y &\sim \prod_{k=1}^K \text{Beta}(\omega'_k|4.5, 0.5) \prod_{i=1}^N \text{NegBinomial}(y_{ik}|\mu_{ik}, \omega'_k) \\ &\times \text{Normal}(\log \eta | -0.7, 0.1) \prod_{i=1}^N \text{Normal}(\log d_i | \delta_i, \eta) \\ &\times \text{Normal}(\beta_3 | -3, 1) \text{Normal}(\beta_2 | 0, 1) \text{Normal}(\beta_1 | 6, 1) \text{Normal}^+(\tau | 0, 2) \\ &\times \prod_{g_i=1}^2 \text{Dirichlet}(\rho_{g_i\cdot} | 1, 1) \prod_{g_j=1}^2 \text{Normal}(\alpha_0^{g_i g_j} | 0, 2) \prod_{n=1}^{13} \text{Normal}(\alpha_n^{g_i g_j} | 0, \tau),\end{aligned}$$

which is not easy to sample from directly. We appeal to Stan [7], a probabilistic programming language that implements Hamiltonian Monte Carlo to obtain samples from the posterior [2].

5 Results

We present degree and kernel mixing estimates from aggregated relational data (by asking survey respondents how many people they know in subpopulation \mathcal{G}_k) gathered from responses to questions where \mathcal{G}_k are twelve first names. We use data from the Contact Prevalence study ($N = 1,190$), a nationally representative cross-sectional survey of American adults aged 18 and older, which was designed following the recommendations outlined in McCormick et al. [5]. This survey

was administered in 2015 through GfK's Knowledge Panel (commonly known as Knowledge Networks). All data were collected online. When necessary, GfK provided Internet access to those who did not have it. The names used in the survey along with their mean and standard deviation, μ_{kg_j} and $\sigma_{kg_j}^2$, as defined in Eq. 3 are collected in Table 1. We choose names that are prevalent across a broad range of ages so that we can minimize barrier effects. Since each of the above subpopulations are specific to either males or females, however, only one value of g_j is defined for each subpopulation \mathcal{G}_k . Consequently, in the negative binomial expectation of Eq. 4, the first summation is taken over only one value of g_j for each subpopulation \mathcal{G}_k .

Table 1. The twelve subpopulations defined by first names with their μ_{kg_j} and $\sigma_{kg_j}^2$ as defined in Eq. 3. We use one value of g_j and take it to be the name's prevalent gender.

Female			Male		
Name	μ_{k1}	σ_{k1}	Name	μ_{k2}	σ_{k2}
Emily	28.4	23.4	Jacob	22.7	18.5
Stephanie	35.6	14.4	Kyle	25.6	10.8
Jennifer	37.4	10.6	Adam	31.2	16.6
Kimberly	39.8	13.0	Kevin	38.8	16.2
Karen	56.1	14.0	Mark	49.3	15.1
Linda	63.3	10.5	Bruce	62.5	16.8

Figure 3 displays the degree estimates obtained from fitting the kernel mixing model to the names responses, for six different age-sex groups. The pattern of degree increasing with age is surprising because we usually expect degree to decrease for older individuals. However, this may partially be explained by a combination of people retiring at increasingly older ages and in general being more active at older ages than in the past.

Figure 4 displays the kernel estimates for male and female egos of three different ages, each. Compared to the mixing matrix in Fig. 1, the kernels show more regularity in their structure from ego to ego (by design). Additionally, estimating kernel bandwidth as a function of age allows us to clearly see how the age concentration of alters changes by age. Meanwhile, the gender mixing matrix, while simpler than the age-gender mixing matrix, is still able to capture the variability in network composition by gender. In particular, it appears that female egos' social networks are roughly equally split between males (51.1%) and females (49.9%), but male egos' social networks contain more males (55.5%) than females (44.5%).

Figure 5 displays the kernel bandwidth spline estimates by ego and alter gender. Male egos' social network age distributions are more spread out (i.e. larger bandwidth) when they first enter the work force by their late 20s, and then the

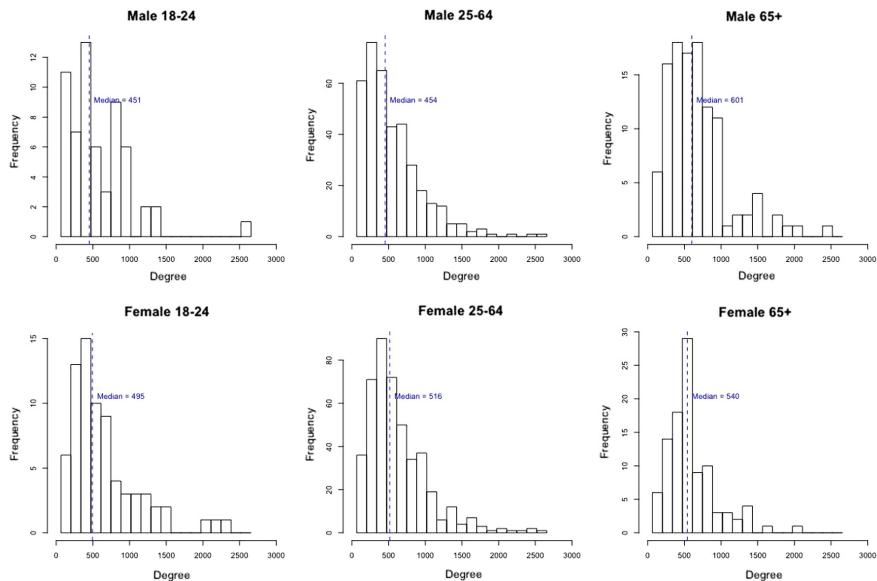


Fig. 3. Name-based degree estimates for both genders across three age groups. A pattern of degree increasing with age is clear.

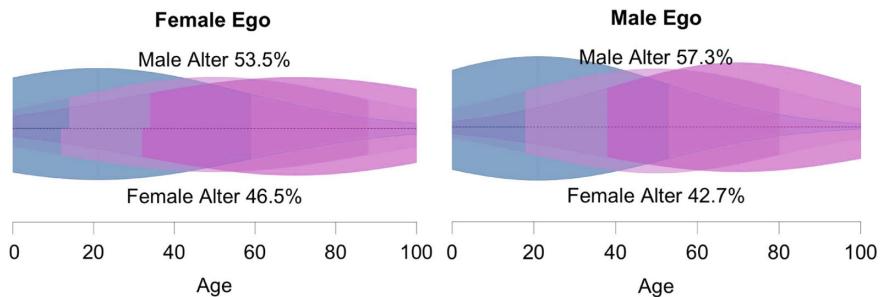


Fig. 4. Name-based kernel estimates for male and female egos of age 21, 38, and 70. The darker regions show one standard deviation of the kernel.

networks' age distributions gradually tighten as the egos get older. Female egos' social network age distributions, on the other hand, spread out slightly earlier on, and sharply decrease going into the 30s, perhaps due to childbirth. However, after age 40, their networks' age distributions spread out again, stabilizing in the 60s. Bandwidths between opposite sexes reflect a milder version of the social structure displayed between actors of the same sex.

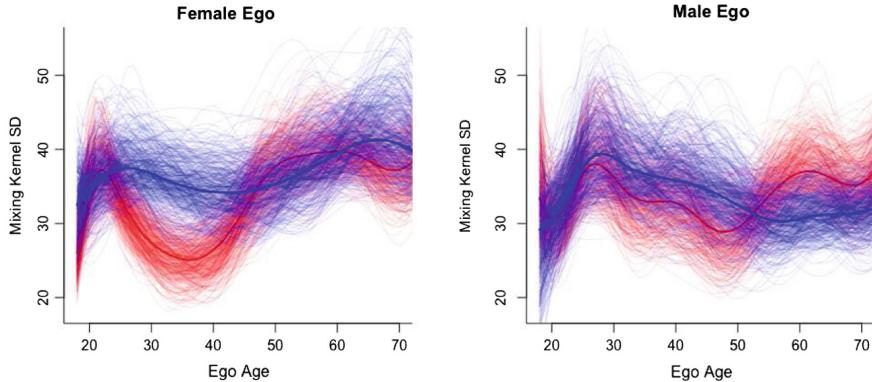


Fig. 5. Posterior draws of name-based kernel bandwidth splines for male and female egos. Blue draws correspond to male alters while red draws correspond to female alters. Individual draws are shown with 0.1 alpha while medians are shown with 1.0 alpha.

6 Discussion

This paper presents a latent kernel framework for social mixing using ARD. We show that, through the choice of models with appropriate structures, we can produce an interpretable representation of latent mixing patterns in survey data.

Recent work demonstrates that features of social mixing, such as homophily (tendency for actors to form ties with similar others), are distinguishable after aggregation. The latent kernel model, provides a structured, yet flexible, framework to estimate social mixing between individuals of any age and does not suffer from identifiability issues present in McCormick et al.'s model [5]. The addition of the kernel bandwidth spline provides further insights by estimating social mixing variability granularly over time.

The Gaussian kernel, although unimodal and symmetric, still provides room for future flexibility. Mixtures of Gaussian distributions, for example, would provide a more flexible representation of latent features and could provide additional insights into inter-generational mixing patterns, while still maintaining computational tractability. The bandwidth spline framework could also be applied to the gender mixing rates to provide insight into how gender mixing changes by age. At the same time, additional categorial variables could be used in place of gender. A spline framework applied to race could be used to understand social mixing rates between races over time, for example.

Acknowledgements. This research is supported by NSF grant SES 1023176.

References

1. De Boor, C.: A Practical Guide to Splines, vol. 27. Springer, New York (1978)
2. Gelman, A., Stern, H.S., Carlin, J.B., Dunson, D.B., Vehtari, A., Rubin, D.B.: Bayesian Data Analysis. Chapman and Hall/CRC (2013)
3. Killworth, P.D., Johnsen, E.C., McCarty, C., Shelley, G.A., Bernard, H.R.: A social network approach to estimating seroprevalence in the United States. *Soc. Netw.* **20**(1), 23–50 (1998)
4. McCarty, C., Killworth, P.D., Bernard, H.R., Johnsen, E.C., Shelley, G.A.: Comparing two methods for estimating network size. *Hum. Organ.* **60**(1), 28–39 (2001)
5. McCormick, T.H., Salganik, M.J., Zheng, T.: How many people do you know? Efficiently estimating personal network size. *J. Am. Stat. Assoc.* **105**(489), 59–70 (2010)
6. McPherson, M., Smith-Lovin, L., Cook, J.M.: Birds of a feather: homophily in social networks. *Ann. Rev. Sociol.* **27**(1), 415–444 (2001)
7. Stan Development Team: Stan modeling language: User's guide and reference manual (2016). Version 2.14.0. <http://mc-stan.org/>
8. Zheng, T., Salganik, M.J., Gelman, A.: How many people do you know in prison? Using overdispersion in count data to estimate social structure in networks. *J. Am. Stat. Assoc.* **101**(474), 409–423 (2006)



Forman's Ricci Curvature - From Networks to Hypernetworks

Emil Saucan^{1(✉)} and Melanie Weber^{2(✉)}

¹ ORT Braude College, 2161002 Karmiel, Israel
semil@braude.ac.il

² Princeton University, Princeton, NJ 08544, USA
mw25@math.princeton.edu

Abstract. Networks and their higher order generalizations, such as hypernetworks or multiplex networks are ever more popular models in the applied sciences. However, methods developed for the study of their structural properties go little beyond the common name and the heavy reliance of combinatorial tools. We show that, in fact, a geometric unifying approach is possible, by viewing them as polyhedral complexes endowed with a simple, yet, the powerful notion of curvature – the Forman Ricci curvature. We systematically explore some aspects related to the modeling of weighted and directed hypernetworks and present expressive and natural choices involved in their definitions. A benefit of this approach is a simple method of structure-preserving embedding of hypernetworks in Euclidean N -space. Furthermore, we introduce a simple and efficient manner of computing the well established Ollivier-Ricci curvature of a hypernetwork.

1 Introduction

Networks are popular means for modeling (pairwise) relationships between elements in a complex system. While the simplicity of such models allows for efficient representation and computational analysis, its ability to represent complex relations is limited. In the context of applications in the social and biological sciences, we often encounter relationships between groups of elements or systems with different types of elements that can participate in a specified set of relationship types. As an effective model for such higher-order relationships, hypernetworks have recently attracted increasing interest in the network science community. However, most network analysis tools are targeted to classic networks and do not readily transfer to the case of hypernetworks.

In the present work, we introduce geometric tools for the analysis of hypernetworks. We suggest a parametrization that allows for the application of curvature-based tools [13, 19] to hypernetworks, allowing for an efficient analysis of their structure and intrinsic geometry. The introduced methods are based on a discrete notion of Ricci curvature [7] whose applicability for network analysis and data science applications has recently been studied by the authors and collaborators for a wide range of network data [13, 14, 19–21].

We suggest representing hypernetworks as weighted cell complexes, more specifically as simplicial or polyhedral complexes. In this parametrization, faces encode relationships between groups of nodes; for example, triangles may represent relationships between triples of nodes. Note, that we distinguish between *pairwise* relationships, as encoded in the classic networks model, and *higher-order* relationships. For instance, tetrahedra represent higher-order relationships between quadruples of nodes, whereas quadrangles denote pairwise relationships between four nodes. For practical purposes, simplicial complexes are the most suitable parametrization, as they are easy to analyze and topologically robust [4, 11]. However, this simplified model falls short of capturing relationships among groups of more than three vertices. In [19] we demonstrate empirically, that such relationships occur frequently in real-world data sets, creating a need for higher-order models that incorporate these structures. We will, therefore, introduce a parametrization that represents hypernetworks as polyhedral complexes instead.

We have touched on these ideas and their significance for modeling complex systems in [19] and [12] and discussed some of its geometrical and topological implications. Here, we will extend these ideas and discuss (potential) applications.

We would like to thank Jürgen Jost and Areejit Samal for helpful discussions.

2 Hypernetworks as Polyhedral Complexes

In classic graphs, an edge connects exactly two nodes, i.e. it represents a *pairwise* relation. Here, we consider the more general notion of *hypernetworks* that allows for modeling relations between groups of nodes.

2.1 Parametrizing Hypernetworks as Polyhedral Complexes

Definition 1 (Hypernetworks). We define a *hypernetwork* as a *hypergraph* $\mathcal{H} = (\mathcal{V}, \mathcal{E})$ with the *node set* V partitioned into *type sets*, i.e. $\mathcal{V} = (V_1, \dots, V_p)$, and hyperedges $[(v_1, \dots, v_n), (v_1, \dots, v_m)] \in E$ connecting groups of nodes.

Since for $m = n = 1$, one returns to the classical case, it is natural to assign to each pair of nodes appertaining to a hyperedge a segment/edge. This observation gives rise to a natural parametrization of a hyperedge with m/n -dimensional simplices, forming a simplicial complex S . For instance, we can parametrize parent-child relationships as a directed complex (see Fig. 1). Note that the resulting complex is simplicial, if and only if we only allow connections between nodes of the same type. Otherwise, the resulting complex is polyhedral. Thus, hypernetworks can be parametrized as *bi-partite* simplicial or polyhedral complexes. A closely related model for hypernetworks is that of bipartite, directed graphs, albeit with the restriction that edges only connect nodes of the same

type. This model does not necessarily capture higher-dimensional (geometric) information. In the simplicial/polyhedral complex model, such information can be encoded in the weights of faces and higher order simplices. We will discuss such parametrizations in Sect. 2.2. Related work on such representations includes the *clique expansion* and the *star expansion* – see, e.g. [15]. An additional advantage of this model arises from the *structure-preserving* low-dimensional embedding of a simplicial complex with *indecomposable edges* (see [15]), that is not readily available for bipartite graphs. To understand this observation, let us introduce the following notation: Let $\dim(\mathcal{V}) = \max\{\dim(V) \mid V \in \mathcal{V}\}$, $\dim(\mathcal{E}) = \max\{\dim(E) \mid E \in \mathcal{E}\}$ denote the maximal dimension of the hyperedges, and define the *dimension* of the hypernetwork \mathcal{H} as $\dim(\mathcal{H}) = \max\{\dim(\mathcal{V}), \dim(\mathcal{E})\}$. We can now state the following

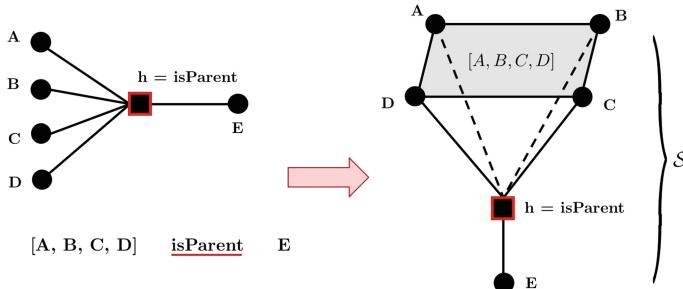


Fig. 1. Parametrization of hypernetworks as simplicial complexes. A child-parent relationship is modeled as a hyperedge h : A, B, C and D are parents of E (left). We represent the co-parent relationship as a face $[A, B, C, D]$ and represent h as a simplicial complex $\mathcal{S} = [A, B, C, D, E]$ (right).

Theorem 1. *Any hypernetwork has a structure-preserving embedding as an Euclidean complex in $\mathbb{R}^{2\dim(H)+1}$.*

Proof. By a classical result in PL-Topology [6, 16], any n -dimensional polyhedral complex admits a (combinatorial) embedding in \mathbb{R}^{2n+1} . Therefore, the hypernetwork H , viewed as a complex, admits an embedding in the lowest dimension in which all of its constituting components can be embedded, namely $\dim(H)$. \square

2.2 Choice of Weights

The problem of choosing appropriate weights of the variously dimensional faces has two different, yet complementary aspects. The first is empirical: The choice depends on the specific application to be modeled, and it is the resort of the practitioners to decide on the relevant features and information to be encoded. The second is the geometric aspect that studies the choice of weights from a purely mathematical viewpoint. In

this section, we discuss how to produce expressive weights for edges and higher dimensional faces from geometric and empirical information.

For *combinatorial weights*, i.e. for the standard allocation of weight one to each edge, one also canonically assigns weight equal to one to each face, in any dimension. We do not concentrate on this model here, since we consider it overly simplistic, given that it captures only the basic underlying topology and very little of the geometry of a (hyper)network.

For choosing edge weights in the *general case*, one can either settle for generalized weights in Forman's framework [7] or metricize the graph. More expressive metrics than the combinatorial one include the consideration of both edge weights viewed as abstractions of lengths and node weights as measures concentrated at the nodes. A number of choices are available, mainly the *path degree metric* [5] and the well known Wasserstein distance W_1 (see e.g. [17]).

The case of higher dimensional faces is more delicate, as uniqueness is usually not guaranteed. Our choice of weights introduced below is purely geometric and extends to any dimension without imposing thresholds. Related approaches that rely on thresholding (see, e.g., [11]) introduce additional degrees of freedom in the choice of the threshold value.

For 2-dimensional faces in simplicial complexes (triangles), the obvious geometric choice is that of the area. Moreover, since such complexes are usually envisioned as *piecewise flat*, i.e. composed of Euclidean triangles, etc., in a metrized graph, the area (weight) of any face T can be easily expressed as a function of the weights (lengths) of the edges, via the *Heron Formula*:

$$w_2(T) = \sqrt{p(p - w_1)(p - w_2)(p - w_3)}, \quad (1)$$

where $p = (w_1 + w_2 + w_3)/2$, and where w_1, w_2, w_3 represent the weights (lengths) of the edges. However, piecewise linear (*PL*) does not necessarily mean piecewise *flat* and, indeed network embeddings in Hyperbolic space have been considered [2, 10], wherein hyperbolic 2-simplices represent the faces. However, the area of a hyperbolic triangle is, as opposed to the Euclidean one, a function of the *angle defect*, i.e., $S(T) = \pi - (\alpha + \beta + \gamma)$, for any triangle T of angles α, β, γ . Therefore, a simple expression as a function of the are cannot be constructed. We suggest the following angle-based expression (see, e.g., [9]):

$$w_2(T_{hyp}) = 2 \arcsin \frac{1}{2} \frac{\sqrt{\sinh p \sinh(p - w_1) \sinh(p - w_2) \sinh(p - w_3)}}{\cosh \frac{w_1}{2} \cosh \frac{w_2}{2} \cosh \frac{w_3}{2}}. \quad (2)$$

Spherical embeddings are used more rarely, but a corresponding formalism can be derived using the following relation from spherical geometry:

$$w_2(T_{sph}) = 2 \arcsin \frac{1}{2} \frac{\sqrt{\sin p \sin(p - w_1) \sin(p - w_2) \sin(p - w_3)}}{\cos \frac{w_1}{2} \cos \frac{w_2}{2} \cos \frac{w_3}{2}}. \quad (3)$$

Clearly, for more general faces, one can use their decomposition into triangles, and the additivity and unicity of the area function.

For general weights, simpler, combinatorial choice, such as $w_2(T) = w_1 + w_2 + w_3$ or $w_2(T) = w_1 w_2 w_3$, or their scaled version (to dimensionally correspond to area) $w_2(T) = (w_1 + w_2 + w_3)^2$ (or $w_2(T) = (w_1 w_2 w_3)^{2/3}$, respectively) may be employed. An example of such weights can be found in [11]. In this case, one can not use the decomposition into triangles to define the area of p -gons, with $p \geq 4$, but rather use a direct extension of the idea above, e.g. for a quadrilateral Q , define its weight as $w_2(AQ) = w_1 + w_2 + w_3, w_4$, etc., where the w_i are defined as before. This type of abstract faces weights (and their higher-dimensional counterparts – see below) can be trivially generalized to the directed hypernetworks case.

The choice of weight $w_m(\tau)$ for an m -dimensional ($m \geq 3$) Euclidean simplex τ of vertices v_0, \dots, v_m and edge lengths $d_{ij}; 0 \leq i, j \leq m$ arises naturally from the *Cayley-Menger determinant*:

$$D(\tau) = D(v_0, v_1, \dots, v_m) = \begin{vmatrix} 0 & 1 & 1 & \dots & 1 \\ 1 & 0 & d_{01}^2 & \dots & d_{0m}^2 \\ 1 & d_{10}^2 & 0 & \dots & d_{1m}^2 \\ \dots & \dots & \dots & \dots & \dots \\ 1 & d_{m0}^2 & d_{m1}^2 & \dots & 0 \end{vmatrix}, \quad (4)$$

given the fact that

$$\text{Vol}^2(\tau) = \frac{-1^{k+1}}{2^k (k!)^2} D(\tau); \quad (5)$$

(see [3]). Thus one can either use $w_m(\tau) = \text{Vol}(\tau)$ or simply put $w_m(\tau) = D(\tau)$.

This approach extends, like Heron's formula, to spherical and hyperbolic simplices, respectively, (see [3]):

$$\Delta(\tau) = \Delta(v_0, \dots, v_n) = \det(\cos d_{ij}), 0 \leq i, j \leq m.; \quad (6)$$

$$\mathcal{D}(\tau) = \mathcal{D}(v_0, \dots, v_n) = \det(\cosh d_{ij}); 0 \leq i, j \leq m. \quad (7)$$

Again, one can either use the volume formula from each of the last two formulas, or simply set $w_m(\tau) = \Delta(\tau)$ (resp. $w_m(\tau) = \mathcal{D}(\tau)$).

In the case when one does not wish to proceed through the metrization process of the abstract, given weights, one can again introduce combinatorial weights, such as $w_{01} + \dots + w_{m-1,m}$, or $w_1 \dots w_{m-1,m}$; where $w_{0,1} \dots w_{m-1,m}$ denote the weights of the edges $e(v_0, v_1)$, etc.

2.3 Directionality

Directionality (or *orientations*) of associations (interactions or commonalities) between elements is an important feature of complex data sets and hence has to be incorporated into network analysis tools. When analyzing higher-order relations, this even more important than in pairwise once, as it determines which complexes are “filled in”, i.e., considered in the analysis.

To make this more clear, we start with the basic and most important case of triangles. In this case, there are two possible sets of orientations

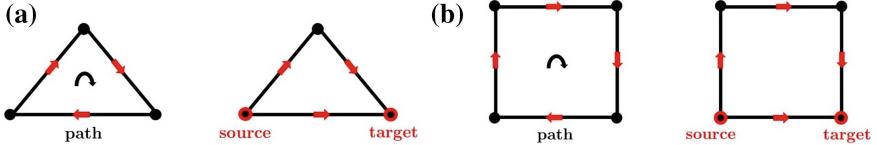


Fig. 2. Orientations on the edges in **a** 3-faces (triangles) and **b** 4-faces (quadrangles). The edges define either a cycle (i.e., a closed path) or an acyclic path with a unique “source” and “sink” (see Fig. 2a).

on the edges: Either the edges define a cycle (closed path), or there is a unique “source” and “sink” (see Fig. 2a). One can distinguish between the two cases by observing that in the acyclic case the information flow is disrupted, whereas in the cyclic case it is not. This approach generalizes readily to higher order p -gons (see Fig. 2 and its accompanying caption). From a purely geometric viewpoint, the “correct” choice is that of cycles. It has the advantage of allowing for the standard extension to higher dimensional simplices, e.g. tetrahedra, by filling-in only those with cyclic faces, while requiring, that adjacent faces have opposite orientations. While this process inductively extends to all dimensions, it clearly drastically reduces the number of oriented simplices considered in the analysis. In a practical context, the second option is more intuitive as it relates to information flow in the network (see, e.g. [4, 11]). If one is interested in modeling the information flow in the network, one has to fill in only faces that form acyclic paths with a unique source and target. This is illustrated in Fig. 3. Note, that for computing the Forman-Ricci curvature of a cell, one has to ascertain that its parents and children are uniquely defined (cf. Eq. 8). Thus one has to extend the genealogical correlation of nodes/faces to the directed case, see Fig. 3.

3 Discrete Curvatures for Hypernetworks

3.1 Forman-Ricci Curvature

In this section, we will introduce the Forman Ricci curvature $\text{Ric}_F(h)$ for hyperedges h . The curvature bounds, more so than the actual values, give insight into the structure of the hypernetwork. With the formalism introduced below, we compute global curvature bounds

$$\text{Ric}_m^E = \min\{\text{Ric}(E) \mid E \in \mathcal{E}\} \quad \text{and} \quad \text{Ric}_M^E = \max\{\text{Ric}(E) \mid E \in \mathcal{E}\},$$

which characterize (global) geometric and topological properties of the hypernetwork.

Of course, one could also compute the “graph” Forman curvature $\mathbf{F}(e)$ (i.e., the 1-dimensional, edge-based notion; see, e.g., [19]) for every edge in the complex parametrization and fit lower and upper bounds respectively:

$$\mathbf{F}_m^E = \min\{\mathbf{F}(E) \mid E \in \mathcal{E}\} \quad \text{and} \quad \mathbf{F}_M^E = \max\{\mathbf{F}(E) \mid E \in \mathcal{E}\}.$$

These represent bounds for the information flow along the hyperedges. While this approach is computationally much simpler, it loses some of the dimensional information (encoded in the higher-dimensional relationships).

Thus one can employ Forman's Ricci curvature at two levels: Either at a small scale as the Forman curvature of the simplices (polyhedra) that represent the hyperedges of the network, or the "graph" Forman curvature $\mathbf{F}(e)$ of all edge in the network for an analysis of the coarse geometry. The two methods, used in tandem, should yield the most comprehensive understanding of a hypernetwork.

Recall, that Forman's Ricci curvature for cell complexes is given as follow [7]:

Definition 2 (Forman-Ricci curvature for weighted cell complexes). For each p -cell α we define

$$\begin{aligned} \mathcal{F}(\alpha^p) = \omega(\alpha^p) & \left[\left(\sum_{\beta^{p+1} > \alpha^p} \frac{\omega(\alpha^p)}{\omega(\beta^{p+1})} + \sum_{\gamma^{p-1} < \alpha^p} \frac{w(\gamma^{p-1})}{w(\alpha^p)} \right) \right. \\ & \left. - \sum_{\alpha_1^p \parallel \alpha^p, \alpha_1^p \neq \alpha^p} \left| \sum_{\substack{\beta^{p+1} > \alpha_1^p \\ \beta^{p+1} > \alpha^p}} \frac{\sqrt{\omega(\alpha^p)\omega(\alpha_1^p)}}{\omega(\beta^{p+1})} - \sum_{\substack{\gamma^{p-1} < \alpha_1^p \\ \gamma^{p-1} < \alpha^p}} \frac{\omega(\gamma^{p-1})}{\sqrt{\omega(\alpha^p)\omega(\alpha_1^p)}} \right| \right]; \end{aligned} \quad (8)$$

where $\alpha < \beta$ indicates, that α is a *parent* of β and β a *child* of α , and $\alpha_1 \parallel \alpha_2$ that the simplices α_1 and α_2 are *parallel*. For details, see [18] and the references therein.

Weights are usually chosen to resemble intuitive geometric notions, such as length and volume. In practice, weights can often be constructed to

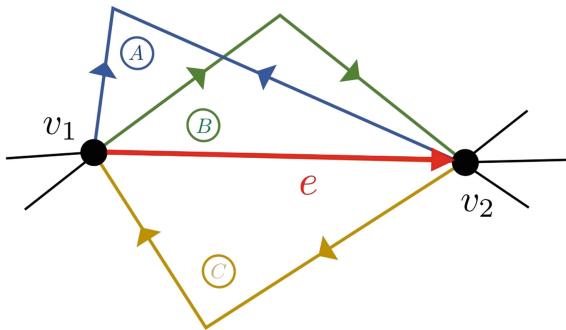


Fig. 3. When considering the information flow within the network, we have to decide which faces to "fill in", i.e., which oriented triangles are relevant for our analysis. In the context of information flow within the system, one is interested in the parallel routing of information from a source to a target, therefore one should consider only triangles of form (B). However, if one is interested in the collection of information at the source and its spreading at the target, then both "(B)" and "(C)" triangles should be considered since they contribute to the information flow along the edge e . In contrast, triangle "(A)" does not contribute to the information flow, so it should be discarded.

incorporate meta-information. For instance, in a word network built from co-occurrences, the co-occurrence frequencies can be implemented in the edge weights. For the case of hypernetworks, we are interested in computing Forman's curvature for each hyperedge. Given a hyperedge h , let us denote the set of *left neighbors* $V_l(h)$ as the set of all vertices in incoming direction with respect to h , and $V_r(h)$ as the set of *right neighbors* in the outgoing direction, respectively (see Fig. 1). Then, one can define the Forman-Ricci curvature as the total amount of "information flow" through h :

Definition 3 (Forman's curvature for hyperedges). Let $h = (\{v_r^i\}, \{v_l^j\})$ be a hyperedge with left neighbors $V_l(h) = \{v_l^i\}$ and right neighbors $V_r(h) = \{v_r^j\}$.

$$\text{Ric}_F(h) = \text{Ric}_{F,V_1(h)}(h) - \text{Ric}_{F,V_r(h)}(h),$$

where $\text{Ric}_{F,V_1(h)}$ measures curvature in the incoming and $\text{Ric}_{F,V_r(h)}$ in the outgoing direction. For the incoming direction, we compute the Forman-Ricci curvature over the simplex $\mathcal{S} = [\{v_l^i\}, h]$ (see Fig. 1) as

$$\begin{aligned} \text{Ric}_{F,V_1(h)}(h) &= \omega \left(\left[v_1^1, \dots, v_l^{|V_l(h)|} \right] \right) \left(\frac{\omega \left(\left[v_1^1, \dots, v_l^{|V_l(h)|} \right] \right)}{\omega(\mathcal{S})} + \sum_{\substack{u, v \in V_l(h) \\ u \sim v}} \frac{\omega(e = (u, v))}{\omega \left(\left[v_1^1, \dots, v_l^{|V_l(h)|} \right] \right)} \right. \\ &\quad - \sum_{\substack{u, v \in V_l(h) \\ u \sim v}} \frac{\sqrt{\omega(\Delta_{u, v, h}) \omega \left(\left[v_1^1, \dots, v_l^{|V_l(h)|} \right] \right)}}{\omega(\mathcal{S})} \\ &\quad \left. + \sum_{\substack{u, v \in V_l(h) \\ u \sim v}} \frac{\omega(e = (u, v))}{\sqrt{\omega(\Delta_{u, v, h}) \omega \left(\left[v_1^1, \dots, v_l^{|V_l(h)|} \right] \right)}} \right). \end{aligned}$$

An analog expression can be derived for the outgoing direction.

3.2 Ollivier's Ricci Curvature

We have focused on Forman's Ricci curvature so far, due to the intuitive geometric definition that makes it ideally suited for our parametrization of (hyper)networks as polyhedral complexes. In addition, its computational efficiency makes it ideally suited for large-scale applications. However, one is naturally interested in extending Ollivier's Ricci curvature (see, e.g. [17]) from networks (where it has become an important tool and subject of research) to hypernetworks. Unfortunately, generalizing the notion of optimal transport from node to node along edges to higher dimensional faces is far from straightforward. Recently, a new type of curvature, the *coarse scalar curvature*, was introduced [1] as a first extension of Ollivier's methods to hypernetworks. We show that, based on the parametrization of hypernetworks as polyhedral complexes, it is possible to extend Ollivier's curvature to hypernetworks in a more general setting.

This extension is based on a simple, geometric idea: We pass from complex parametrization to its dual complex (see, e.g. [8]). For instance,

for each two-dimensional face we can construct a corresponding (dual) node, and two such nodes are connected by an edge if their corresponding faces are adjacent in the complex. The volume (weight) of the original face is then chosen as the weight of the (dual) node.

The basic idea is quite simple and natural, from a geometric viewpoint (and common in Topological Graph Theory and its applications). Namely, one passes to the dual of the polyhedral complex (network) (see. e.g. [8]). For instance, to each top-dimensional face, there corresponds a node, and two such nodes are connected by a (dual) edge if their corresponding faces are adjacent in the given complex/hypernetwork. The measure (e.g. volume) of the original face/simplex is concentrated in the weight of the node. In most instances, the mass transport of interest is between the highest dimensional faces, but this approach can be applied for lower dimensional ones as well, and, after some adaptation, to transport between faces of different dimensions.

More precisely, if c_1^k, c_2^k are two adjacent k -cells of weights w_1^k, w_2^k , let us denote their barycenters, i.e., their *dual* nodes, by $n_1 = n_1^0, n_2 = n_2^0$ and a *dual* edge (n_1, n_2) connecting them by $e_{1,2}$. Furthermore, let $w_{12} = w^1(e_{12})$ be the weight of the edge e_{12} . To the dual nodes n_1, n_2 we assign the weight of the cells c_1, c_2 , respectively, i.e. $w_1^0 = w(n_1^0) = w_1^k$ and $w_2^0 = w(n_2^0) = w_2^k$. There exist a number of possible choices for the (dual) edge weights. The simplest is a combinatorial weight of 1, whereas the most natural one from a geometric perspective is $w_{12} = w_{12}^{k-1}$, where w_{12}^{k-1} represents the weight of f_{12}^{k-1} – the $(k-1)$ -dimensional face common to c_1 and c_2 . A further option is that of the Euclidean distance between the barycenters of the original faces in an \mathbb{R}^N -embedding of the hypernetwork (see also Sect. 2).

The (coarse) Ollivier curvature [22] of the edge e_{12} is then given by

$$\kappa(e_{12}) = \kappa(n_1, n_2) = 1 - \frac{W_1(n_1, n_2)}{d(n_1, n_2)}; \quad (9)$$

where $W_1(n_1, n_2)$ represents the Wasserstein distance between the nodes n_1, n_2 , which in the discrete case at hand is expressed as

$$W_1(w'_1, w'_2) = \inf_{\mu_{n_1, n_2} \in \prod(w_1, w_2)} \sum_{(m, n) \in V \times V} d(m, n) \mu_{n_1, n_2}(m, n). \quad (10)$$

Here, $\prod(w_1, w_2)$ is the set of probability measures μ_{n_1, n_2} that satisfy

$$\sum_{n \in V} \mu_{n_1, n_2}(m, n) = w_1(m), \quad \sum_{m \in V} \mu_{n_1, n_2}(m, n) = w_2(n), \quad (11)$$

where w'_1, w'_2 represent the *normalized* weights corresponding to the vertices n_1, n_2 . Here d denotes the metric corresponding to the choice of edge weights above.

4 Applications

4.1 Applications in Network Analysis

One of the main advantages of Forman's Ricci curvature resides in the fact that it reveals important algebraic-topological properties of multiplex networks [12]. Moreover, as we have argued above, these benefits

extend to hypernetworks, thus allowing the exploration of the structure of such networks, in a manner akin to that of Persistent Homology. More precisely, positive Forman Ricci curvature implies that the first homology group $H_1(\mathcal{N}, \mathbb{R})$ vanishes. In fact, if Forman Ricci curvature is strictly positive, the fundamental group $\pi_1(\mathcal{N})$ is finite. This capability of determining properties of the homotopy, and not just the homology of a hypernetwork, represents an advantage of our approach over Persistent Homology. Note that both properties above generalize, essentially, to any dimension n , by passing to the higher-dimensional curvatures generalizing Ricci curvature and introduced by Forman [7].

Yet the benefits of viewing hypernetworks as polyhedral complexes and, in Forman's and Ollivier's Ricci curvature are not restricted to the detection of their core topological structure. Exploring the two curvatures in different dimensions allows for the understanding and classification of various types of networks, especially the differentiation of artificial versus natural hypernetworks. To these advantages, we have already added that of naturally producing structure-preserving embeddings into Euclidean space, which allow for a simpler and more intuitive understanding of these structures.

4.2 Use Case

We want to give a brief illustration of the geometric tools introduced in Sect. 3. For this, we computed the Forman-Ricci for a hypernetwork that models historic relationships between object-oriented programming languages (see Fig. 4). We parametrize the hypernetwork as polyhedral complexes (Sect. 2). In the unweighted case, the Forman-Ricci curvature

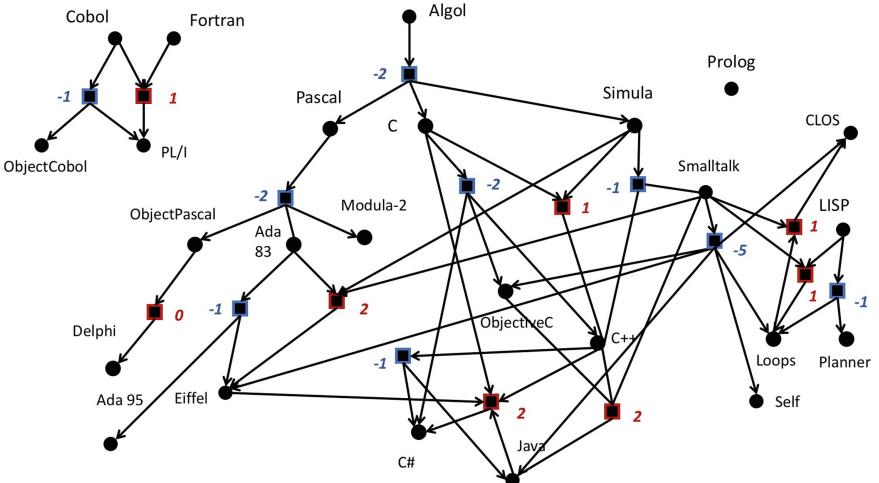


Fig. 4. Family tree of (object-oriented) programming languages as hypernetwork. Hyperedges that represent *co-child* relationships are marked in blue, *co-parent* relationships in red. Each hyperedge is labeled with its respective curvature value.

reduces to a function of the *in* and *out degree* (i.e., $\text{Ric}_F(h) = \deg_{IN}(h) - \deg_{OUT}(h)$). This resembles a combinatorial measure of the information flow through the hyperedge.

5 Conclusions

We introduced a parametrization of hypernetworks as polyhedral complexes that allows for studying higher order relations in complex systems with network-theoretic tools. Furthermore, we discuss discrete notions of curvature in the context of this parametrization and give an explicit formalism for computing these in practice. For the Forman-Ricci curvature, we demonstrate the computation on a small use case.

The present work is merely an exposition of theoretical aspects. Future directions include the empirical explorations of the introduced concepts, in particular, an empirical study of larger datasets from different branches of network science. Moreover, we would like to investigate systematically, how curvature-based analysis of hypernetworks can give insight into the coarse geometry and therefore global structural patterns of such objects.

For this, we plan to systematically explore different type of weights and faces of varying dimension. This should include a statistical comparison of the results, to ascertain their relative efficiency, especially in comparison with other invariants and network characteristics. We also plan to further investigate the commonalities and differences for the approaches to Ricci curvature that we introduced, i.e. (i) 'graph' Forman curvature and 'complex' Forman curvature and (ii) Forman- and Ollivier-Ricci curvature for hypernetworks. In practice, Forman's curvature will likely be more applicable and scalable, due to its simple notion that avoids computationally expensive tasks, such as the computation of Wasserstein distances.

References

1. Asoodeh, S., Gao, T., Evans, J.: Curvature of Hypergraphs via Multi-Marginal Optimal Transport [arXiv:1803.08584v1](https://arxiv.org/abs/1803.08584v1) [cs.IT] (2018)
2. Bianconi, G., Rahmede, C.: Emergent hyperbolic network geometry. Sci. Rep. **7**, 41974 (2017)
3. Blumenthal, L.M.: Distance Geometry - Theory and Applications. Clarendon Press, Oxford (1953)
4. Courtney, O.T., Bianconi, G.: Dense power-law networks and simplicial complexes. Phys. Rev. E **97**, 052303 (2018)
5. Dodziuk, J., Kendall, W.S.: Combinatorial Laplacians and isoperimetric inequality. From local times to global geometry, control and physics. Pitman Res. Notes Math. Ser. **150**, 68–74 (1986)
6. Flores, A.: Über n-dimensionale Komplexe, die im R^{2n+1} absolut selbstverschlungen sind. Ergeb. Math. Kolloq. **34**, 4–6 (1933)
7. Forman, R.: Bochner's method for cell complexes and combinatorial Ricci curvature. Discrete Comput. Geom. **29**(3), 323–374 (2003)

8. Hudson, J.F.: Piecewise Linear Topology. Benjamin, New York (1969)
9. Janson, S.: Euclidean, spherical and hyperbolic trigonometry. Notes (2015)
10. Krioukov, D., Papadopoulos, F., Kitsak, M., Vahdat, A., Boguna, M.: Hyperbolic geometry of complex networks. *Phys. Rev. E* **82**, 036106 (2010)
11. Reimann, M.W., et al.: Cliques of neurons bound into cavities provide a missing link between structure and function front. *Comput. Neurosci.* (2017). <https://doi.org/10.3389/fncom.2017.00048>
12. Saucan, E., Jost, J.: Network topology versus geometry: from persistent homology to curvature. In: Proceedings of NIPS LHDS 2016 (2017)
13. Saucan, E., Samal, A., Weber, M., Jost, J.: Discrete curvatures and network analysis. *MATCH Commun. Math. Comput. Chem.* **20**(3), 605–622 (2018)
14. Sreejith, R.P., Vivek-Ananth, R.P., Jost, J., Saucan, E., Samal, A.: Discrete Ricci curvatures for directed networks, preprint (2018)
15. Tu, K., Cui, P., Wang, X., Wang, F., Zhu, W.: Structural deep embedding for hyper-networks, In: CoRR, vol. abs/1711.10146 (2017)
16. Van Kampen, E.R.: Komplexe in euklidischen Räumen. *Abhandlungen aus dem Mathematischen Seminar der Universität Hamburg.* **9**(1), 72–78 (1933)
17. Villani, C.: Optimal Transport: Old and New. Springer, Berlin-Heidelberg (2009)
18. Weber, M., Saucan, E., Jost, J.: Characterizing complex networks with Forman-Ricci curvature and associated geometric flows. *J. Complex Netw.* **5**(4), 527–550 (2017)
19. Weber, M., Saucan, E., Jost, J.: Coarse geometry of evolving networks. *J. Complex Netw.* (2017). <https://doi.org/10.1093/comnet/cnx049>
20. Weber, M., Jost, J., Saucan, E.: Forman-Ricci flow for change detection in large dynamics data sets. *Axioms* **5**(4) (2016). <https://doi.org/10.3390/axioms5040026>.
21. Weber, M., Stelzer, J., Saucan, E., Naitsat, A., Lohmann, G., Jost, J.: Curvature-based methods for brain network analysis (2017). [arXiv:1707.00180](https://arxiv.org/abs/1707.00180)
22. Ollivier, Y.: Ricci curvature of Markov chains on metric spaces. *J. Funct. Anal.* **256**(3), 810–864 (2009)



Mapping Structural Diversity in Networks Sharing a Given Degree Distribution and Global Clustering: Adaptive Resolution Grid Search Evolution with Diophantine Equation-Based Mutations

Peter Overbury¹, István Z. Kiss², and Luc Berthouze¹(✉)

¹ Department of Informatics, University of Sussex, Brighton, UK
{po36, L.Berthouze}@sussex.ac.uk

² Department of Mathematics, University of Sussex, Brighton, UK
i.z.kiss@sussex.ac.uk

Abstract. Methods that generate networks sharing a given degree distribution and global clustering can induce changes in structural properties other than that controlled for. Diversity in structural properties, in turn, can affect the outcomes of dynamical processes operating on those networks. Since exhaustive sampling is not possible, we propose a novel evolutionary framework for mapping this structural diversity. The three main features of this framework are: (a) subgraph-based encoding of networks, (b) exact mutations based on solving systems of Diophantine equations, and (c) heuristic diversity-driven mechanism to drive resolution changes in the MapElite algorithm. We show that our framework can elicit networks with diversity in their higher-order structure and that this diversity affects the behaviour of the complex contagion model. Through a comparison with state of the art clustered network generation methods, we demonstrate that our approach can uncover a comparably diverse range of networks without needing computationally unfeasible mixing times. Further, we suggest that the subgraph-based encoding provides greater confidence in the diversity of higher-order network structure for low numbers of samples and is the basis for explaining our results with complex contagion model. We believe that this framework could be applied to other complex landscapes that cannot be practically mapped via exhaustive sampling.

1 Introduction

Almost any complex system involving the interaction of constituent components can be represented as a network and networks have become a paradigm of choice for modelling and analysing such systems. It is now well known that node-level and structural properties of networks (e.g., degree-distribution, assortativity,

clustering, modularity) can fundamentally affect the way the system operates [17, 22, 26, 28]. Clustering, in particular, has been the subject of much work with both empirical and analytical results [6, 11, 34]. However, there is also some growing awareness that local structure (e.g., subgraph composition) may also have an important impact on dynamics [13, 15, 30].

To be able to show this more compellingly, there is a need for methods to sample the space of networks satisfying set constraints, e.g., degree sequence, assortativity, global clustering. Currently available network generative methods can be categorised in terms of where they fall within the ‘one-shot’ - ‘growing/developmental’ spectrum (see [2] for a more comprehensive treatment). On the ‘one-shot’ end of the spectrum, an algorithm produces a single network. One of the most popular example of such generative models is the exponential random graphs model [31] whereby one assumes that links are random variables and each realisation comes from a probability distribution of graphs of a given number of nodes. Variants of such models that provide more control over relationships between nodes include simplicial network models [3] and graphons [19]. At the other end of the spectrum, one can find methods that fundamentally involve rewiring existing networks and are typically based on Markov Chain Monte Carlo processes [5]. To construct networks with fixed degree sequence and global clustering coefficient, BigV [14] starts from a random network and performs a series of degree-preserving rewiring operations which increase clustering. The process is repeated until the desired clustering coefficient is achieved. Conversely, dk-series decomposition [24] uses rewiring to generate randomised versions of a given network that preserve network characteristics from average degree ($k = 0$) to global clustering coefficient ($k = 2.1$). In principle, rewiring approaches could be used to sample the network space, however, they do not actually provide any control over which local higher-order structure property is being changed. Further the question remains of whether these approaches will necessarily cover the full range of possible networks depending on the seed network [24]. Two common features to both approaches are that (a) there is great computational cost to mapping the network space and (b) they do not lend themselves to controlling/assessing the make-up of networks beyond the specified characteristics.

An alternative approach is therefore to not attempt to being as exhaustive as possible but rather to maximise the diversity of networks found within a given amount of time. This can be couched in terms of a multi-objective optimisation problem. In the kind of scenario we consider, population-based algorithms (see [10, 35] for reviews, and also [18]) can prove particularly helpful. More specifically, there has been a growing body of research into so-called quality diversity (or illuminative) algorithms, whose focus is to discover both quality and diversity at the same time, see [4, 27] for example. In the MapElite approach [21], the space of features (or behaviours) an individual might possess is divided into cells that act as niches to the population, forcing new individuals to only compete with individuals in the same cell. As a result, only the fittest (the elite) in each cell remain in the population, thus providing a collection of diverse, high-performing individuals. In previous work [25], we combined the MapElite method with CMA, the cardinality matching algorithm [30]. CMA breaks the problem of

generating networks down to the subgraph level (subgraphs being small structures) arranging set populations of these subgraphs in such a way to satisfy a prescribed degree distribution and global clustering coefficient. Although our method elicited a wide range of diversity at both structural and behaviour levels compared with other methods of network generation, it suffered from being very slow and ineffective at producing large pools of networks.

In this paper, we substantially improve the method through two major changes:

1. an exact Diophantine equation-based *mutation* method that guarantees that all individuals in the population are fit, i.e., they satisfy the constraints,
2. an adaptive resolution mechanism whereby the size of a niche changes during evolution in response to the level of variation between individuals. This allows us to efficiently control the trade-off between coverage and diversity.

In what follows, we detail both mechanisms after a brief reminder of how networks are encoded.

2 Methods

2.1 Defining the Search Space: Network Encoding

Manipulating diversity in higher-order structure whilst maintaining degree sequence and global clustering coefficient requires a parametrisation of networks (and a generative mechanism) that is both parsimonious enough that it enables systematic exploration and complete enough that it allows explicit control of global features. As in [25], we encode networks in terms of the population counts of each subgraph in an arbitrarily chosen family of subgraphs, e.g., $\{\triangle, \square, \boxtimes, \boxdot\}$, provided this family contains at least one clustering-inducing subgraph. To generate networks from a sequence of subgraph counts, we use the cardinality-matching algorithm (CMA) [30]. It is important to stress at the outset that the algorithm is not exact: first, to mitigate the the combinatorial complexity of satisfying all constraints, it is necessary to specify a fraction of edges not accounted for by the subgraphs (free edges); second, as described in [30], the allocation process (particularly when free edges are involved) can lead to by-products. For example, the addition of a free edge can lead to two distinct \triangle turning into one \square and one \triangle . Figure 1 illustrates this problem by showing that whilst CMA yields fairly good control over the \boxtimes , there is more noise for \triangle and \square (note that by-products of non clustering-inducing subgraphs is not an issue since they do not have any impact over the clustering coefficient). Nevertheless, the right-hand side panel in Fig. 1 demonstrates that despite the by-products, the process yields acceptable control over the global clustering coefficient with the obtained clustering values never exceeding the target value by more than 0.003, i.e., at most 21 \triangle in a regular network of size $N = 1000$ and degree $k = 7$.

2.2 Defining Movement Within the Search Space: Exact Mutations

In a standard evolutionary framework, mutations involve random changes in the make-up of each individual. Here, such approach is extremely wasteful because preserving the global properties of the network impose constraints on what changes are possible in one or more subgraphs given a change in another. For example, a constant global clustering coefficient imposes that the addition of one \square come with the loss of two \triangle . However, simply reducing the number of \triangle by 2 does not suffice because such operation would leave a deficit of 1 edge at network level. In this paper, we cast the problem of identifying degree- and clustering-preserving mutations (exact mutations, thereafter) in terms of solving a Diophantine problem, i.e., finding the integer solutions to an undetermined system of linear equations. Formally, an exact mutation is an integer solution of the system $A\mathbf{x} = b$ where: \mathbf{x} is a column vector of n rows and specifies the change in the number of each of the subgraphs specifying the network (i.e., n is the cardinality of the family of subgraphs used to parameterise the networks; $n = 5$ throughout the paper); A and b are a $3 \times n$ matrix and a column vector of 3 rows, specifying the 3 constraints that a mutation \mathbf{x} must satisfy, namely: (i) the change in the total number of triangles in the network must be 0, (ii) the change in the total number of edges in the network must be 0, (iii) the size of the change for the subgraph count(s) being mutated has the required size (see below). Note that the third constraint is purely for programming convenience as only the first two rows specify constraints between subgraphs. To illustrate the principle, given individual ($\boxtimes:61$, $\triangle:283$, $\square:110$, $\diamond:142$, $\square:87$) and a required mutation of size 2 in the number of \square , a possible vector \mathbf{x} is ($\boxtimes:-1$, $\triangle:0$, $\square:-1$, $\diamond:0$, $\square:2$) leading to the new network specification ($\boxtimes:60$, $\triangle:283$, $\square:109$, $\diamond:142$, $\square:89$). It is easily verified that the gain of four \triangle via the addition of two \square is compensated by the loss of one \boxtimes , whereas the resulting excess of 4 edges ($2 \times 5 - 1 \times 6$) is absorbed by the loss of one \square .

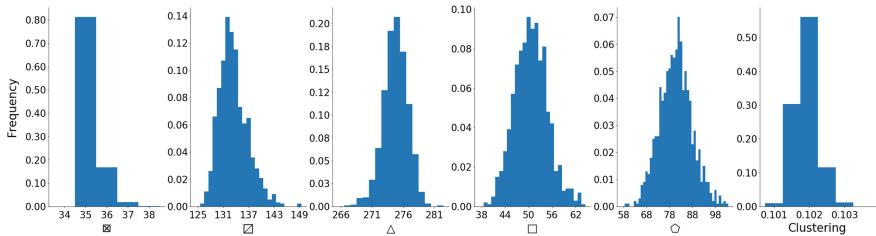


Fig. 1. First five panels: Histograms of the prevalence of each subgraph for 1000 networks generated by CMA for specification ($\boxtimes:35$, $\square:128$, $\triangle:277$, $\square:35$, $\diamond:42$). The subgraphs were counted using the method in [29]. The number of \triangle denotes the number of \triangle not involved in any other clustering-inducing subgraphs. Last panel: Histogram of global clustering coefficient values [16] for the 1,000 networks considered. The target global clustering coefficient was 0.1.

Solving an underdetermined system of Diophantine equations in general is a hard problem, however, finding solutions with the lowest Euclidean norm is easier [12]. We used the following implementation: <http://github.com/tclose/Diophantine>. There are two implications to this. The first is that it makes an exhaustive search impossible (although this is not strictly the aim of this work). The second is that solutions tend to be homogeneous (with little difference between absolute value of the components of the solution), which significantly biases how the space of solutions is sampled. For this reason, we built a catalog of solutions by systematically enforcing values for each component of the solution vector. In the experiments that follow, the catalog of possible mutations for 7-regular networks with global clustering of 0.1 comprised 1582 exact mutations with mutation sizes ranging from 1 to 128 and involving between 1 and 3 subgraphs.

There are two additional observations to be made. First, because all computations are integer, given a particular family of subgraphs, some mutations are not possible (i.e., the solver returns no solutions). A trivial example is that given a family where the only clustering-inducing subgraphs are \triangle , \square and \square , it is not possible to mutate the number of triangles by an odd number. Second, even if there is a solution, there is no guarantee that the network thus specified will be graphical [8] or realisable (in a configuration model sense). In our implementation, we leave it to the CMA algorithm to make this determination.

2.3 Adaptive Resolution Change Mechanism

A challenge with MapElite and indeed any novelty-driven method of search that involves cells/niches is the selection of a suitable cell/niche size in the absence of prior knowledge about the space. Here, a particular feature of the problem at hand is that there is a trade-off between maximising novelty by discovering as many (valid) network descriptions as possible and maximising novelty through exploring the diversity existing in network realisations satisfying a single network specification (subgraph decomposition). To do so, we propose a novel adaptive resolution mechanism defined as follows:

- Start search at the same, low, resolution across all dimensions (where the number of dimensions is the cardinality of the family of subgraphs used to parameterise networks).
- When the ratio between the number of cells being revisited (by mutations) and the number of new cells being discovered exceeds a threshold (set to 2 in our experiments), halve the resolution (across all dimensions) of a number of the cells with the highest measure of interestingness (see Sect. 2.3), and adjust mutation size (for those cells) accordingly.

Two critical components of this mechanism are the measure of interestingness and the relationship between mutation size and cell resolution. They are detailed below.

Measure of Interestingness Changing the resolution of a cell means increasing the likelihood of exploring this area of the space and therefore a criterion is needed that reflects the value of this cell in maximising the second component of the novelty described above, namely, diversity in the structure of network realisations of a single network specification. We propose for this criterion to be the variance in a measure of network structure that is not uniquely determined by the network' subgraph decomposition. In our experiments we used betweenness centrality [9] although others (e.g., [1]) could be used equally.

Practically, each cell maintains a copy of the specification of the fittest individual (since with varying resolution, the cell only specifies a range of values for each dimension of the specification), along with the variance in the measure of interestingness calculated over all individuals sampled when the cell was visited.

Relationship Between Mutation Size and Cell Resolution The value of adaptive mutation mechanisms in controlling the trade-off between exploration and exploitation is well established [7], including within the MapElite framework (see [23] for example). Here, we link the range within which a mutation size is selected to the resolution of the cell in which an individual exists; specifically between 1 (minimum) and 2 (maximum) cell sizes. This guarantees that mutations are small enough to preserve locality (excessively large mutations would lose the benefit of locally heterogeneous resolution) whilst ensuring that any mutation will result in a different cell being explored (since at least one dimension of the network specification will change by at least one cell).

As time increases, the total number of cells in the space will increase such that the average cell size will decrease, and with it, the average mutation size. This means that the evolution process gradually moves from a global search to a local search with emphasis on those areas of the space yielding most diversity.

Implementation Although the idea of starting with a coarse discretization and then increasing granularity was mentioned by the authors of the MapElite framework [21], we are not aware of any such implementation and further we are not aware of any discussion as to the computational requirements. Indeed, even in those papers in which cell size is a point of interest, e.g. [32, 33], the total number of cells is known a priori. For a mechanism such as ours to be computationally tractable in a high-dimensional search space, there is a need for efficient operations for adding, deleting and updating cells. Our implementation relies on a tree data structure developed in-house and available at <https://github.com/harrygcollins/TreeBasedGA>.

3 Results

To allow comparison with previous work, all results that follow concern the exploration of homogeneous networks with $N = 1000$, degree $k = 7$ and global clustering coefficient $C = 0.1$. All runs started from the same starting population

of 5 (randomly picked) valid CMA-generated networks. We analysed the impact of our methodological changes in terms of 3 measures:

- rate of discovery: the number of iterations needed to get a number of networks,
- quality of discovery: the diversity in network specifications uncovered,
- behavioural diversity: whether network diversity impacts dynamics.

3.1 Impact of Diophantine-Based Mutation on Rate of Discovery

To characterise the impact of the use of exact mutations on the search process, we compare it with the baseline of random mutations. Setting the cell resolution to its maximum (i.e., one cell per network specification), we systematically varied mutation size between 2 and 128 in powers of 2. For random mutations, the size was randomly picked. In all cases, we evaluated the rate at which new (valid) networks were discovered as a function of mutations (4700 in all cases) as well as the diversity in network specifications (the coverage). As shown by Fig. 2, there is a significant gain in speed (and number of networks obtained) when using exact mutations, irrespective of the mutation size. It is worth noting that a higher rate of discovery does not necessarily result in a greater number of networks. This is because a substantial number of iterations are lost, either due to out-of-bounds mutations or higher rate of revisits.

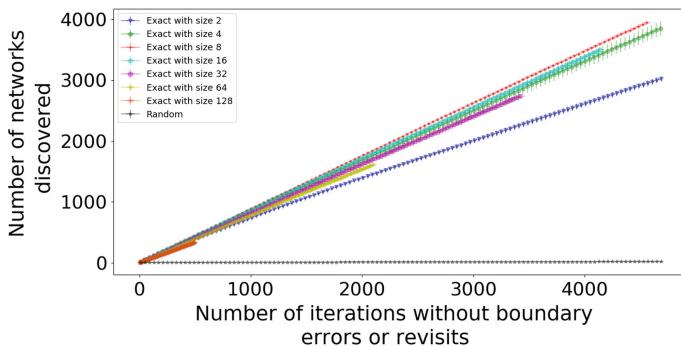


Fig. 2. Rate of discovery of valid networks when exact (with different mutation sizes) and random mutations are used.

Figure 3 shows the frequencies at which subgraphs occur for both exact and random mutations. Whilst random mutations show fairly uniform frequencies, coverage of the space is extremely patchy due to the difficulty obtaining realisable networks. In contrast, exact mutations lead to dense coverage of the space (including beyond that sampled by the random mutations). The Figure clearly shows the impact of mutation size with small mutations (e.g., size 2) resulting in well defined peaks of higher frequency and large mutations (e.g., size 128) resulting in a more uniform histogram although the number of networks found drops

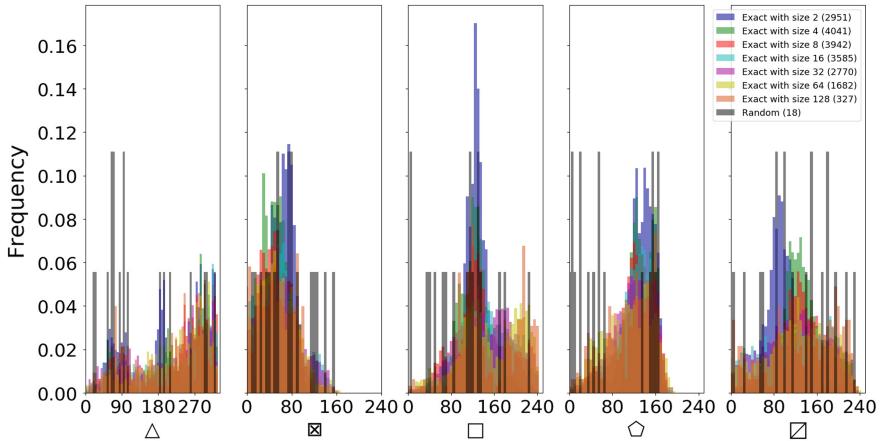


Fig. 3. Frequencies of subgraph population counts for both exact (with mutation sizes from 2 to 128) and random mutations. Bin size was set to 5 for all subgraphs. The number of networks found by each method after 4700 iterations is provided in the legend.

significantly (as explained before). There is therefore a balance to be reached which will be explored below via the adaptive resolution mechanism. In what follows, all experiments start with a cell size of 64 (as a cell size of 128 leads to too few networks).

3.2 Impact of Adaptive Resolution Search on Quality of Discovery

To illustrate the benefit of using adaptive rather than fixed resolution search, we compared the network specifications discovered by our method with those obtained using either a fixed mutation size of 64 (coarsest resolution enabling greatest coverage) or a fixed mutation size of 8 (shown previously to yield the highest rate of discovery). As shown by Fig. 4, networks uncovered using the adaptive resolution search showed the largest breadth of subgraph counts, e.g., largest range of \square . Interestingly, even though using a fixed mutation size of 8 yields a much larger ensemble of networks (almost 20 times larger than using either our method or fixed resolution size of 64), we observe fairly unimodal distributions of subgraph counts reflecting the lack of coverage.

To address the concern that such differences may be a random artefact, we assessed the range of betweenness centrality found in the above networks and that of an identical number of networks generated using BigV and dk-2.1 randomisation. Both methods were used so as to maintain the same distribution of global clustering coefficients (see right panel in Fig. 5), i.e., the BigV rewiring process was stopped when the required clustering coefficient was reached and the dk2.1 randomisation was seeded by CMA-generated networks with the required clustering coefficient. Nevertheless, as shown by Fig. 5 (left panel), we found the

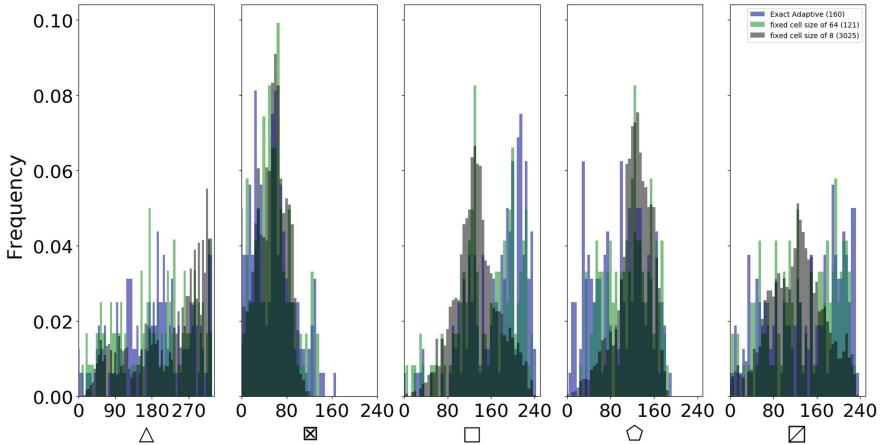


Fig. 4. Frequencies of subgraph population counts for proposed method (blue), fixed mutation size of 64 (green) and fixed mutation size of 8 (black). Bin size was set to 5 for all subgraphs. The number of networks found by each method after 4700 iterations is provided in the legend.

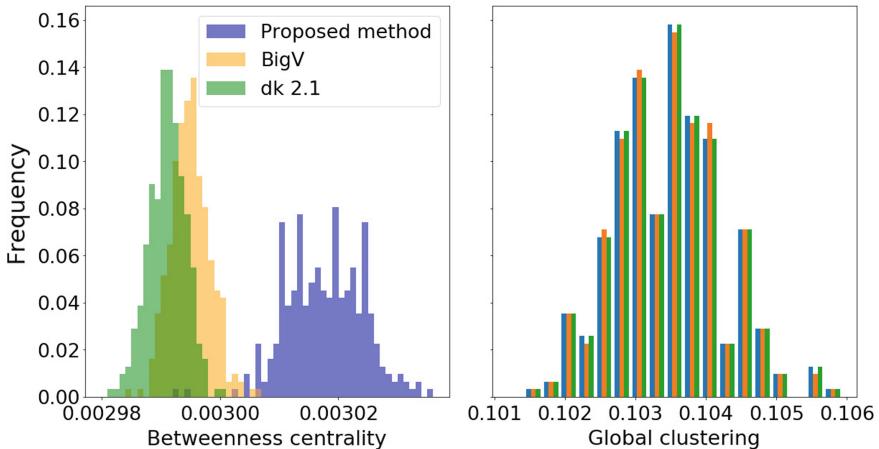


Fig. 5. Histograms of betweenness centrality (left) and clustering coefficients (right) for GA, BigV and dK-generated 7-regular networks.

networks to span a different range of betweenness centrality values. An explanation for this finding will be provided elsewhere (it pertains to CMA seeking to prevent subgraphs around a node from sharing edges) but this demonstrates that the networks are structurally different. To our knowledge, this is also the first evidence that, as considered by its authors, dk2.1 randomisation may not provide uniform sampling.

3.3 Impact of Diversity on Behaviour: Complex Contagion

To illustrate that diversity in such higher-order structure does impact behaviour, we consider the complex contagion model [20]. This model differs from a classical SIR (susceptible-infected-recovered) epidemic by requiring that susceptible nodes are exposed to multiple infectious events before becoming infected. Further, these events must be from different infectious neighbours as only the first infection attempt from an infectious node counts; and infected individuals remain infected for the duration of the epidemic. This dynamics is known to exhibit a critical transition in relation to the number of infected nodes at the start of the epidemic. In preliminary work, we showed that given a degree distribution and a global clustering coefficient, the parameter value at which the transition occurred could fluctuate.

Here, we compared the range of parameter values at which the transition occurred for maximally different (Euclidean distance in their subgraph counts) pairs of networks found (a) through random exploration and (b) through our proposed search mechanism. For each of the networks and for each parameter value, we ran 100 simulations to robustly identify the critical transition. These

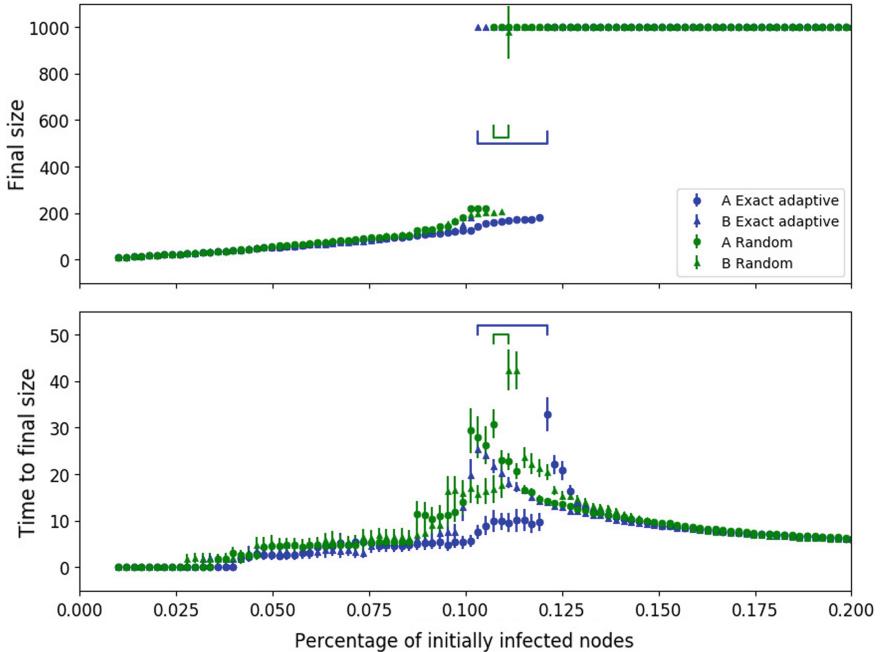


Fig. 6. Ranges of parameter values over which the critical transition of complex contagion simulations occurs in networks found by the proposed search mechanism (blue) and random mutations (green). The critical transition is identified as the value parameter at which there is maximal variability in both final size and the time needed to reach this final size. All $N = 1000$ individuals had threshold $r = 3$ and transmission rate $\beta = 1.0$.

continuous-time simulations were obtained through the event-driven approach described in the appendix of [17]. Figure 6 shows a substantially greater range of parameter values for networks found through the proposed search mechanism, thus confirming greater diversity was achieved with our method (even though the computational cost of eliciting the same number of networks through random mutation was far greater).

4 Discussion

In this paper, we presented a computational framework that makes it possible to explore structural diversity in networks sharing a set of properties. Although we only provided examples when specifying degree distribution and global clustering coefficient, the framework is applicable to other scenarios. Differently from classical network generating methods, which rely on very long mixing times to provide uniform coverage, our approach borrows concepts from evolutionary algorithms to more rapidly identify interesting regions of the solution space, i.e., regions of the space containing structurally more diverse networks. Encoding networks based on their subgraph decomposition provides control over the local structure around nodes. Our experiments have revealed that our methodology makes it easier to elicit structural differences that have an impact on dynamics running on the networks. A promising line of enquiry is to systematically study the importance of a given subgraph on dynamics by restricting the search process to mutations that increase/decrease the number of this subgraph. Another line of further work involves exploring other measures of interestingness, e.g., local clustering diversity, vertex-level entropy. Finally, proper sensitivity analysis on a number of parameters is required but beyond the scope of this paper.

References

1. Anand, K., Bianconi, G.: Entropy measures for networks: toward an information theory of complex topologies. *Phys. Rev. E* **80**(4), 045,102 (2009)
2. Betzel, R.F., Bassett, D.S.: Generative models for network neuroscience: prospects and promise. *J. R. Soc. Interface* **14**, 20170,623 (2017)
3. Courtney, O.T., Bianconi, G.: Generalized network structures: the configuration model and the canonical ensemble of simplicial complexes. *Phys. Rev. E* **93**(6), 062,311 (2016)
4. Cully, A., Demiris, Y.: Quality and diversity optimization: a unifying modular framework. *IEEE Trans. Evol. Comput.* **22**(2), 245–259 (2018)
5. Del Genio, C.I., Kim, H., Toroczkai, Z., Bassler, K.E.: Efficient and exact sampling of simple graphs with given arbitrary degree sequence. *PLoS ONE* **5**(4), 1–7 (2010)
6. Eames, K.T.: Modelling disease spread through random and regular contacts in clustered populations. *Theor. Popul. Biol.* **73**(1), 104–111 (2008)
7. Eiben, Á.E., Hinterding, R., Michalewicz, Z.: Parameter control in evolutionary algorithms. *IEEE Trans. Evol. Comput.* **3**(2), 124–141 (1999)
8. Erdos, P., Gallai, T.: Gráfok előiről fokszámú pontokkal. *Matematikai Lapok* **11**, 264–274 (1960)

9. Freeman, L.C.: A set of measures of centrality based on betweenness. *Sociometry* **35**, 35–41 (1977)
10. Giagkiozis, I., Purshouse, R.C., Fleming, P.J.: An overview of population-based algorithms for multi-objective optimisation. *Int. J. Syst. Sci.* **46**(9), 1572–1599 (2015)
11. Green, D.M., Kiss, I.Z.: Large-scale properties of clustered networks: implications for disease dynamics. *J. Biol. Dyn.* **4**(5), 431–445 (2010)
12. Havas, G., Majewski, B.S., Matthews, K.R.: Extended GCD and hermite normal form algorithms via lattice basis reduction. *Exp. Math.* **7**(2), 125–136 (1998)
13. House, T., Davies, G., Danon, L., Keeling, M.J.: A motif-based approach to network epidemics. *Bull. Math. Biol.* **71**(7), 1693–1706 (2009)
14. House, T., Keeling, M.J.: The impact of contact tracing in clustered populations. *PLoS Comput. Biol.* **6**(3), e1000,721 (2010)
15. Karrer, B., Newman, M.E.: Random graphs containing arbitrary distributions of subgraphs. *Phys. Rev. E* **82**(6), 066,118 (2010)
16. Keeling, M.J., et al.: Networks and the epidemiology of infectious disease. *Interdiscip. Perspect. Infect. Dis.* (2011)
17. Kiss, I.Z., Miller, J.C., Simon, P.L.: Mathematics of Epidemics on Networks. Springer (2017)
18. Liu, H.L., Chen, L., Deb, K., Goodman, E.D.: Investigating the effect of imbalance between convergence and diversity in evolutionary multiobjective algorithms. *IEEE Trans. Evol. Comput.* **21**(3), 408–425 (2017)
19. Lovász, L.: Large networks and graph limits. *Am. Math. Soc.* **60** (2012)
20. Miller, J.C.: Complex contagions and hybrid phase transitions. *J. Complex Netw.* **4**(2), 201–223 (2015)
21. Mouret, J.B., Clune, J.: Illuminating search spaces by mapping elites. [arXiv:1504.04909 v1](https://arxiv.org/abs/1504.04909) (2015)
22. Newman, M.E.: Spread of epidemic disease on networks. *Phys. Rev. E* **66**(1), 016,128 (2002)
23. Nordmoen, J., Samuelsen, E., Ellefsen, K.O., Glette, K.: Dynamic mutation in map-elites for robotic repertoire generation. In: Artificial Life Conference Proceedings, pp. 598–605. MIT Press (2018)
24. Orsini, C., et al.: Quantifying randomness in real networks. *Nat. Commun.* **6**, 8627 (2015)
25. Overbury, P., Kiss, I.Z., Berthouze, L.: A genetic algorithm-based approach to mapping the diversity of networks sharing a given degree distribution and global clustering. In: Complex Networks & Their Applications V, vol. 693, pp. 223–233 (2016)
26. Pastor-Satorras, R., Castellano, C., Van Mieghem, P., Vespignani, A.: Epidemic processes in complex networks. *Rev. Mod. Phys.* **87**(3), 925 (2015)
27. Pugh, J.K., Soros, L.B., Stanley, K.O.: Quality diversity: a new frontier for evolutionary computation. *Front Robot. AI* **3**, 40 (2016)
28. Read, J.M., Keeling, M.J.: Disease evolution on networks: the role of contact structure. *Proc. R. Soc. B* **270**(1516), 699–708 (2003)
29. Ritchie, M., Berthouze, L., House, T., Kiss, I.Z.: Higher-order structure and epidemic dynamics in clustered networks. *J. Theor. Biol.* **348**, 21–32 (2014)
30. Ritchie, M., Berthouze, L., Kiss, I.Z.: Generation and analysis of networks with a prescribed degree sequence and subgraph family: higher-order structure matters. *J. Complex Netw.* **5**(1), 1–31 (2017)
31. Robins, G., Pattison, P., Kalish, Y., Lusher, D.: An introduction to exponential random graph (p^*) models for social networks. *Soc. Netw.* **29**(2), 173–191 (2007)

32. Vassiliades, V., Chatzilygeroudis, K., Mouret, J.B.: A comparison of illumination algorithms in unbounded spaces. In: Proceedings of the Genetic and Evolutionary Computation Conference Companion, pp. 1578–1581. ACM (2017)
33. Vassiliades, V., Chatzilygeroudis, K., Mouret, J.B.: Using centroidal Voronoi tessellations to scale up the multi-dimensional archive of phenotypic elites algorithm. *IEEE Trans. Evol. Comput.* (2017)
34. Watts, D.J., Strogatz, S.H.: Collective dynamics of ‘small-world’ networks. *Nature* **393**(6684), 440 (1998)
35. Zhou, A., Qu, B.Y., Li, H., Zhao, S.Z., Suganthan, P.N., Zhang, Q.: Multiobjective evolutionary algorithms: a survey of the state of the art. *Swarm Evol. Comput.* **1**(1), 32–49 (2011)



Specialist Cops Catching Robbers on Complex Networks

Shiraj Arora, Abhishek Jain, Yenda Ramesh, and M. V. Panduranga Rao^(✉)

Indian Institute of Technology Hyderabad, Sangareddy, India
`cs14resch11010, cs16mtech11001, cs16resch11005, mvp`@iith.ac.in

Abstract. We study a variant of the folklore Cops and Robbers (also known as pursuit evasion) problem on graphs. In this variant, there are different *specializations* of cops and a minimum number of each specialization are necessary to catch a robber. To the best of our knowledge, this variant has not been investigated so far. We believe that this problem will find relevance in several domains like biological systems and epidemic response strategies. We seek to compare the ease of catching robbers executing random walks on various graphs, especially complex networks. We use Statistical Model Checking for the analysis. In this initial work, we report experiments that yield interesting results. For example, we show empirically that it is easier to catch robbers on the Barabási-Albert model, than on the Erdős-Rényi model.

Keywords: Cops and Robbers on graphs · Statistical model checking · Complex networks

1 Introduction

The Cops and Robbers problem, also referred to as Pursuit-Evasion and Graph Searching, has been extensively studied over the years in different flavors. The underlying theme in all variants is that of cop(s) pursuing robber(s) over a certain “environment”. The environment can be continuous, as in the case of planes and polygons, or discrete, as in the case of graphs. In addition to environments, variations of this general theme emerge when parameters like information available to cops and robbers, their movement modes, definition of capture etc change. A popular and intuitive definition of capture involves a cop sharing the same location (vertex) as that of the robber at the same instant of time. While there is a lot of work done on pursuit evasion in continuous environments, we restrict our attention to graphs.

The Cops and Robbers problem was introduced by Quilliot [14] and Nowakowski and Winkler [13]. Traditionally viewed as a game between the cop(s) and robber(s), questions of the following kinds have been investigated in the past: (i) what *strategy* of the cops (robbers) would minimize (maximize) the time that

a robber can evade them? (ii) what classes of graphs would lead to (faster) capture of the robber? (iii) for a given graph, what is the least number of cops (called the *cop number*) needed to catch the robber [3]? Given the richness of the problem, it has received attention from researchers in graph theory, game theory, robot motion planning etc. Indeed, several informative surveys exist in literature with emphasis on different view-points [4, 6, 10].

1.1 Random Walks on Complex Networks

Randomized versions of the pursuit-evasion have been investigated in the past. Adler et al. [1] obtained optimal strategies for a randomized version with one pursuer (a hunter) and one evader (a rabbit). Both the hunter and the rabbit have sources of randomness at their disposal for incorporation into their strategies.

This paper concerns a variant of the problem that, to the best of our knowledge, has not been investigated so far. In this variant, which we call Specialist Cops and Robbers, every cop has one of a finite number of specializations or *types*. Further, we require that a certain minimum number of each specialization have to occupy the same vertex as a robber at the same time, to be able to catch the robber. This represents the scenarios where law enforcement officials with different expertise are required to identify and capture a criminal. Such variants can be used to study various biological processes—for instance, different types of insects being needed to fight invasions, and different types of cells being needed for raising an immune response in the human body. It is also possible to imagine applications in computational epidemiology where coincidence of epidemic limiting factors like climatic conditions and specialized health care units help control the epidemic.

Our problem statement is closest to the work of Cooper et al. [8], who estimated the expected time for a prey population to become extinct when hunted by a predator population, while executing a random walk on a random r -regular graph. The problem is also relevant to research into dynamical phenomena over multiplex complex networks.

In variants of the problem such as ours, it is difficult to derive closed form expressions for quantities like expected values of time elapsed before the robber is caught. Indeed, one can imagine even more complicated variants that incorporate complex behaviors and interaction modes of cops or robbers. Moreover, it should be possible to ask more complex queries of the system, rather than coarse-grained queries.

In such circumstances, we propose the use of statistical model checking instead. Model checking has been a popular technique for analysis of complex man-made or natural systems. The goal of (probabilistic) model checking is to decide whether or not a (stochastic) system in an appropriate modeling formalism satisfies properties specified in an appropriate system of logic. While there exist methods that involve exhaustive state space exploration, statistical model checking offers a faster and widely applicable alternative. In the case of complex behavior of cops and robbers, they can be modeled as multiagent systems and statistical model checking can be applied.

For statistical model checking, we use a tool being developed in-house, that we seek to eventually optimize for multi-agent systems on graphs. Essentially, the tool takes two inputs (i) a description of agents and their behavior and (ii) a Probabilistic Bounded Linear Temporal Logic (PBLTL) formula, along with SMC parameters. In the present case, the agents are the cops and robbers and their behavior is simply their movement—either a random walk, or a generic, randomly chosen stochastic transition matrix. The tool then generates a simulator and performs statistical model checking on the runs.

We report results where the cops and the robbers execute random walks on random r -regular graphs, and Erdős-Rényi (ER) random networks [9], Barabási-Albert networks (BA) [5] and Markovian movement on a real world street topology. Finally, we argue through an example that (statistical) model checking can be a very useful tool for intricate analysis of such dynamics on networks.

The paper is organized as follows. The next section provides a brief primer on statistical model checking, and the temporal logic PBLTL. Section 3 describes our formulation of the problem and discusses experimental results. Section 4 concludes the paper and discusses future directions.

2 Preliminaries and Related Work

2.1 (Statistical) Model Checking

Model checking of stochastic systems can be done either using numerical or statistical techniques. While numerical model checking algorithms generate the exact probability of a property being true in a given system, they often do it at the cost of more time and memory. Statistical model checking, on the other hand, requires lesser time and memory for the computation but generates results with some confidence parameter. Statistical model checking works by simulating finitely many executions of the system and using various statistical techniques to infer whether a property is satisfied or not [12, 15, 16, 18, 19]. The interested reader is also referred to [11] and [2], that survey recent advances in the area of statistical model checking. There are two important approaches for statistical model checking, namely, Hypothesis Testing and Estimation methods. Hypothesis testing is used to verify whether the probability of a property being true crosses the required threshold probability or not.

Estimation methods, on the contrary, estimate the probability of the given property to be true. If p is the actual probability and p' is the estimated probability, and we require that

$$\Pr[|p - p'| \geq \varepsilon] \leq \delta,$$

for $\varepsilon \geq 0$ and $\delta \geq 0$, then the number of samples required is given by

$$N \geq \frac{\ln(\frac{2}{\delta})}{2\varepsilon^2}, \quad (1)$$

as a consequence of the Chernoff-Hoeffding bound.

The results discussed in this paper use this estimation method and Wald's Sequential Ratio Probability Test (SPRT) [17].

The samples that are drawn are checked against a property stated in a system of logic. The query logic that we use in our tool is Probabilistic BLTL.

2.2 Probabilistic BLTL

We briefly sketch the definition of PBLTL. The interested reader is referred to, for example, [7] for details.

Suppose we are given a model \mathcal{M} of a system, that includes the set V of its real-valued variables and the set S , its discrete state space. For $x \in V$, we call as atomic propositions, predicates of the form $x \sim \nu$ where $\sim \in \{\geq, \leq, =\}$ and $\nu \in \mathbb{Q}$. Bounded Linear Temporal Logic (BLTL) is an extension of LTL that allows bounding of its operators and is used to formally express a property ϕ of \mathcal{M} . *Syntax of BLTL*:

$$\phi ::= x \sim \nu \mid \phi_1 \wedge \phi_2 \mid \neg \phi \mid X^{\leq t} \phi \mid \phi_1 U^{\leq t} \phi_2$$

where $t \in \mathbb{Q}_{\geq 0}$.

A trace $\rho = (s_0, t_0)(s_1, t_1)\dots$ is a sequence of states s_i in an execution of the system along with its sojourn times t_i in those states. We denote the fact that a trace ρ satisfies a BLTL formula ϕ by $\rho \models \phi$. Further, let ρ^t denote the trace suffix that starts at step t . Then, the semantics of BLTL is as follows:

- $\rho^t \models x \sim \nu$ iff the atomic proposition holds in the state s_t
- $\rho^t \models \phi_1 \wedge \phi_2$ iff $\rho^t \models \phi_1$ and $\rho^t \models \phi_2$
- $\rho^t \models \neg \phi$ iff $\rho^t \not\models \phi$ (other boolean connectives like disjunction and implication can be derived from these)
- $\rho^t \models X^{\leq t'} \phi$ iff $\exists i \leq t' \mid \rho^i \models \phi$
- $\rho^t \models \phi_1 U^{\leq t'} \phi_2$ iff $\exists i \leq t' \mid \rho^i \models \phi_2$ and $\forall j < i, \rho^j \models \phi_1$.

Probabilistic BLTL is a simple extension: $Pr_{\geq \theta}(\phi)$ where ϕ is a BLTL formula and $0 \leq \theta \leq 1$. A (stochastic) model is said to satisfy the formula $Pr_{\geq \theta}(\phi)$ if the probability that a run of the model satisfies the BLTL formula ϕ with a probability at least θ .

3 Randomized Cops and Robber

3.1 Problem Statement

Given: (i) A directed graph, (ii) an initial placement of n_r robbers and n_1 cops of type 1, n_2 cops of type 2 and n_3 cops of type 3 on the vertices of the graph (iii) transition functions defining the movement of the cops and robbers on the graph and (iv) capturing condition—how many cops of each type are needed to be there on the same vertex as that of the robber for capture: l_1 , l_2 and l_3 , and (iv) a positive integer t .

Find: (i) What is the probability that all the robbers are caught within t steps? (ii) What is the cop number to ensure capture within t steps?

In all of our experiments, the following remain the same. Initial placement of the cops and robbers is random. The total number of cops is always 300. If there are three specializations or types of cops, then there are 100 cops of each type, and one of each type is needed to catch a robber. If there are cops of only one type, then three cops of that type are needed. Whenever there are multiple robbers, *isCaught* stands for all robbers being caught. Across all the experiments, the confidence parameter δ is 0.01 and the error bound, ε is 0.01.

The query that we use is:

$$\Pr_{\geq \theta}[\text{true} \quad U^{\leq t} \quad \text{isCaught}]$$

where *isCaught* is an atomic proposition defined as an algorithm (Algorithm 1) that is given as input to the model checker. Unless otherwise specified, $t = 150$ in all the experiments.

Algorithm 1 isCaught

```

1: all type counters set to 0;
2: robber_current_location  $\leftarrow$  location(robber)
3: for all Type 1 Cops  $c_1$  do
4:   if location( $c_1$ )=robber_current_location then  $Type1\_counter \leftarrow Type1\_counter + 1$ ;
5: for all Type 2 Cops  $c_2$  do
6:   if location( $c_2$ )=robber_current_location then  $Type2\_counter \leftarrow Type2\_counter + 1$ ;
7: for all Type 3 Cops  $c_3$  do
8:   if location( $c_3$ )=robber_current_location then  $Type3\_counter \leftarrow Type3\_counter + 1$ ;
9: if ( $Type1\_counter \geq l_1$ )  $\wedge$  ( $Type2\_counter \geq l_2$ )  $\wedge$  ( $Type3\_counter \geq l_3$ ) then
10:   return True
10: else return False

```

We now discuss representative results for different scenarios. Figure 1 shows the results for k -regular graphs. It is seen that k does not have a pronounced effect on the estimated probability of all the robbers getting caught. This agrees with the result that Cooper et al. [8] derived for the case when all cops are of the same specialization or type. It can also be seen that the number of types has a larger impact than the total number of robbers—it is easier to catch a larger number of robbers if a fewer types of cops are required to catch one.

Figures 2 and 3 show an interesting phenomenon. In both ER and BA networks, it is harder to catch a robber when one each of three types is needed. However, it is easier to catch robbers in BA networks than in ER networks. So much so that the probabilities are comparable for the three types case in BA and the one type case in ER. Figure 4 yields yet another interesting observation. The

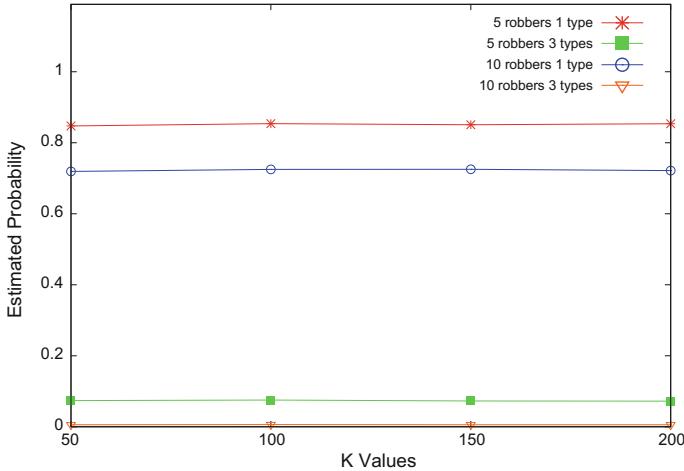


Fig. 1. Probability of robbers getting caught versus different values of k for k -regular graphs

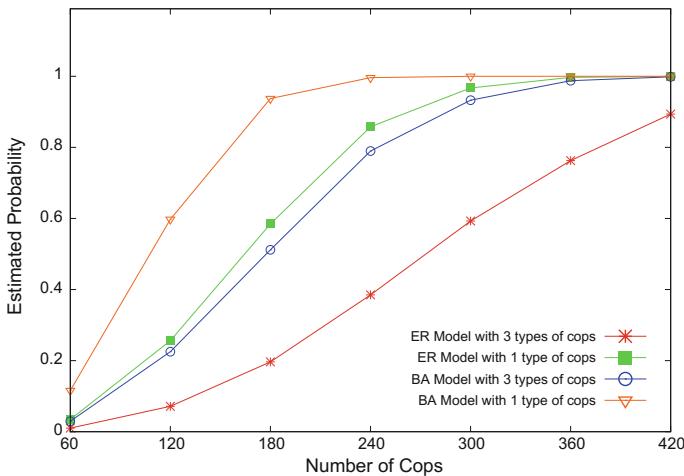


Fig. 2. Probability of a single robber getting caught versus total number of cops.

probability of a robber getting caught does not change with increasing density of the graph, except for the case of three types in BA networks, where it decreases. Finally Fig. 5 shows the fall in probability as the number of robbers needed to be caught increases, with 300 cops of same type.

It turns out that in the simulations for both networks, if 100 cops are chasing robbers, and we require that just one cop being on the same vertex is sufficient to catch the robbers on that vertex, the success probability is close to 1. This indicates that it would be wise to form patrol units with all specialized cops in

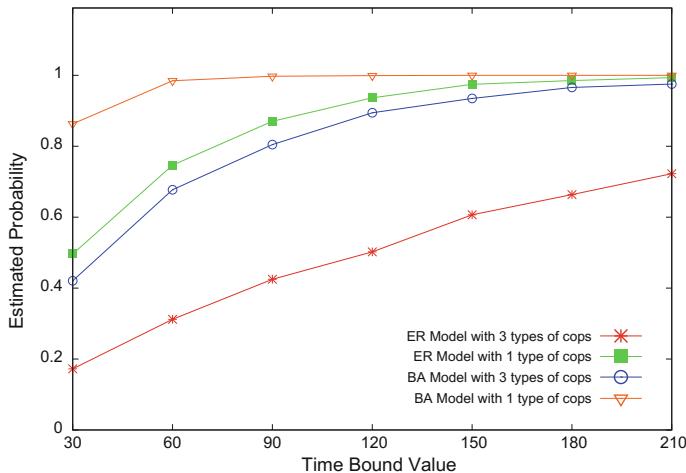


Fig. 3. Probability of a single robber getting caught versus number of steps

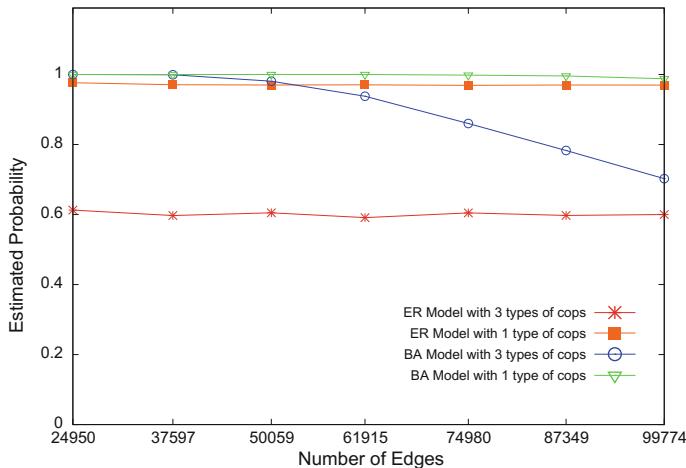


Fig. 4. Probability of a single robber getting caught versus density of the graph

it. The above discussion then is for primarily for circumstances (like epidemics) where the “cops” necessarily move independently.

3.2 Cop Number

Let all the cops be of the same type, say Type 1. Suppose further that to catch a robber, at least l_1 cops are needed. Then, what is the total number n_1 of cops required to catch a robber within t steps with a probability at least (say) 0.8? We answer this query on a “real world network”.

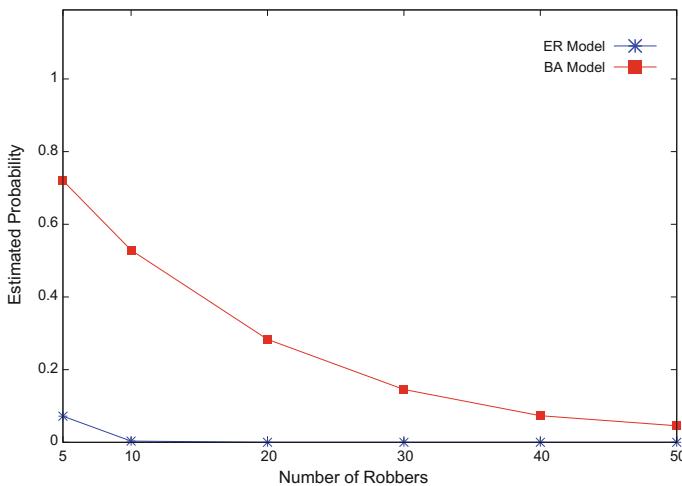


Fig. 5. Probability of several robbers getting caught

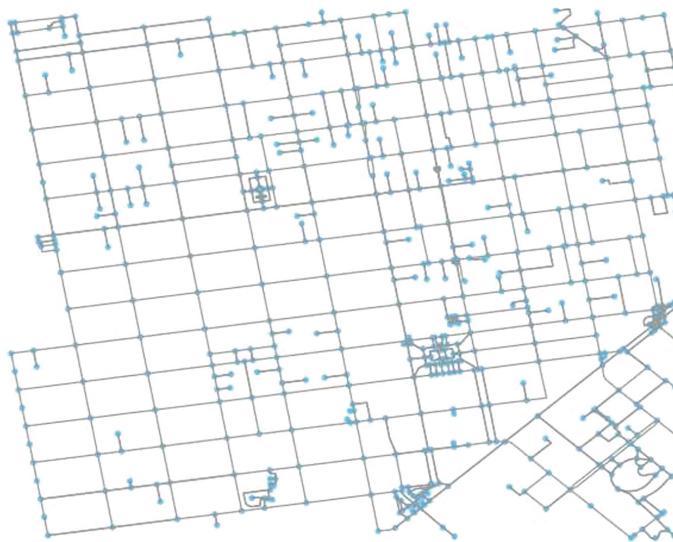


Fig. 6. Graph for Street Network

The graph is generated using the Python library OSMNX which models the real cities and streets as the graphs using the latitude and longitude of the geographical location. For this scenario we have used a location in San Francisco, USA. The network shown in Fig. 6 has 654 nodes and 1400 edges. Some edges in the street network are unidirectional whereas others are bidirectional. The cops and the robber are allowed to move across the different nodes in the graph as per their respective randomly generated stochastic transition matrices.

For the estimation of *Cop Number*, Sequential Ratio Probability Test (SPRT) [17] was used. Initial placement of cops and robber is done randomly and further movement is based on the stochastic probability distribution which are generated before the start of the simulation.

The probability value p for the estimation of *Cop Number* is 0.8.

Table 1. Estimated Cop Number

Number of Cops needed for capture	Estimated Cop Number
1	182
2	762
3	1385

Table 1 shows the estimated *Cop Number* in the street network considered for simulations when different number of cops are required at the robber’s location at the same time instant to catch the robber.

3.3 Scope of (Statistical) Model Checking

We conclude by arguing that (Statistical) Model Checking can be used effectively for intricate analysis of cops and robbers variants. Suppose that there is one “mastermind” robber b who can be identified and testified against by any of a set A of “henchmen”. This works only if b is caught when at least one of A is already in custody. One strategy could be the following. If a henchman is caught, we can hold him until b is caught. However, laws usually disallow arbitrarily long detention of suspects. Therefore b has to be caught within t time units of detaining a henchman. Then, given a t prescribed by the law, a natural question to ask is, what is the probability that b is caught within t time units of a henchman being caught? Formally, in PBLTL,

$$\Pr_{\geq \theta}[\text{true} \quad U^{\leq t} \quad A \rightarrow X^{\leq t}(\text{isCaught}(b))].$$

Indeed, this could also help in fixing a reasonable t by legislative bodies.

Figure 7, for example, shows that a detention of about 8–10 time units of the henchmen works with very high probability for arresting the “mastermind” robber. The experiment that we ran is for five henchmen and one mastermind being hunted by 100 cops over networks of 500 nodes, where every entity is performing a random walk. Otherwise, the simulation parameters are the same as those for previous experiments.

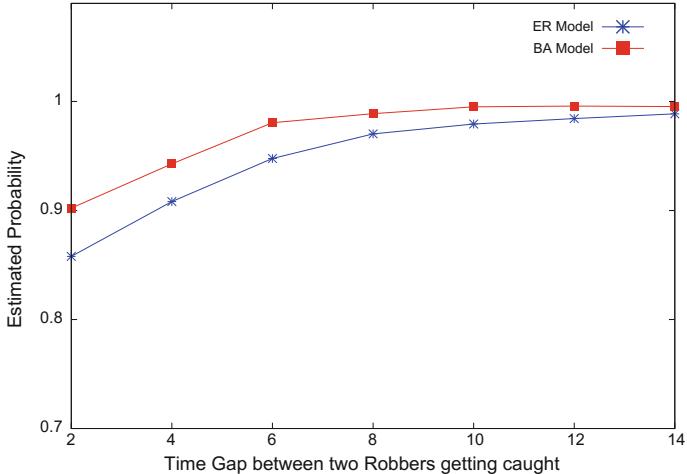


Fig. 7. Probability of the mastermind getting caught within t steps of a henchman getting caught

4 Future Work

We believe that it would be interesting to investigate the cops and robbers problem with more complex cop/robber behavior using statistical model checking. The technique will prove particularly useful when obtaining closed form solutions is difficult, and complex queries have to be asked of the system. To that end, we are in the process of building a statistical model checker optimized for generic versions of the cops and robbers problem.

A direct application of finding cop number would be in logistics of epidemic management. Suppose only a limited number of health care units exist, which have to move around between different localities where an epidemic has spread. If the presence of a health care unit in a locality accelerates recovery rates and/or reduces infection rates, one can ask the question, for example, what number of such health care units are needed if the epidemic is to be eradicated within a specific time period? Formally, we seek to verify

$$\Pr_{\geq \theta} [\text{true} \quad U^{\leq t} \quad \text{notInfected}]$$

where t is the time bound and *notInfected* is the predicate that indicates all localities being epidemic free. The analogue of multiple types of cops is the case when there are factors other than health-care units that are essentially independent of the movement of health-care units, like weather. A tool that facilitates this analysis would help health-care authorities in planning an effective epidemic response strategy.

References

1. Adler, M., Räcke, H., Sivadasan, N., Sohler, C., Vöcking, B.: Randomized pursuit-evasion in graphs. In: Widmayer, P., Eidenbenz, S., Triguero, F., Morales, R., Conejo, R., Hennessy, M. (eds.) *Automata, Languages and Programming*, pp. 901–912. Springer, Berlin Heidelberg, Berlin, Heidelberg (2002)
2. Agha, G., Palmskog, K.: A survey of statistical model checking. *ACM Trans. Model. Comput. Simul.* **28**(1), 6:1–6, 39 (2018). <https://doi.org/10.1145/3158668>
3. Aigner, M., Fromme, M.: A game of cops and robbers. *Discret. Appl. Math.* **8**(1), 1–12 (1984)
4. Alspach, B.: Searching and sweeping graphs: a brief survey. *Le Matematiche* **59**, 5–37 (2004)
5. Barabási, A.L., Albert, R.: Emergence of scaling in random networks. *Science* **286**(5439), 509–512 (1999). <https://doi.org/10.1126/science.286.5439.509>. <http://science.scienmag.org/content/286/5439/509>
6. Chung, T.H., Hollinger, G.A., Isler, V.: Search and pursuit-evasion in mobile robotics. *Auton. Robot.* **31**(4), 299 (2011). <https://doi.org/10.1007/s10514-011-9241-4>
7. Clarke, E.M., Zuliani, P.: Statistical model checking for cyber-physical systems. In: Bultan, T., Hsiung, P.A. (eds.) *Automated Technology for Verification and Analysis*, pp. 1–12. Springer, Berlin Heidelberg, Berlin, Heidelberg (2011)
8. Cooper, C., Frieze, A., Radzik, T.: Multiple random walks and interacting particle systems. In: Albers, S., Marchetti-Spaccamela, A., Matias, Y., Nikoletseas, S., Thomas, W. (eds.) *Automata, Languages and Programming*, pp. 399–410. Springer, Berlin Heidelberg, Berlin, Heidelberg (2009)
9. Erdős, P., Rényi, A.: On the evolution of random graphs. In: Publication of the Mathematical Institute of the Hungarian Academy of Science, pp. 17–61 (1960)
10. Fomin, F.V., Thilikos, D.M.: An annotated bibliography on guaranteed graph searching. *Theor. Comput. Sci.* **399**, 236–245 (2008)
11. Legay, A., Delahaye, B., Bensalem, S.: Statistical model checking: an overview. In: Barringer, H., et al. (eds.) *Runtime Verification. Lecture Notes in Computer Science*, vol. 6418, pp. 122–135. Springer, Berlin Heidelberg (2010)
12. Nimal, V.: Statistical approaches for probabilistic model checking. Ph.D. thesis, University of Oxford (2010)
13. Nowakowski, R., Winkler, P.: Vertex-to-vertex pursuit in a graph. *Discret. Math.* **43**(2–3), 235–239 (1983)
14. Quilliot, A.: Problèmes de jeux, de point fixe, de connectivité et de représentation sur des graphes, des ensembles ordonnés et des hypergraphes, pp. 131–145. These d’Etat, Université de Paris VI pp (1983)
15. Sen, K., Viswanathan, M., Agha, G.: Computer Aided Verification: 16th International Conference, CAV 2004, Boston, MA, USA, 13–17 July 2004. Proceedings, chap. *Statistical Model Checking of Black-Box Probabilistic Systems*, pp. 202–215. Springer Berlin Heidelberg, Berlin, Heidelberg (2004)
16. Sen, K., Viswanathan, M., Agha, G.: On statistical model checking of stochastic systems. In: 17th International Conference on Computer Aided Verification, CAV 2005, Edinburgh, Scotland, UK, 6–10 July 2005, pp. 266–280 (2005)
17. Wald, A.: Sequential tests of statistical hypotheses. *Ann. Math. Stat.* **16**(2), 117–186 (1945)
18. Younes, H.L.: Verification and planning for stochastic processes with asynchronous events. Technical report, Carnegie-Mellon University (2005)

19. Younes, H.L.S., Simmons, R.G.: Probabilistic verification of discrete event systems using acceptance sampling. In: 14th International Conference on Computer Aided Verification, CAV 2002, Copenhagen, Denmark, 27–31 July 2002, pp. 223–235 (2002)



Evaluating the Natural Variability in Generative Models for Complex Networks

Viplove Arora and Mario Ventresca^(✉)

School of Industrial Engineering, Purdue University, West Lafayette IN 47907, USA
mventresca@purdue.edu

Abstract. Complex networks are used to represent real-world systems using sets of nodes and edges that represent elements and their interactions, respectively. A principled approach to understand these network structures (and the processes that give rise to them) is to formulate generative models and infer their parameters from given data. Ideally, a generative model should be able to synthesize networks that belong to the same population as the observed data, but most models are not designed to accomplish this task. Due to the scarcity of data in the form of populations of networks, generative models are typically formulated to learn parameters from a single network observation, hence ignoring the natural variability of network populations. In this paper, we evaluate four generative models with respect to their ability to synthesize networks that belong to the same population as the observed network. Our empirical analysis quantifying the ability of network models to replicate characteristics of a population of networks highlights the need for rethinking the way we evaluate the goodness of fit of new and existing network models.

Keywords: Network models · Network populations · Network analysis

1 Introduction

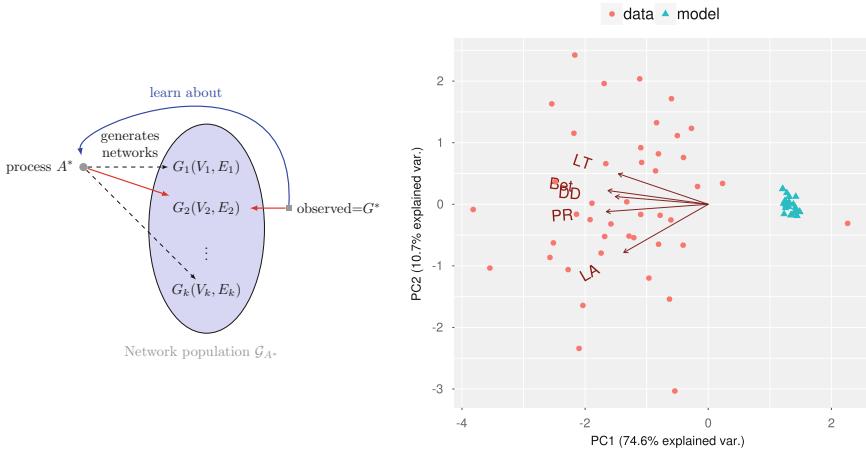
Many natural and artificial systems can be described as networks composed of sets of nodes and edges that represent system elements and their interactions, respectively [20]. The structure of these complex networks can also capture the functional abilities of the system they represent. The analysis and modeling of complex networks has provided transformative perspectives, models and methods in diverse application domains such as computer science, sociology, chemistry, biology, anthropology, psychology, geography, history and engineering [11, 20]. In particular, the increasing availability of network data from a wide variety of sources such as the internet, online social networks, citation and collaboration networks, biological networks (brain connectivity, protein-protein interactions), etc. has fueled a great deal of interest in the analysis and modeling of networks.

A goal of network modeling is to solve the problem of decoding how the observed structure of a network supports its perceived/desired function [1]. As a consequence, a long-standing question in the network science community has been regarding the existence of a model capable of generating synthetic networks that are statistically representative of real networks. Most of the existing models either make assumptions biased by system-specific observations that are not plausible across domains, or focus on replicating a few predefined topological features, such as degree distribution and clustering, at the expense of other potentially more important characteristics. Without any indication that they are either necessary or sufficient as descriptors for the actual network data, these summary quantities can often be highly misleading [11]. Further, even when a model is capable of consistently reproducing a set of target properties, it might fail to capture the naturally occurring stochasticity in those properties [12]. This is depicted in Fig. 1b, which shows the variance in a population of real-world networks (evaluated using five different network properties) compared with the networks synthesized by dk -random graphs [21].

Definition 1 (Network Population). *Let $G_1(V_1, E_1)$ be a network that has non-zero probability of being generated using the process A^* . A set of such realizations $\mathcal{G}_{A^*} = \{G_1(V_1, E_1), \dots, G_k(V_k, E_k)\}$ is called a network population.*

The inability of certain network models to reproduce the naturally occurring variability in networks can be attributed to the fact that they sample each edge independently through Bernoulli distributions [19]. Further, modeling networks based on a single network observation might bias a network model to synthesize networks that over-fit the observed network, thus ignoring the natural variability. In statistical analysis, the goodness of fit of a model is evaluated by measuring the discrepancy between observed values and the values expected under the model in question. Similarly, in the context of networks, we would like to *evaluate the ability of a model to approximate the population of networks that can be created by a process A^* using a single observed network G^** (see Fig. 1a for a pictorial representation). Unfortunately, we generally do not have a population of independent instances of networks that can be used to draw a set of samples [17]. To recap, an ideal generative model M would exactly correspond to the true process A^* that defines the dynamical processes responsible for the observed data G^* . That is, if A^* defines a probability distribution $\mathbb{P}_G(A^*) \forall G \in A^*$, then $\mathbb{P}_G(M)$ and $\mathbb{P}_G(A^*)$ would be identical. As stated above, A^* is usually unknown and the number of observed networks in the data G^* are usually small (sometimes only one).

In this work, we study the distributional properties of four competing generative models: Chung-Lu model and dk -random graphs, which are designed to match specific properties of the observed network, and exponential random graphs and action-based network generators, which were designed to capture local graph properties (Sect. 2 briefly describes each model). We consider networks drawn from three known processes and three real-world populations. For each model, we learn parameters from a representative sample (see Fig. 1a).



(a) Assuming G^* is not an outlier, how well do existing network models approximate the process A^* ?

(b) Variance in real-world data of Indian villages versus networks synthesized by dk -random graphs.

Fig. 1. Evaluation of network models: Fig. 1a depicts the procedure used for evaluating network models in this paper, while Fig. 1b highlights the need for such a procedure. Social networks in Indian villages [4] is the data used in Fig. 1b. The networks are compared using the Kolmogorov-Smirnov distance on five different network properties, namely: degree distribution (DD), local assortativity (LA), local transitivity (LT), PageRank (PR), and Betweenness (Bet). Principal component analysis of the five properties was performed, where the first two principal components were able to account for more than 85% of the variance.

Then, the learned models are used to synthesize networks followed by an investigation of their distributional properties. This evaluation is done by comparing the statistical properties of the synthesized networks with the properties of the corresponding population of networks.

2 Background

In this section, we briefly introduce the four generative models that are used in the empirical analysis in Sect. 4. Where applicable, details regarding user-defined inputs have also been provided. Further, as shown in Fig. 1a, each model uses the network G^* to learn a fixed set of parameters.

2.1 Chung-Lu Model

In the Chung-Lu model [6, 7], a vertex i is assigned a degree d_i from the given degree distribution and an edge is placed between the vertex pair (i, j) with probability proportional to $d_i d_j$, i.e. the probability that an edge exists between nodes i and j is given by

$$\mathbb{P}_{i,j} = \frac{d_i d_j}{\sum_k d_k}$$

The Chung-Lu model is often used as the baseline for comparison owing to its simplicity and ability to synthesize fairly realistic networks [24]. Unfortunately, the Chung-Lu model synthesizes networks with low clustering coefficients making it unsuitable for most real-world applications.

2.2 Exponential Random Graphs

One of the most popular statistical network models in the social science literature are the exponential random graph models (ERGM) [29, 32]. ERGMs represent probability distributions over networks with an exponential linear model that uses feature counts of local graph properties considered relevant by the modeler (for example, edges, triangles, paths, etc.):

$$P(\mathbf{Y} = G^* | \boldsymbol{\theta}) = \frac{1}{Z} \exp(\boldsymbol{\theta}^T \phi(G^*))$$

where (i) $\phi(G^*)$ are feature counts of G^* ; (ii) $\boldsymbol{\theta}$ are parameters to be learned; (iii) Z is a normalizing constant. Though ERGMs are the most widely used models for social networks, they are plagued with the degeneracy problem (i.e., the probability distribution is biased towards empty and complete networks), whereas real-world networks are sparse. In our experimental evaluation, the following feature counts $\phi(G^*)$ were used as they are known to be capable of circumventing the degeneracy problem (see [13, 28] for more details): (i) total number of edges, (ii) geometrically weighted degree distribution, (iii) geometrically weighted dyadwise shared partner distribution, and (iv) geometrically weighted edgewise shared partner distribution.

2.3 *dk*-Random Graphs

In [21] it was observed that fixing some structural properties in a network model to those observed in the given network can lead to the appearance of other statistical properties as a consequence. These observations follow from earlier research on the *dk*-series [16], which defines a series of null models or random graph ensembles [21]. Consequently, *dk*-random graphs [21] model networks as random ensembles, where ensemble size is controlled using *dk*-distributions. *dk*-random graphs rely on ergodic edge-swapping operations to sample networks from an ensemble defined using the chosen *dk*-distributions. Experimental results [2, 21, 27] have shown that the networks synthesized by *dk*-random graphs have very low dissimilarity to most real-world networks. Despite this fact, the limited inferential capabilities and inability to perform tasks such as compression, extrapolation, etc. limit the utility of *dk*-random graphs. In our empirical analysis, we used the *dk2.5* variant as it is known to outperform other network models on a variety of measures [2, 27].

2.4 Action-Based Network Generators

The action-based approach of [2] models networks using local node interactions based on simple link creation processes known as actions. An action is a decision

process a node uses to form a link with another node. Given a pre-defined set of actions, the aim of action-based networks is to learn a probability distribution over these actions, such that the resultant model can synthesize networks statistically similar to a given network observation. A synthesis algorithm $f(\mathbf{M}, n)$ can then be used to synthesize networks containing n nodes using the learned action-based model \mathbf{M} , leading to action-based network generators (ABNG). The fundamental idea behind action-based networks is to define a unifying network generative process, which follows from observations by [33] who note that there must exist an assembling algorithm to combine local mechanisms for emergence of different complex network structures. For an observed network C^* , the action-based model \mathbf{M} is determined by solving a multi-objective optimization problem. In our empirical analysis, we used degree distribution, local assortativity [25], and local transitivity of the observed network as the set of network properties in the objective function.

3 Experimental Setup

Evaluation of the distributional properties of a generative model requires a well-defined methodology that correctly represents the distribution over networks. Although a model-based technique for hypothesis testing of networks has been proposed in the literature [18], it heavily relies on the choice of a baseline model. Alternatively, one could build on the concept of a network morphospace [3], which provides a coarse-grained approach for classifying and mapping network architectures according to a set of network-level structural characteristics. The network morphospace can be transformed to a network dissimilarity space ($\mathfrak{D}_G \subset \mathbb{R}^d$), where networks are placed based on their dissimilarity to the observed network $C^* \in G$ with respect to a variety of node-level structural characteristics. The true process and network models also have counterpart distributions $\mathbb{P}_{\mathfrak{D}_G}(A^*)$ and $\mathbb{P}_{\mathfrak{D}_G}(M)$ in the network dissimilarity space. In an appropriately defined dissimilarity space, if $\mathbb{P}_{\mathfrak{D}_G}(M)$ sufficiently approximates $\mathbb{P}_{\mathfrak{D}_G}(A^*)$, we might be able to conclude that model M can synthesize networks that belong to the same population as the observed network G^* .

The utility of such a network dissimilarity space relies heavily on the choice of node-level metrics used for network comparison. Network science provides numerous quantitative tools to measure and classify different patterns of local and global network architectures across disparate types of systems. A set of node-level measures that could prove particularly useful for the network dissimilarity space is provided by the dk -series [21], which is a systematic series of properties (Y_0, Y_1, \dots) of network structure defined in a way such that each Y_i provides more detailed information about the network structure and Y_n fully characterizes a network with n nodes. [21] have shown that the first three terms in the dk -series ($Y = \text{degrees} + \text{correlations} + \text{clustering/transitivity}$) are capable of almost fully defining local and global organization of most real-world networks that do not exhibit community structure.

4 Experimental Results

In our experiments to evaluate the distributional properties of generative models, we propose to use the Kolmogorov-Smirnov statistic for evaluating the dissimilarity between networks based on node-level properties of degree, correlations and clustering. To examine the ability of existing generative models to approximate the ground truth process using a single network observation (assuming it is representative of the true process with respect to the measures of interest), we propose two different experiments: (i) a controlled experiment where the true process is known, and (ii) set of real-world networks that have most likely evolved from a common generative process (for example, social interaction networks of different villages).

The following datasets were used for real-world network populations: (i) Contact Networks: 69 daily cumulated networks where nodes represent visitors of a Science Gallery while the edges represent close-range face-to-face proximity between the concerned persons [14]; (ii) Social Networks in Indian Villages: Data from a survey of social networks in 75 villages in rural southern Karnataka, a state in India [4]; and (iii) Autonomous Systems: The graph of routers comprising the Internet organized into sub-graphs called Autonomous Systems (AS). The dataset [30] contains 733 daily instances which span an interval of 785 days from November 8 1997 to January 2 2000. The first 100 networks were used in this study. The networks obtained from each of these datasets have most likely evolved from similar underlying social mechanisms, hence belonging to the same network population.

Figure 2 shows the results for the first set of experiments when the Barabási-Albert [5] and Forest Fire models [15] are used as the true processes A^* . For the second experiment, we consider the three real-world network populations described above, with results presented in Figs. 3 and 5. Results presented in Figs. 2, 3, 4 and 5 are composed of three different plots:

1. Scatter plots below the diagonal show each synthesized/real network as a point in the network dissimilarity space, where the coordinates are computed using the Kolmogorov-Smirnov distance of the associated properties when the network is compared to the observed network G^* (the observed network itself is at the (0,0) position). Network models (colored triangles) showing higher overlap with networks originating from the true process (black dots) are better.
2. In the blocks above the diagonal, we evaluate the amount of overlap between $\mathbb{P}_{\mathfrak{D}_G}(A^*)$ and $\mathbb{P}_{\mathfrak{D}_G}(M)$ using the 2-D KS distance [22] (lower the better). This quantifies the extent to which a given generative model is able to reproduce the distributional properties of the population representing the true process.
3. Plots along the diagonal show the density distributions of the Kolmogorov-Smirnov distance of the associated properties when the network is compared to the observed network G^* .

Based on Figs. 2 and 3 we can easily conclude that ABNG consistently outperforms the other models considered here by replicating the natural variability

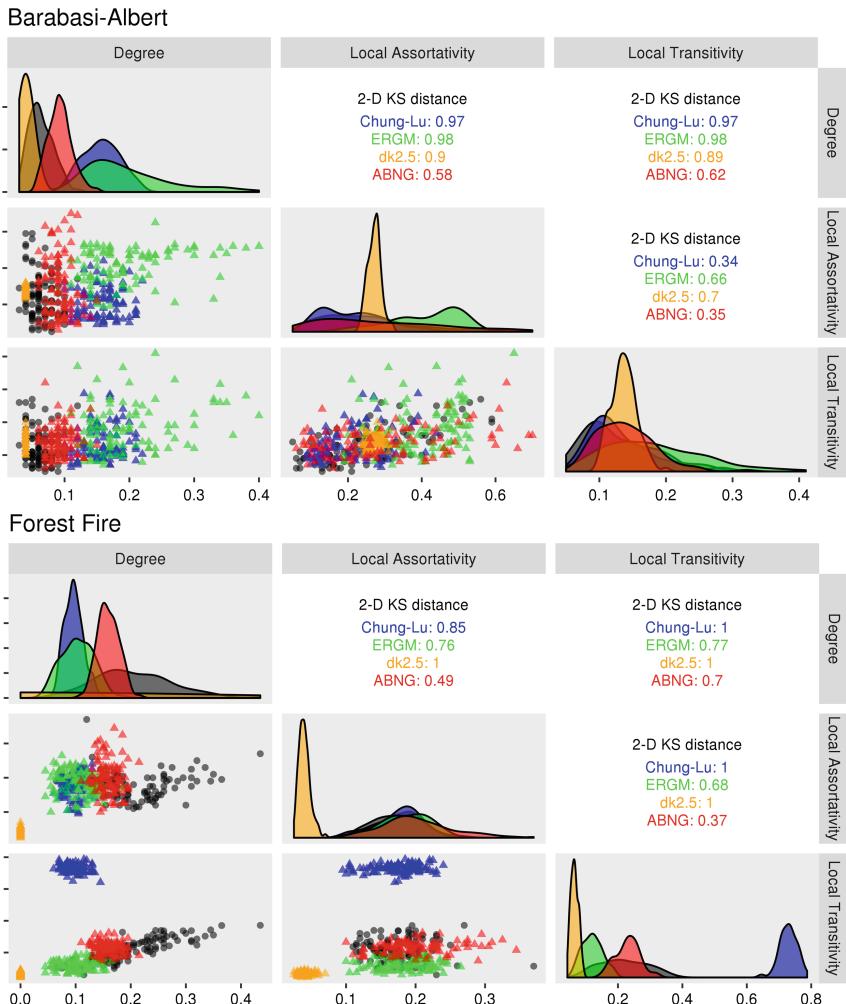


Fig. 2. Empirical evaluation of the ability of network models to approximate the ground truth system based on observation of a single network. The Barabási-Albert and Forest Fire models are used as the true generators.

in the network population of the true process in both the experimental settings. The plots also show that dk -random graphs, which are considered to be the state-of-art, fail to capture the variability of the true generative process and potentially over-fit the observed network. This leads us to question the fundamental idea behind dk -random graphs, i.e. whether exactly preserving the distribution of differently sized subgraphs of a given network leads to a good model for real-world networks. In fact, in most cases we see that the Chung-Lu model, by matching the degree distribution in expectation, outperforms dk -random graphs. These results highlight the need for evaluating the ability of a generative model to

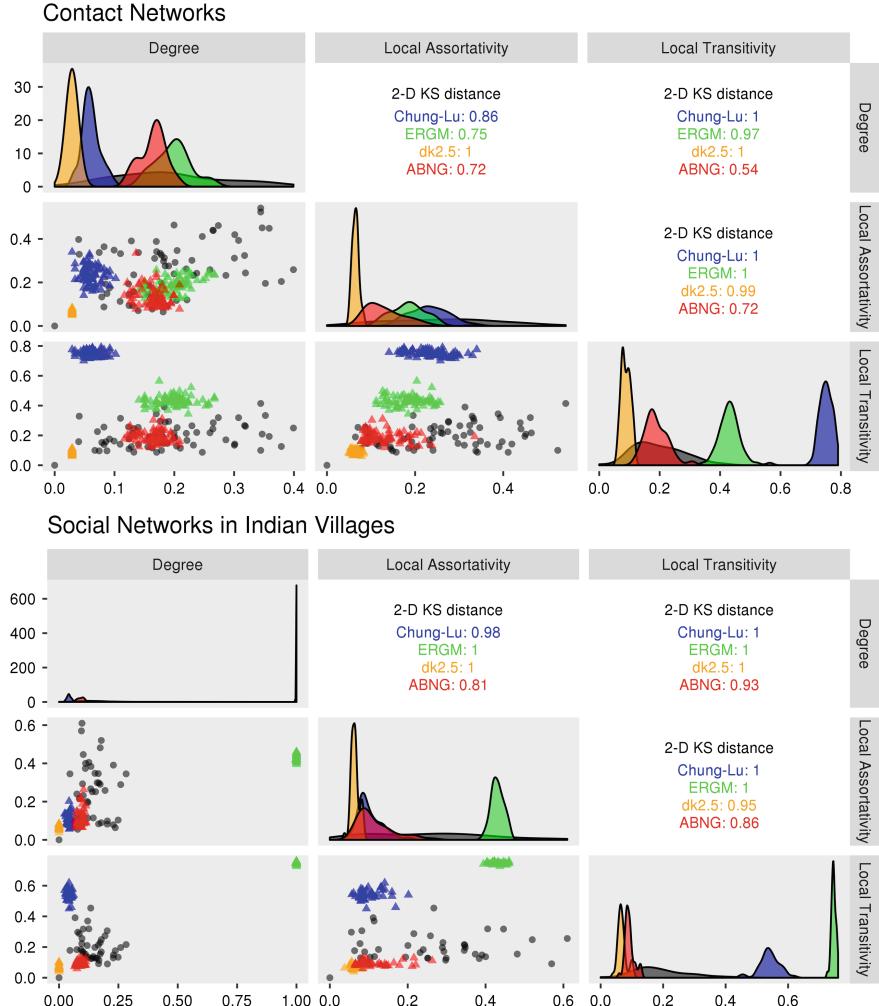


Fig. 3. Empirical evaluation of the ability of network models to approximate the ground truth system based on observation of a single network. Two real-world datasets were considered: contact networks, and social networks in Indian villages.

capture the distributional properties of a network population as comparing only with the observed network might produce misleading results.

4.1 Networks with Community Structure

While the network dissimilarity space defined in Sect. 3 works well for networks without communities, it will prove ineffective for networks with community structures, which is property seen in most real-world networks [10]. In this section, we extend the network dissimilarity space by adding a fourth dimension to compare

the community structures of two networks. To compare community structures, we first use a community detection algorithm¹ to decide the membership for each node, followed by evaluation of the normalized mutual information measure [8]. We also add the microcanonical stochastic block model (referred to as SBM-fit in the plots) [23] to our set of generative models and evaluate its ability to replicate the community structure of these networks.

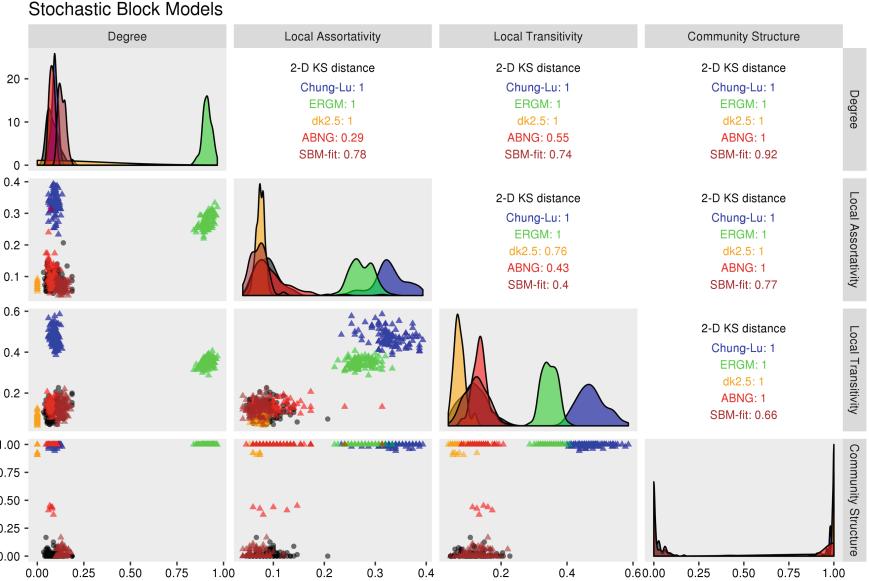


Fig. 4. Empirical evaluation of the ability of network models to approximate the ground truth system based on observation of a single network. The stochastic block model is used as the true generator, and the ability of different models to replicate the community structure is tested.

Again, we performed two different experiments to test the validity of our extended network dissimilarity space: (i) a controlled experiment where the true process is known, and (ii) set of real-world networks (with communities) that have most likely evolved from a common generative process. For the first case, we used the standard version of the stochastic block model [9,31] with 3 communities of different sizes, and the results can be seen in Fig. 4. As expected, ABNG performs well on the original measures, but fails to reproduce the community structure, while the fitted SBM is the most likely candidate capable of replicating the true process. This is an expected result as the four original models are not designed to create networks with communities. Fig. 5 shows the results for the network of Autonomous Systems, where only the fitted SBM was able to capture some of the features of the true process. Results presented in Fig. 5 show the

¹ Infomap community detection algorithm [26] was used in our experiments.

inability of the microcanonical block model to reproduce the local transitivity of the true generative process, thus creating an exciting direction for future research.

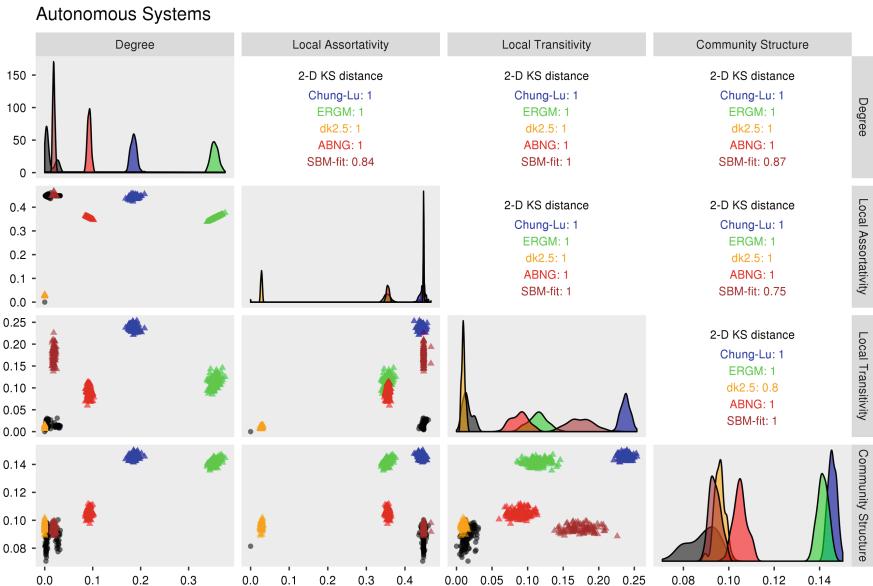


Fig. 5. Empirical evaluation of the ability of network models to approximate the ground truth system based on observation of a single network. The ability of different models to replicate the community structure of networks of autonomous systems is tested.

In summary, our empirical analysis has highlighted the importance of considering distributional properties of network populations for evaluating generative models of complex networks. This shows that there is an urgent need to rethink the network modeling problem and create new models that can reproduce the variability in the structural properties of network populations.

5 Conclusions

Traditional approaches for evaluating the ability of a network model to synthesize networks exhibiting real-world characteristics have compared the similarity of the synthesized networks with the observed network. While this approach assumes that the particular observation is representative of the underlying process that created the observation, it does not account for the natural variability of the population from which it is sampled. Our experiments have highlighted the importance of considering network populations for evaluating generative models. Although it is difficult to obtain data corresponding to network populations, we have shown that it is possible to establish a baseline test set to evaluate the

ability of a network model to capture the distribution of network populations. This test set can then be used for preliminary validation of a network model before it is used for drawing conclusions about real-world networks.

Acknowledgements. This material is based upon work supported by the National Science Foundation under Grant No. 1762633.

References

1. Alderson, D.L.: Catching the “network science” bug: insight and opportunity for the operations researcher. *Oper. Res.* **56**(5), 1047–1065 (2008). <https://doi.org/10.1287/opre.1080.0606>
2. Arora, V., Ventresca, M.: Action-based modeling of complex networks. *Sci. Rep.* **7**(1), 6673 (2017). <https://doi.org/10.1038/s41598-017-05444-4>
3. Avena-Koenigsberger, A., Goni, J., Sole, R., Sporns, O.: Network morphospace. *J. R. Soc. Interf.* **12**(103), 20140,881–20140,881 (2014). <https://doi.org/10.1098/rsif.2014.0881>
4. Banerjee, A., Chandrasekhar, A.G., Duflo, E., Jackson, M.O.: The diffusion of microfinance. *Science* **341**(6144), 1236,498 (2013)
5. Barabasi, A.L., Albert, R.: Emergence of scaling in random networks. *Science* **286**(October), 509–512 (1999). <https://doi.org/10.1126/science.286.5439.509>
6. Chung, F., Lu, L.: Connected components in random graphs with given expected degree sequences. *Annal. Comb.* **6**(2), 125–145 (2002)
7. Chung, F., Lu, L.: The average distances in random graphs with given expected degrees. *Proc. Nat. Acad. Sci.* **99**(25), 15879–15882 (2002)
8. Danon, L., Diaz-Guilera, A., Duch, J., Arenas, A.: Comparing community structure identification. *J. Stat. Mech. Theory Exp.* (09), P09,008–P09,008 (2005). <https://doi.org/10.1088/1742-5468/2005/09/P09008>
9. Faust, K., Wasserman, S.: Blockmodels: interpretation and evaluation. *Soc. Netw.* **14**(1–2), 5–61 (1992). [https://doi.org/10.1016/0378-8733\(92\)90013-W](https://doi.org/10.1016/0378-8733(92)90013-W)
10. Fortunato, S., Hric, D.: Community detection in networks: a user guide. *Phys. Rep.* **659**, 1–44 (2016). <https://doi.org/10.1016/j.physrep.2016.09.002>
11. Goldenberg, A.: A survey of statistical network models. *Foundations and trends®R. Mach. Learn.* **2**(3), 235–274 (2009). <https://doi.org/10.1561/2200000008>
12. Gutfraind, A., Meyers, L.A., Safro, : I.: Multiscale Network Generation. *arXiv:1207.4266*, 28 (2012)
13. Hunter, D.R.: Curved exponential family models for social networks. *Soc. Netw.* **29**(2), 216–230 (2007). <https://doi.org/10.1016/j.socnet.2006.08.005>
14. Isella, L., Stehlé, J., Barrat, A., Cattuto, C., Pinton, J.F., den Broeck, W.: What’s in a crowd? analysis of face-to-face behavioral networks. *J. Theoret. Biol.* **271**(1), 166–180 (2011)
15. Leskovec, J., Kleinberg, J., Faloutsos, C.: Graph evolution. *ACM Trans. Knowl. Discov. Data* **1**(1), 2-es (2007). <https://doi.org/10.1145/1217299.1217301>
16. Mahadevan, P., Krioukov, D., Fall, K., Vahdat, A.: Systematic topology analysis and generation using degree correlations. *ACM SIGCOMM Comput. Commun. Rev.* **36**(4), 135 (2006). <https://doi.org/10.1145/1151659.1159930>
17. Moreno, S., Neville, J.: an investigation of the distributional characteristics of generative graph models. In: Proceedings of the 1st Workshop on Information in Networks (2009)

18. Moreno, S., Neville, J.: Network hypothesis testing using Mixed Kronecker product graph models. In: Xiong, H., Karypis, G., Thuraisingham, B.M., Cook, D.J., Wu, X. (eds.), pp. 1163–1168 (2013)
19. Moreno, S., Neville, J., Kirshner, S.: Tied Kronecker product graph models to capture variance in network populations. *ACM Trans. Knowl. Discov. Data* **20**(3), 1–40 (2018). <https://doi.org/10.1145/3161885>
20. Newman, M.E.J.: The structure and function of complex networks. *SIAM Rev.* **45**(2), 167–256 (2003)
21. Orsini, C., Dankulov, M.M., Colomer-de Simón, P., Jamakovic, A., Mahadevan, P., Vahdat, A., Bassler, K.E., Toroczkai, Z., Boguñá, M., Caldarelli, G., Fortunato, S., Krioukov, D.: Quantifying randomness in real networks. *Nat. Commun.* **6**(May), 8627 (2015). <https://doi.org/10.1038/ncomms9627>
22. Peacock, J.A.: Two-dimensional goodness-of-fit testing in astronomy. *Month. Notices R. Astron. Soc.* **202**(3), 615–627 (1983)
23. Peixoto, T.P.: Nonparametric Bayesian inference of the microcanonical stochastic block model. *Phys. Rev. E* **95**(1), 1–21 (2017). <https://doi.org/10.1103/PhysRevE.95.012317>
24. Pinar, A., Seshadhri, C., Kolda, T.G.: The similarity between stochastic Kronecker and Chung-Lu graph models. In: Zaki, M., Obradovic, Z., Tan, P.N., Banerjee, A., Kamath, C., Parthasarathy, S. (eds.), *Proceedings of the 2012 SIAM International Conference on Data Mining*, pp. 1071–1082. Society for Industrial and Applied Mathematics, Philadelphia, PA (2012). <https://doi.org/10.1137/1.9781611972825.92>
25. Piraveenan, M., Prokopenko, M., Zomaya, : A.Y.: Local assortativeness in scale-free networks. *EPL (Europhysics Letters)* **84**(2), 28,002 (2008). <https://doi.org/10.1209/0295-5075/84/28002>
26. Rosvall, M., Bergstrom, C.T.: Maps of random walks on complex networks reveal community structure. *Proc. Nat. Acad. Sci.* **105**(4), 1118–1123 (2008). <https://doi.org/10.1073/pnas.0706851105>
27. Schieber, T.A., Carpi, L., Díaz-Guilera, A., Pardalos, P.M., Masoller, C., Ravetti, M.G.: Quantification of network structural dissimilarities. *Nat. Commun.* **8**(May 2016), 13,928 (2017). <https://doi.org/10.1038/ncomms13928>
28. Snijders, T.A.B., Pattison, P.E., Robins, G.L.: Specifications for exponential random graph models. In: *Sociological Methodology*, p. 44., Handcock, M.S.: New (2004). <https://doi.org/10.1111/j.1467-9531.2006.00176.x>
29. Strauss, D.: On a general class of models for interaction. *SIAM Rev.* **28**(4), 513–527 (1986)
30. Views, R.: University of Oregon route views project (2000)
31. Wasserman, S., Anderson, C.: Stochastic a posteriori blockmodels: construction and assessment. *Soc. Netw.* **9**(1), 1–36 (1987). [https://doi.org/10.1016/0378-8733\(87\)90015-3](https://doi.org/10.1016/0378-8733(87)90015-3)
32. Wasserman, S., Pattison, P.: Logit models and logistic regressions for social networks. *Psychometrika* **60**, 401–425 (1996)
33. Zheng, B., Wu, H., Kuang, L., Qin, J., Du, W., Wang, J., Li, D.: A simple model clarifies the complicated relationships of complex networks. *Sci. Rep.* **4**, 6197 (2014). <https://doi.org/10.1038/srep06197>

Multilayer Networks



Py3plex: A Library for Scalable Multilayer Network Analysis and Visualization

Blaž Škrlj^{1,2(✉)}, Jan Kralj¹, and Nada Lavrač^{2,3(✉)}

¹ Jožef Stefan International Postgraduate School, Jamova 39, 1000 Ljubljana, Slovenia

{blaz.skrlj,jan.kralj,nada.lavrac}@ijs.si

² Jožef Stefan Institute, Jamova 39, 1000 Ljubljana, Slovenia

³ University of Nova Gorica, Glavni trg 8, 5271 Vipava, Slovenia

Abstract. Real-life systems are commonly represented as networks of interacting entities. While homogeneous networks consist of nodes of a single node type, multilayer networks are characterized by multiple types of nodes or edges, all present in the same system. Analysis and visualization of such networks represent a challenge for real-life complex network applications. The presented Py3plex Python-based library facilitates the exploration and visualization of multilayer networks. The library includes a diagonal projection-based network visualization, developed specifically for large networks with multiple node (and edge) types. The library also includes state-of-the-art methods for network decomposition and statistical analysis. The Py3plex functionality is showcased on real-world multilayer networks from the domains of biology and on synthetic networks.

1 Introduction

Analysis and visualization of complex networks offers novel opportunities to study intractable systems, such as protein interaction networks, transportation networks or social networks [4, 23]. As this vibrant research field offers novel tools at an increasing pace, development of freely available, scalable software resources is becoming a relevant research direction. Despite many existing tools for analysis and visualization of homogeneous networks, i.e. networks with only a single node type and no annotated edges, not many tools consider multilayer (heterogeneous) networks. In multilayer networks, many different types of annotations are taken into account, including e.g., relation-labeled edges, multiple node types etc. Such networks are considered, for example, when multiple layers of biological information (e.g., protein-protein, gene-gene interactions etc.) are available and need to be taken into account when studying diseases [5, 16]. In this work we consider networks as multilayer when they contain at least two types of nodes and/or edges.

This paper presents Py3plex, a new Python library for analysis and visualization of large, heterogeneous (and homogeneous) networks. The visualization suite simplifies displaying of multilayered networks and network communities as well as network embeddings [9]. Further, current version of Py3plex implements a state-of-the-art procedure for converting heterogeneous networks to homogeneous networks [14]. Finally, the library includes a set of subroutines for partition enrichment—the process of learning qualitative explanations relating individual partitions (e.g., communities) to expert-curated domain knowledge [21].

This paper is structured as follows. First, we present the related work on multilayer network analysis, followed by the description of the Py3plex architecture, its comparison with the state-of-the-art approaches, and the proposed multilayer network visualization approach. We showcase visualization performance od Py3plex on selected real-world biological networks, and present the comparison of Py3plex results with the results of the state-of-the-art Pymnet visualization library on synthetic networks. We empirically demonstrate that Py3plex, despite being a lightweight analysis library, offers a novel method for visualization of multilayer networks, performing significantly faster than the current alternatives.

2 Related Work

This section presents the state-of-the-art network analysis libraries, relevant to the development of Py3plex. The most common approaches to network analysis can be split into two groups: GUI-based solutions and API-based solutions.

The GUI-based solutions include Cytoscape [19] and Gephi [2]. Cytoscape [19] is one of the largest network analysis projects to date. It supports custom manipulation of the loaded network, and is hence flexible both in terms of network visualization as well as analysis. The Gephi [2] suite offers a similar set of functionalities, yet it is known for better visualization capabilities—a key aspect of the modern network science. The two solutions are mostly used in the final step of a network analysis project, where pre-computed node properties are used as part of the input.

The API-based solutions, which provide programmatic access from popular languages (such as Python, R, JavaScript, C++), are preferred when the entire network analysis and visualization should be carried out in the same environment. The NetworkX library [10] is prevalent for the Python environment, while the igraph library [17] is commonly used among R users. Further, C, and C++ alternatives include SNAP [15] and Boost Graph Library (BGL) [20]. Compared to GUI-based analysis, API-based approaches result in a series of high-level function calls, which generate the desired output. Such approaches are commonly used for analysis when either the networks are large or the number of networks under consideration is high.

The aforementioned approaches focus on homogeneous networks consisting of single node (and edge) types. On the other hand, recent advances in multilayer network analysis prove that additional information associated with node and

edge types provides insights regarding network structure and dynamics. Multi-layer networks can be represented as higher order tensors [7], encoding inter- as well as intra-layer connections of varying intensity. In this paper we focus on state-of-the-art implementations of this formalism, their functionality and some of their drawbacks. The currently used libraries for visualization and analysis of multilayer networks include:

- libraries for the Python environment Pymnet¹ [13] and MultinetX² [1], and
- library for the R environment Muxviz³ [6].

Detailed analysis of these approaches is presented in Sect. 4, where they are compared to the proposed Py3plex library.

3 Py3plex Library Architecture

This section explains the proposed Py3plex library’s architecture, followed by the description of individual components and subroutines.

A high-level organization of the Py3plex library is shown in Fig. 1. The library includes methods for parsing and converting graphs from and to various formats, while the core library includes three modules:

- **The visualization module** consists of different subroutines used for network visualization of multilayer and single layer networks. Detailed description of the in-house developed multilayer visualization is given in Sect. 5.
- **The wrappers module** is implemented as follows. As many procedures are given as standalone executables, Py3plex offers a set of wrapper subroutines, useful for calling external state-of-the-art algorithms, for example, highly optimized network embedding routines [9] as well as the InfoMap community detection algorithm [18].
- **The algorithms module** includes implementations of many commonly used algorithms, such as: Louvain community detection [3], node ranking, network statistics and the recently introduced network topology enrichment [21].

All the modules are built in an extensible manner, allowing for new routines to be easily added. As Py3plex is built on top of the NetworkX [10], network operations can include any of the methods primarily designed for homogeneous networks. This functionality provides flexibility when implementing novel multilayer network analysis algorithms.

¹ <http://www.mkivela.com/pymnet/>.

² <https://github.com/nkoub/multinetx>.

³ <http://muxviz.net/>.

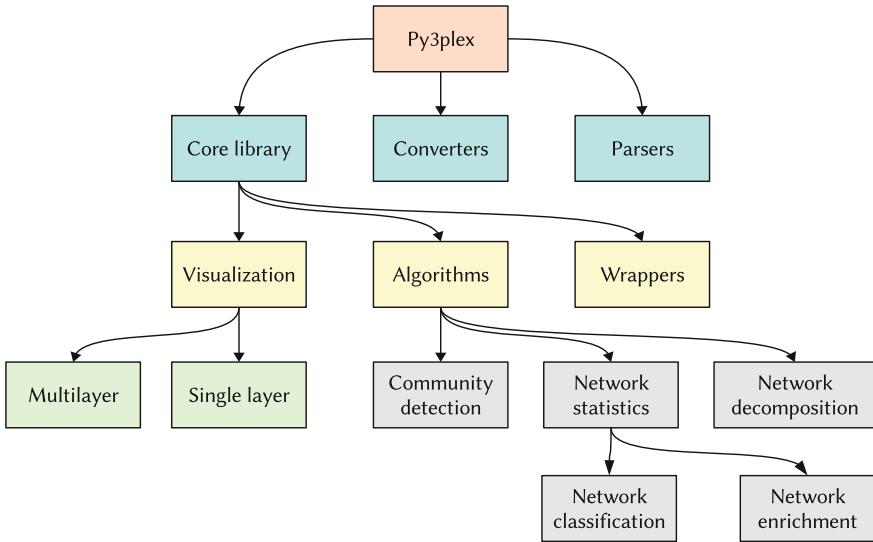


Fig. 1. The Py3plex library architecture. The library is organized in hierarchical manner—a set of core data structures (blue) is used in algorithms for visualization and analysis of multilayer networks.

4 Py3plex Comparison to Other Libraries

In this section we compare the functionality of different multilayer network analysis libraries. Each library offers some functionality that is not available in other libraries, hence no library is exhaustive. We evaluate different aspects, ranging from computational efficiency to general richness in terms of various analysis functions. The results are summarized in Table 1.

To ensure sufficient speed of execution, Pymnet [13], Py3plex and MultinetX [1] include subroutine implementations in C (via Cython), but also Numpy [22] and Scipy [12] libraries for optimized linear algebra-based computation. The R-based MuxViz [6] uses the Octave library [8] for efficient array-based operations.

In terms of visualization, all compared packages include at least some form of network visualization. Apart from 2D layouts, The MuxViz and Pymnet libraries also support 3D layouts. As discussed in Sect. 5, Py3plex offers a new approach to multilayer network visualization, whereas MultinetX, for example, displays the supra-adjacency matrix [7] of the network in a block-diagonal manner. The multitude of different multilayer network visualization applications indicates no single type of visualization is optimal for a given task, as each visualization is optimal for a particular range of network size and density.

Apart from MultinetX, all other libraries include different methods for network aggregation and decomposition. A similar compendium of functionality is offered by Pymnet and MuxViz. All packages offer intuitive access to a network's supra-adjacency matrix, and hence provide many opportunities for implementation of computationally efficient multilayer network analysis algorithms.

Table 1. Comparison of multilayer network analysis libraries.

	Py3plex	Pymnet [13]	MuxViz [6]	MultinetX [1]
Core features				
Programming language	Python 3 and C (via Numpy and Cython)	Python 3 and C (via Numpy)	R	Python 3 and C (via Numpy)
Basic statistics	✓	✓	✓	✓
Visualization of large networks	✓	-	✓	-
Visualization in 3D	-	✓	-	✓
Aggregation/decomposition	✓	✓	✓	-
Random graph generators	✓	✓	✓	✓
Adjacency matrix manipulation	✓	✓	✓	✓
Tensorial manipulation	✓	✓	✓	✓
Additional features				
Machine learning for multilayer networks	✓	-	-	-
GUI version	-	-	✓	-
Community detection	✓	-	✓	-
Isomorphisms	✓	✓	-	-
Node ranking	✓	✓	✓	✓
Semantic topology enrichment	✓	-	-	-
Temporal networks	-	-	✓	✓
Network embedding functionality	✓	-	-	-

Py3plex also leverages full functionality of the recently developed HINMINE [14] methodology for heterogeneous network decomposition. As standard aggregation schemes yield a homogeneous network by summing over edges in individual layers, HINMINE-based aggregation is based on frequency of relation-annotated directed paths of length two between the target node type. Compared with standard aggregation, HINMINE is thus suitable for situations where domain-knowledge was used for annotating the network edges. Further, Py3plex also includes basic aggregation subroutines. All libraries include methods for obtaining quick insights into a network’s structure. The Pymnet and Py3plex are currently the only libraries supporting different isomorphism algorithms for evaluating network similarity.

In terms of other functionalities, we observe the following. MuxViz also supports GUI-based analysis, as, once installed, it can be executed locally within the browser as a standalone software. To our knowledge MuxViz is the only alternative for GUI-based multilayer network exploration.

We observe Pymnet offers one of the most intuitive API interfaces for manipulation of tensor representations of multilayer networks, such as slicing, indexing etc. In comparison, MultinetX and MuxViz offer similar functionality, whereas Py3plex operates on attribute-rich list-based representations of a network and is not necessarily suitable for all multilayer slicing tasks.

Apart from network analysis and visualization capabilities, Py3plex is the only library which also supports various forms of learning from multilayer net-

works. It provides the methodology for semantic enrichment on complex networks. The main objective in this emerging field is to associate previously known domain knowledge with topological structures of a network. For example, the functional characterization of a network’s communities can only be assessed using curated domain knowledge. We refer to the use of such knowledge to understand network-topological properties as network enrichment. Currently, Py3plex is the only library that supports both rule-based enrichment as well as standard Fisher’s exact test-based enrichment, both recently introduced as part of the Community-based Semantic Subgroup Discovery methodology [21]. Further, classification using Personalized PageRank vectors is also supported [14].

Current implementation of Py3plex contains wrapper routines for widely used network embedding algorithms, such as Node2vec and similar [9]. Such functionality is not offered in any other libraries.

Overall, Py3plex thus offers novel algorithms for understanding network structure, an intuitive interface for manipulation of multilayer networks as well as network decomposition and aggregation, however, the primary features of the current version of Py3plex are primarily network visualization and spatially efficient network manipulation (linear in terms of nodes and edges). This functionality offers additional state-of-the-art performance, not observed in the other libraries. We prove this claim in the following sections, where we first present the visualization algorithm, followed by empirical evaluation of its performance on a set of artificial random multilayer networks.

5 Multilayer Network Visualization

One of the key contributions of this paper is a novel method for visualization of multilayer networks. In this section we present the proposed visualization method and its implementation. Next, we evaluate the method’s performance on a series of random networks, where we also compare the results with Pymnet’s network visualization capabilities.

5.1 Visualization Methodology and Implementation

The implementation of the proposed layout algorithm operates in three main steps, described below in detail.

- **Intra-layer layout computation.** First, subnetworks consisting of same-typed nodes and edges between them are considered. For each node type, we compute a fast, force-directed layout on the single-type subnetwork. This results in (x, y) coordinate pairs for individual nodes which are normalized so that both coordinates are between 0 and 1.
- **Multilayer network drawing.** In the second step, the actual coordinates of each point are then computed by separating each consecutive layer from the preceding layer by exactly a single coordinate unit on both axes. Consequently, nodes are drawn around a diagonal line based on their types. As nodes in different layers are separated by a coordinate unit, each subnetwork corresponding to different node type is drawn in a separate region—this

ensures no overlap between different layers is possible, and that each layer includes a network with its own local organization, independent of the inter-layer connections.

– **Inter-layer edge drawing.** The final step involves drawing of inter-layer edges, which are represented as arches connecting different nodes. One of the main problems we faced during the implementation of this step is edge positioning and parameterization, as there are many different options for drawing an arch between two nodes. We consider three possible scenarios in terms of inter-layer edge drawing:

1. edges are only on the upper part of the layer diagonal,
2. edges are only on the bottom part of the layer diagonal,
3. edges are on both sides of the diagonal projection.

To further explain the inter-layer edge drawing step of Py3plex, We first discuss how edges are drawn when only the upper part is considered, and continue with the extension to bottom-only and both parts of the diagonal projection.

An arch connecting nodes $A = (x_1, y_1)$ and $B = (x_2, y_2)$ is drawn through an artificially introduced node $C = (x_f, y_f)$, which lies between A and B at the desired part of the diagonal. Point C is calculated by taking the midpoint between A and B and scaling its y coordinate by a factor of τ (a parameter of the visualization method) from the line.

Once A , B and C are obtained, the points A and B are connected by a parabolic arch that passes through all three points. Even though the current implementation constructs inter-layer edges using interpolation over three points, Py3plex supports adding an arbitrary number of intermediary points, in which case cubic arch interpolation is used to produce the line between A and B . In our experiments, however, we show that even a single intermediary point allows for clear visualization.

The proposed setting parameterized with τ offers simple manipulation of inter-layer edges, as

1. when $\tau > 1$, the arch will be located above the diagonal,
2. when $\tau = 1$, the arch will be a line between the two nodes (a third point on $f(x)$),
3. when $\tau < 1$, the arch will be located below the diagonal.

We further propose a heuristic for automatically determining an arch's position, i.e. whether an arch should be located above or below the diagonal. We achieve this as follows. Given A and B as defined in the previous paragraph, the arch is located above the diagonal if and only if $\text{int}(c) > c$, where $\text{int}(c)$ represents the integer representation of coordinate c . Integer-based rounding works, as layers are separated by exactly a single coordinate unit. All operations for arch computation, scaling and transformation are vectorized for better performance. Should the network appear incomprehensible, natural logarithms are applied to node representations—filled circles based on individual node degrees—which simplifies visualization of denser networks.

Individual, single-layer networks are derived from the existing NetworkX graph library [10] hence the rich set of functionalities available in NetworkX can be used. Py3plex provides the means to customize majority of network properties, including edge shapes and colors, individual layer layouts, node sizes and colors, and overall network organization. Computationally expensive operations are vectorized using the Numpy numeric library [22]. The Barnes-Hut approximation of the force-directed layout algorithm implementation, which was transpiled from Python to C, is used as the default option during visualization [11]. Py3plex can easily visualize more than ten layers with tens of thousands of nodes. Compared to existing solutions, diagonal projection of multiple layers enables visualization using standard layout algorithms, with additional specification of inter-layer edges.

Example visualization using the proposed Py3plex library is shown in Fig. 2, where an alternative visualization using a single layer force-directed layout of the whole network is shown for comparison.

5.2 Comparison of Py3plex and Pymnet Visualization Performance

In this section we present the results for the times needed to obtain a visualization of networks of different complexities. Even though Pymnet does not support the diagonal projection and Py3plex does not support the default 3D linear projections, both methods are useful.

Note that individual node types are drawn along the diagonal axis. Edges can exist either on the intra-layer as well as inter-layer levels. Curves, representing inter-layer edges are adapted based on the distance between the two layers under consideration. Such visualization offers quick insights into inter- and intra-layer dynamics, which can not be detected in hairball (spring-based, single-layered) plots for visualization of different aspects of multilayer networks. The main aim of this section is to demonstrate that Py3plex offers better support for visualization of larger networks.

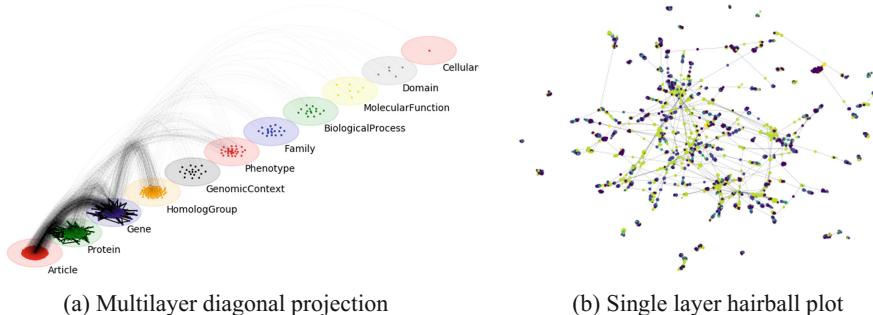


Fig. 2. Comparison between a multilayer visualization and single layer (hairball) layout supported in Py3plex. The proposed diagonal projection is shown in subfigure (a), and the hairball plot in (b). The inter-layer connectivity is more clearly expressed in (a) than in (b). Further, organization between the nodes of the same type can not be observed in subfigure (b), as the layout algorithm considers all of the connections. Py3plex supports both visualization styles.

The experimental setting for this task was designed as follows. Random multilayer Erdős-Rényi (ER) networks were generated using the Pymnet [13] library. Such random networks are parameterized using parameter N corresponding to the total number of nodes, parameter L corresponding to the number of layers, and parameter p corresponding to the re-wiring probability. We generated the networks in the following parameter ranges:

$$\begin{aligned} p &\in \{0.05, 0.1, 0.2, 0.3\}, \\ N &\in \{5, 10, 20, 50, 80, 100\}, \\ L &\in \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}. \end{aligned}$$

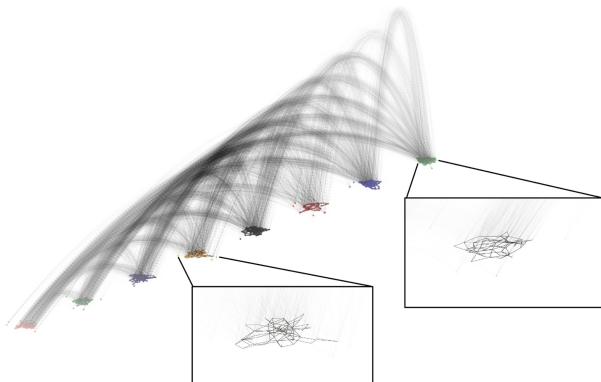
Once generated, the time needed for visualization was recorded. We compared visualization times of Pymnet and Py3plex, as the remaining Python-based alternative, MultinetX, does not support drawing of coupled edges.

We show the final results of our experiment in the form of box plots, where $|N|$ or $|E|$ is plotted on the x -axis and the time needed is plotted on the y -axis (Fig. 3). Networks with more than 100 nodes were not considered for this benchmark, as it took Pymnet more than two hours for visualization. Nonetheless, Py3plex was able to visualize a network with $|N| = 4,000$ and $|E| = 18,600$ under two hours, even though the obtained network is not necessarily useful for visualization purposes. The machine used for benchmark testing was an off-the-shelf Lenovo y510p laptop.

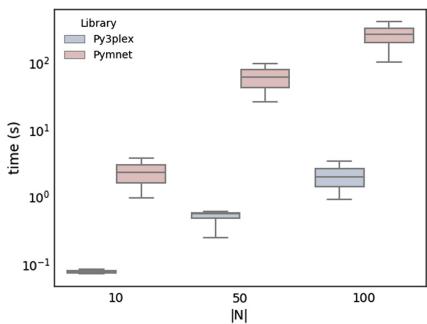
6 Conclusions and Further Work

We have developed and implemented Py3plex, a network analysis and visualization library focused on multilayer networks. Apart from most of the functionality offered by the state-of-the-art libraries, Py3plex introduces a novel visualization, suitable for larger multilayer networks, as to our knowledge, there currently do not exist any methods that can visualize networks composed of thousands of nodes along with their intra- as well as inter-layer organization. The Py3plex is suitable for the development of novel algorithms, as it offers fast lookup and indexing routines, which scale to networks consisting of multiple layers with hundreds of thousands of edges.

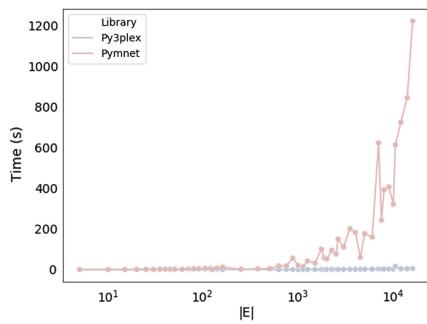
As the Py3plex analysis suite is by no means exhaustive, further work includes the implementation of network dynamics analysis methods as well as the more efficient, GPU-based random samplers. Further, as Py3plex, apart from offering novel visualization capabilities, aims to bridge the gap between machine learning approaches and complex networks, it does not yet include extensive tensor manipulation, unfolding, and construction routines offered in e.g., the Pymnet library. Finally, as part of the further work we believe Py3plex should be tested on more non-synthetic networks.



(a) Py3plex-based visualization of a random network with 6,235 edges and eight layers. The network can be explored interactively (inset images).



(b) Visualization time with respect to the number of nodes.



(c) Visualization time with respect to the number of edges.

Fig. 3. Visualization time comparison on random ER network. Even though initial visualization does not necessarily offer intuitive representation of a given network (a), Py3plex offers interactive exploration of the network, where users can zoom-in into parts of interest—as an example, see the inset images in (a). The time Pymnet needed to visualize the network does not appear linear in terms of the number of nodes (b) and edges (c), hence networks with more than 100 nodes were not considered for this benchmark.

7 Availability

Py3plex as well as all datasets used for this paper are freely accessible at <https://github.com/SkBlaz/Py3plex>, where minimal working examples of the Py3plex functionality are also offered.

Acknowledgements. This work was financially supported by the Slovenian Research Agency (ARRS) grants *HinLife: Analysis of Heterogeneous Information Networks for Knowledge Discovery in Life Sciences* (J7-7303) and *Semantic Data Mining for Linked Open Data* (financed under the ERC Complementary Scheme, N2-0078).

References

1. Amato, R., Kouvaris, N.E., San Miguel, M., Díaz-Guilera, A.: Opinion competition dynamics on multiplex networks. *New J. Phys.* **19**(12) (2017)
2. Bastian, M., Heymann, S., Jacomy, M., others: Gephi: an open source software for exploring and manipulating networks. In: International AAAI Conference on Web and Social Media Third International AAAI Conference on Weblogs and Social Media, vol. 8, pp. 361–362 (2009)
3. Blondel, V.D., Guillaume, J.L., Lambiotte, R., Lefebvre, E.: Fast unfolding of communities in large networks. *J. Stat. Mech. Theor. Exp.* **2008**(10), P10,008 (2008)
4. Boccaletti, S., Bianconi, G., Criado, R., del Genio, C., Gómez-Gardeñes, J., Romance, M., Sendiña-Nadal, I., Wang, Z., Zanin, M.: The structure and dynamics of multilayer networks. *Phys. Rep.* **544**(1), 1–122 (2014)
5. Cho, D.Y., Kim, Y.A., Przytycka, T.M.: Chapter 5: network biology approach to complex diseases. *PLOS Comput. Biol.* **8**(12), 1–11 (2012)
6. De Domenico, M., Porter, M.A., Arenas, A.: MuxViz: a tool for multilayer analysis and visualization of networks. *J. Complex Netw.* **3**(2), 159–176 (2015)
7. De Domenico, M., Solé-Ribalta, A., Cozzo, E., Kivelä, M., Moreno, Y., Porter, M.A., Gómez, S., Arenas, A.: Mathematical formulation of multilayer networks. *Phys. Rev. X* **3**(4) (2013)
8. Eaton, J.W., Bateman, D., Hauberg, S.: GNU octave version 3.0. 1 manual: a high-level interactive language for numerical computations. SoHo Books (2007)
9. Grover, A., Leskovec, J.: Node2vec: scalable feature learning for networks. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD ’16, pp. 855–864. ACM, New York, NY, USA (2016)
10. Hagberg, A., Swart, P., S Chult, D.: Exploring network structure, dynamics, and function using NetworkX. In: Proceedings of the 7th Python in Science Conference (SciPy)(2008)
11. Jacomy, M., Venturini, T., Heymann, S., Bastian, M.: ForceAtlas2, a continuous graph algorithm for handy network visualization designed for the Gephi software. *Plos One* **9**(6), e98,679 (2014)
12. Jones, E., Oliphant, T., Peterson, P., et al.: SciPy: open source scientific tools for Python (2001). <http://www.scipy.org/>
13. Kivelä, M., Arenas, A., Barthelemy, M., Gleeson, J.P., Moreno, Y., Porter, M.A.: Multilayer networks. *J. Complex Netw.* **2**(3), 203–271 (2014)
14. Kralj, J., Robnik-Šikonja, M., Lavrač, N.: HINMINE: heterogeneous information network mining with information retrieval heuristics. *J. Intell. Inf. Syst.* **50**(1), 29–61 (2018)
15. Leskovec, J., Sosić, R.: Snap: a general-purpose network analysis and graph-mining library. *ACM Trans. Intell. Syst. Technol.* **8**(1), 1:1–1:20 (2016)
16. Milano, M., Guzzi, P.H., Cannataro, M.: HetNetAligner: a novel algorithm for local alignment of heterogeneous biological networks. In: Proceedings of the 2018 ACM International Conference on Bioinformatics, Computational Biology, and Health Informatics, BCB ’18, pp. 598–599. ACM, New York, NY, USA (2018)
17. Nepusz, G., Csárdi, G.: The igraph software package for complex network research. *Complex Syst.* **1695**(5), 1–9 (2006)
18. Rosvall, M., Axelsson, D., Bergstrom, C.T.: The map equation. *Eur. Phys. J. Spec. Top.* **178**(1), 13–23 (2009)

19. Shannon, P.: Cytoscape: a software environment for integrated models of biomolecular interaction networks. *Genome Res.* **13**(11), 2498–2504 (2003)
20. Siek, J.G., Lee, L.Q., Lumsdaine, A.: Boost Graph Library: The User Guide and Reference Manual. Pearson Education (2001)
21. Škrlj, B., Kralj, J., Vavpetič, A., Lavrač, N.: Community-based semantic subgroup discovery. In: Appice, A., Loglisci, C., Manco, G., Masciari, E., Ras, Z.W. (eds.) *Proceedings of New Frontiers in Mining Complex Patterns*, pp. 182–196. Springer International Publishing (2018)
22. Walt, S.V.D., Colbert, S.C., Varoquaux, G.: The numpy array: a structure for efficient numerical computation. *Comput. Sci. Eng.* **13**(2), 22–30 (2011)
23. Wang, Z., Wang, L., Szolnoki, A., Perc, M.: Evolutionary games on multilayer networks: a colloquium. *Eur. Phys. J. B* **88**(5), 124 (2015)



Morphogenesis of Complex Networks: A Reaction Diffusion Framework for Spatial Graphs

Michele Tirico^(✉), Stefan Balev, Antoine Dutot, and Damien Olivier

Normandy University, UNIHAVRE, LITIS, 76600 Le Havre, France
michele.tirico@univ-lehavre.fr

Abstract. A large variety of real systems are composed by entities in relationships which can be represented by networks. In many of these systems, elements are embedded in the space and location information impacts properties and evolution. Local interactions between elements generate different kinds of equilibrium and often indicate a self-organized behaviour. In this paper we are interested in essential mechanisms behind morphogenesis of spatial networks such as street networks. We propose a multi-layer model, where a reaction-diffusion mechanism governs the growth of spatial networks. We study its evolution with some metrics.

Keywords: Spatial networks · Reaction diffusion model
Morphogenesis · Gray-Scott model · Network models

1 Spatial Networks: Properties and Models

Network theory is a relevant methodology to represent, describe and analyze complex systems. It provides a powerful representation for many social, biological and technological applications [7, 9, 22]. In some cases, geographical localization plays a crucial role to describe essential properties. Compared to other complex networks, spatial networks often show higher robustness, less peaked and more compact degree distribution [3]. Spatial networks are widely used to represent venation pattern of leaves [20], insect colonies [26] and street networks[31]. In urban studies, centrality metrics are useful to identify loops [23], patterns [18], accessibility [28] and its historical evolution is largely studied with topological indicators [24]. The last ones can characterize some static properties but it is difficult to catch the dynamics with them. Our aim is to explore a simple model able to capture the development of a spatial graph under constraints.

The location of elements and the spatial dependencies should be taken carefully into account in order to identify an appropriate methodology to generate spatial networks. Accordingly, investigating essential evolution mechanisms

of spatial networks is therefore interesting both for purely graph theoretical research and for a large number of real case studies [4]. In last decades, following the concept that the growth of spatial networks is subject to certain optimization processes [14], various authors proposed models that yield to minimize metrics [16], reproduce emergent characteristics [6] or find compromise between antagonist properties [10]. Using a global optimization process, interesting works are developed in order to replicate urban growth. For example, geometrical optimization of global elements of urban network used in [12] reproduces a coherent and realistic growth of cities.

Although global approaches are largely useful to model complex system evolution, in many applications of spatial networks, local dynamics are predominant and must be considered to produce emergence of global properties. The formation of stable structures is often a consequence of elementary local interactions. They are dependent on spatial evolution of elements in a small neighbourhood [25]. A class of structures in dynamic equilibrium emerges as a result of local interactions, not through external and global control. Through a local-driven interaction approach, the network grows spatially and temporally in a coherent way. In many applications, spatial and temporal coherence is strongly required: for instance, in urban studies, a typical street network is growing as a single connected component [1].

The generation of disconnected sub-graphs is often not an appropriate representation of growth for some applications. For example, vascular systems or insect gallery networks must always be connected. Moreover, in venation leaves and street growth pattern formation, the network evolution thanks to local interactions produces networks which often respect the planarity condition. Roughly, a planar graph is a spatial graph that can be drawn in a plane without edge crossing [13]. Growing planar graph models with focus on local interactions are proposed in urban studies: Barthelemy et al. [5] describe a model of street network growth with local optimization of shortest connection road and an handle formation of loops. In Rui et al. [29] model, competition of nodes added step-by-step produces a stable self-regulated street network.

The study of spatial network evolution is a fundamental topic with many applications. The purpose of this paper is to formalize a minimal model, using different kind of local interactions and where evolution is self-regulated, in order to investigate essential mechanisms of morphogenesis and capture the complexity of the system. In Sect. 2 we expose the framework of our model, its essential elements and interaction mechanisms. In Sect. 3 we show and discuss early results through some metrics.

2 Model Description

Morphogenesis is the process of development of the forms of a living organism and by extension of some human organizations like cities. In 1952, Alan Turing has proposed a mathematical model [30] of spatial structure morphogenesis. In this model there are chemical compounds which react together and diffuse. The result is the production of evolving patterns in a continuous medium. Two molecules (A and B) called morphogens interact: A catalyzes its own production but also the production of B. In the same time B inhibits the production of A. B diffuses faster than A.

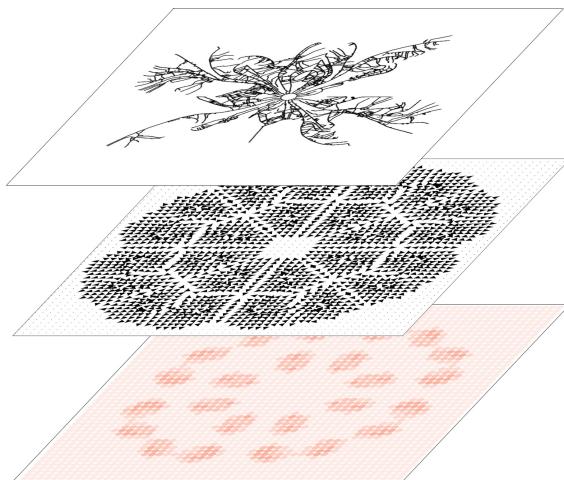


Fig. 1. The model is composed of three interdependent layers: a regular grid governed by a reaction-diffusion mechanism (bottom), a vector field applied to a set of moving seeds (center) and a growing spatial network (top).

Due to the auto-catalytic reaction and different diffusion rates of morphogens, this essential framework is able to produce a variety of spatio-temporal patterns. Through this minimal framework, many studies had been proposed to explain various spatial phenomena in continuous media, fish skin, sea shells and bacteria aggregation [21].

Morphogens, in a real case study such as the urban growth, should represent a set of spatial properties, for example natural constraints, human behaviours, socio-economical effects. The different concentration of these evolving properties impacts the evolution of the street network through constraints or attracting spots located in the space. A set of forces, modelled as a vector field, governs the

morphogenesis of the network. Therefore in this paper, we propose a growing spatial network generated by a discrete self-regulated pattern formation model. The model is composed by three interdependent layers:

- a reaction-diffusion system applied to a square lattice,
- a dynamic vector field applied to a set of moving seeds,
- and a growing spatial graph (Fig. 1).

The first layer of our model dynamically governs the morphogenesis of the growing graph through spatial evolution of two morphogens in two dimensional space. Among different formalizations of the Turing concept, the Gray-Scott model, thanks to the introduction of a third mechanism (to increase or decrease the morphogen concentration), produces a wealth of distinct patterns [15]. The first layer of our model is inspired by latter studies. The continuous system of equations of Gray-Scott model, represents the evolution of morphogen concentration:

$$\begin{cases} \frac{\partial a}{\partial t} = D_a \nabla^2 a - ab^2 + f(1 - a) \\ \frac{\partial b}{\partial t} = D_b \nabla^2 b + ab^2 - (f + k)b \end{cases} \quad (1)$$

where $a = a(X, t)$ and $b = b(X, t)$ indicate the concentration of two morphogens (a and b) at time t at position $X = (x, y)$ and the diffusion rate is computed through Laplacian terms $\nabla^2 a$ and $\nabla^2 b$, incremented by D_a and D_b for a and b respectively. Free parameters f, k represent diminished and increment terms, both proportional to concentration of morphogens.

Based on the microscopic master equation (1) we use a discrete equation system to reproduce the evolution of each morphogens. The reaction-diffusion layer $C = [a(X, t), b(X, t)]$ in our model is a lattice of square cells, in which morphogen concentration in each cell is calculated as follow:

$$\begin{cases} a(X, t + 1) = a(X, t) + D_a \sum_{Y \in N(X)} (a(X, t) - a(Y, t)) - a(X, t)b(X, t)^2 + f(1 - a(X, t)) \\ b(X, t + 1) = b(X, t) + D_b \sum_{Y \in N(X)} (b(X, t) - b(Y, t)) + a(X, t)b(X, t)^2 - (f + k)b(X, t) \end{cases} \quad (2)$$

where $N(X)$ is the Moore neighbourhood of the cell at position X [19]. The diffusion rate is obtained through the Fick's law, to ensure a local mass equilibrium [8].

The different concentration of morphogens can be seen as a set of attraction or repulsion spots for a set of moving elements, in the following called seeds. A force depending on the distance and the morphogen concentration moves the seeds. Accordingly, in a discrete representation, we compute a vector field $B = (\mathbf{b}, t)$ for the morphogen $b(X, t)$. At each cell on layer C is associated the vector:

$$\mathbf{b}(X, t) = \sum_{Y \in N(X)} g \frac{b(X, t)b(Y, t)}{d(XY)^\theta} \hat{XY} \quad (3)$$

where $d(XY)$ is the euclidean distance between centres of cells located in X and Y , \hat{XY} the relative unit vector, g is a gravitational constant and θ is a distance exponent. The second layer is completed by an evolving set of moving seeds $S = [s_1(X_1, t), \dots, s_n(X_n, t)]$. During the simulation, the vector field is applied to the moving seed set and each element follows the evolution of the gradient of morphogen concentration. To each seed we apply a vector r (the sum of corresponding nearest vectors) and the evolution of positions of the seeds represents the latest layer of the model. More precisely, we generate a growing spatial graph $G = (V(X, t), E(t))$, a mathematical entity composed of an evolving set V of vertices embedded in a two dimensional space, and a set E of undirected edges [17]. Using rules exposed in the following, the graph grows spatially and temporally in a coherent way, without edges crossing.

2.1 Spatial Interactions: The Morphogenesis of the Network

The dynamic vector field affects the position of moving seeds, which generate the graph. However, the morphogenesis of the network is not only governed by the reaction-diffusion mechanism. During the simulation, the network grows and its configuration impacts its own morphogenesis.

We start with a small connected graph with a seed in each node. At each seed corresponds a unique vertex in the growing graph. At each time step, we apply the corresponding vector r to each moving seed and calculate its future position X_f as X_t moved by r . In this way we take into account the evolution of morphogen concentration near the seeds. Concerning the seed movement, several cases can occur; given u a node at position X_t and v a future node at position X_f :

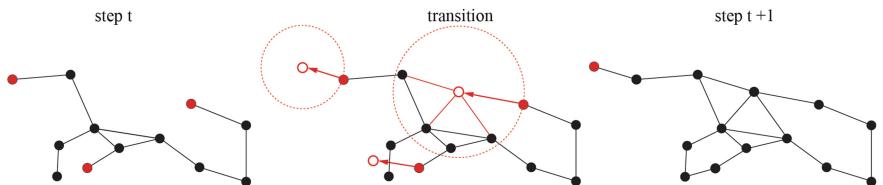


Fig. 2. Interactions between seeds and the graph. In the step t (left), due to the corresponding resultant vector, seeds move and link to the previous position and all nodes in the radius; if a seed crosses an edge, it connects to the nearest node (center). At step $t + 1$, the result of these interactions is a connected graph without crossing edges (right).

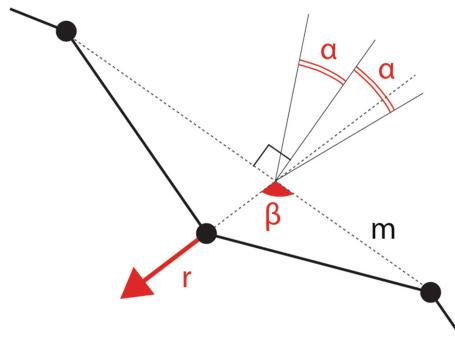


Fig. 3. The seed creation. At each time step, whether the difference between directions of vector r (from all nodes with degree 2) and m (the sum of corresponding nearest vectors) is close to right angle, we add a new seed.

- If the potential edge between u and v crosses another edge, we do not create node v , we connect u to the nearest node, and remove the seed.
- Otherwise we create node v and the edges between u and v .
 - If there are no other nodes in a radius $\|r\|$ around v we keep the seed,
 - otherwise we connect v with each other node in radius $\|r\|$ and remove the seed. (Fig. 2)

Seeds are not only removed, but also created during the simulation. We create a seed at a node u of degree two if the corresponding vector r is almost perpendicular to the line passing between the two neighbours of u (Fig. 3). The parameter α is the maximal allowed deviation from the right angle. The aim of the last mechanism is to consider the evolution during the simulation of the reaction-diffusion layer, in order to handle the generation of new seeds. The vector field evolves during the simulation and the corresponding nearest vector of the vertex in the growing graph change intensity and direction: each vertex in the graph could be a potential source of seeds as consequence of combination of the vector field and the network geometry.

Thanks to this methodology we obtain graphs which grow coherently, embedded in two-dimensional space and where edges do not cross each other. The simulation starts with the creation of three layers and ends when the moving seed set becomes empty or after a predefined number of steps. At each time step, we update the reaction-diffusion layer and compute the corresponding vector field. The graph evolves thanks to the addition of new seeds and the movement of seeds, respecting local rules of interaction. The pseudo-code of simulation is shown in Algorithm 1.

Algorithm 1: Simulation

```

set  $C$  a reaction-diffusion layer
set  $B$  a vector field computed from  $C$  (see eq. 3)
set  $S = \{s_1, \dots, s_n\}$  a moving seed set
set  $G = (V(X), E)$  an initial graph
while  $S \neq \emptyset$  do
    update the reaction-diffusion layer  $C$  (See eq. 2)
    update the vector field  $B$  (See eq. 3)
    foreach vertex  $v \in V(X)$  with degree 2 do      /* Add seeds, see
        fig. 2 */
         $\mathbf{r} \leftarrow$  sum of nearest vectors
         $\mathbf{m} \leftarrow$  vector between the two neighbours of  $v$ 
         $\beta \leftarrow$  angle between  $\mathbf{r}$  and  $\mathbf{m}$ 
        if  $|\beta \% 2\pi| \leq \pi/2 \pm \alpha$  then
             $s \leftarrow$  a new seed at  $v$ 
             $S \leftarrow S + s$ 
    foreach  $s \in S$  do
         $v \leftarrow$  vertex where the seed  $s$  is located
         $X_s \leftarrow$  current position of seed  $s$ 
         $\mathbf{r} \leftarrow$  as the sum of nearest vectors
         $X_f \leftarrow X_s$  moved by  $\mathbf{r}$       /* future position of seed  $s$  */
        if line  $\overline{X_s X_f}$  intersects edge segments then      /* Check edge
            crossing */
            get nearest vertex  $u \in V$  from  $s$ 
             $E \leftarrow E + (u, v)$                       /* Add edge */
             $S \leftarrow S - s$                           /* Remove seed */
        else
             $u \leftarrow$  new vertex at  $X_f$ 
             $V(X) \leftarrow V(X) + u$ 
             $E \leftarrow E + (u, v)$       /* Connect new vertex to previous one
                */
            Move seed  $s$  to  $X_f$ 
             $N(X) \leftarrow$  set of vertices within a distance  $\|\mathbf{r}\|$  from  $X_f$ 
            if  $N(X) \neq \emptyset$  then
                foreach vertex  $v(x_t) \in N(X)$  do /* Connect vertex to
                    nearest vertices */
                    if line  $\overline{X_t X_f}$  does not intersect edge  $e \in E$  then
                         $E \leftarrow E + (v(x_t), v)$ 
                     $S \leftarrow S - s$ 
    
```

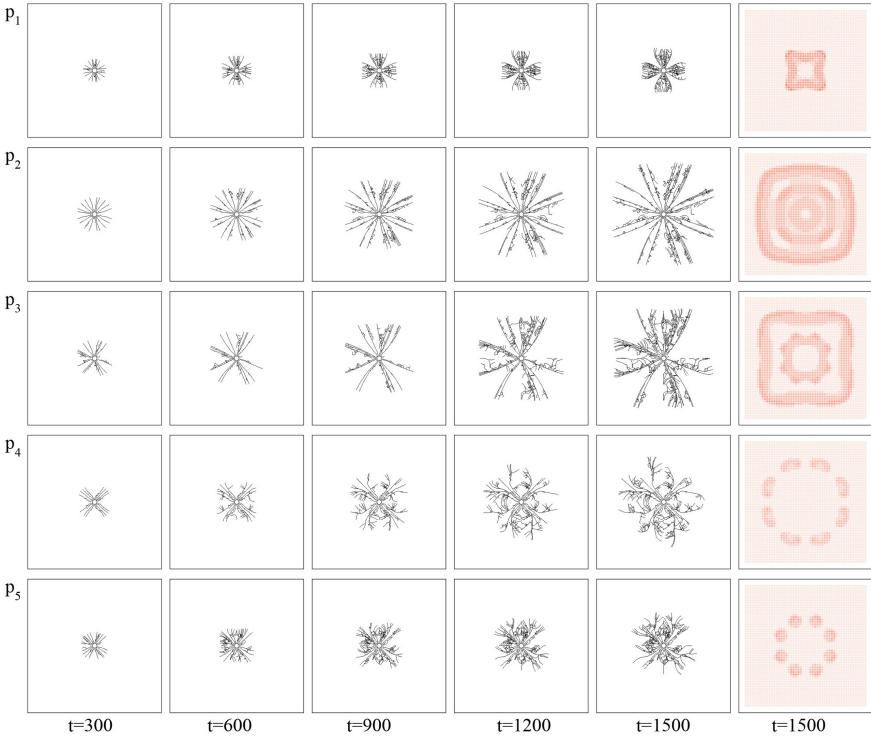


Fig. 4. Simulations of five growing graphs and relative pattern, with $\alpha = 1.0^\circ$. Each row shows the corresponding pattern at final time step $t = 1500$ (last column) and the corresponding graph every 300 time steps.

3 Simulation and Results

The code has been developed in Java and we use the library GraphStream [27] to represent our networks and for some different measures on them. The reaction diffusion layer consists of a lattice of 50x50 unitary cells. The start concentration of morphogens is trivial state $a = 1.0$ and $b = 0.0$, with a small localized perturbing pulse ($b = 1.0$), a given number of moving seeds and associated nodes in the middle of the space. Diffusion parameters are $D_a = 0.1$, $D_b = 0.2$, feed and kill parameter set (f, k) is been chosen in order to obtained some classical patterns: p_1 (0.055, 0.062), p_2 (0.039, 0.058), p_3 (0.029, 0.057), p_4 (0.014, 0.054), p_5 (0.025, 0.060) [2]. At the center of each cell in the reaction diffusion layer we compute the vector \mathbf{b} following the Eq. 3, where $\theta = 2$ and $g = 1$. For each seed, the corresponding vector \mathbf{r} is the sum of the four nearest vectors. For each pattern p , we varied the free parameter $\alpha = [0.1^\circ, 0.2^\circ, \dots, 1.0^\circ]$ and we obtained fifty growing networks¹.

¹ Videos of 3 layers are computed for $\alpha = 1.0^\circ$ until step $t = 2500$, saved every 25th iterations; playback is 3 frames per seconds.

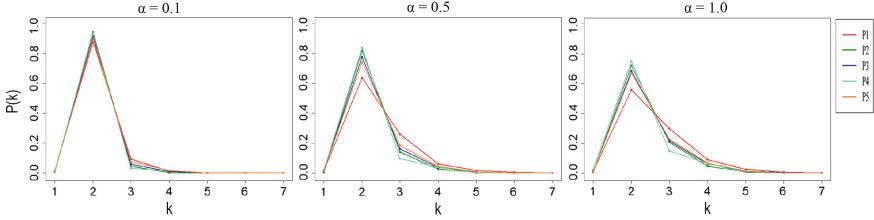


Fig. 5. The degree distribution of five growing graphs calculated for all patterns and for three values of $\alpha = [0.1^\circ, 0.5^\circ, 1.0^\circ]$.

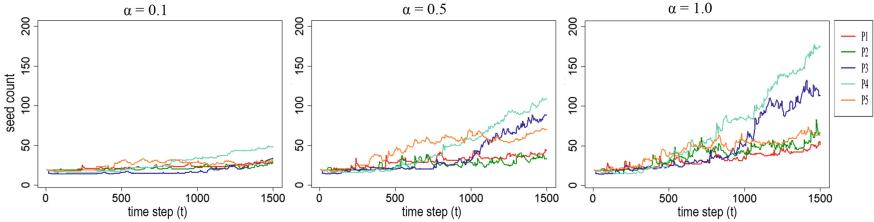


Fig. 6. The evolution of number of moving seeds, calculated for all patterns and for three values of $\alpha = [0.1^\circ, 0.5^\circ, 1.0^\circ]$.

In this section, we study some preliminary properties of all growing graphs generated through the model with few typical topological metrics largely applied in spatial graph analysis [11]. Our interest focus on three aspects: investigate patterns effects on growing networks, evaluate the evolution of graph topology and the impact of the α parameter.

In Fig. 4 we show growing networks every 300 steps, corresponding to reaction-diffusion patterns p with the relative concentration of morphogen b at step $t = 1500$; each growing graph is obtained with $\alpha = 1.0^\circ$. We observe that growing networks is strongly dependent to related reaction-diffusion patterns.

The probability of finding vertices with degree k is $P(k) = v(k)/v$ (where $v(k)$ is the number of vertices with degree k and v is the total number of vertices). In most planar graph applications, the degree of the vertices is comprised between 1 and 7. In our case study, degree distribution shows an hierarchical law behaviour and follows a fast decay from $k = 2$ to $k = 6$, typically observed in street, leaf and ant gallery networks (Fig. 5). In addition, the high probability to generate a new seed, suggested by the frequency $P(2)$, is not actually correlated to real generation of new seeds. During the simulation, only few vectors related

Pattern p_1 : <https://youtu.be/2izGpD2XU0w>
 Pattern p_2 : <https://youtu.be/IwG3oSewSpI>
 Pattern p_3 : <https://youtu.be/ceQVYPadENY>
 Pattern p_4 : <https://youtu.be/LMn6vv9dy7Q>
 Pattern p_5 : <https://youtu.be/vMiAC5rZpzs>.

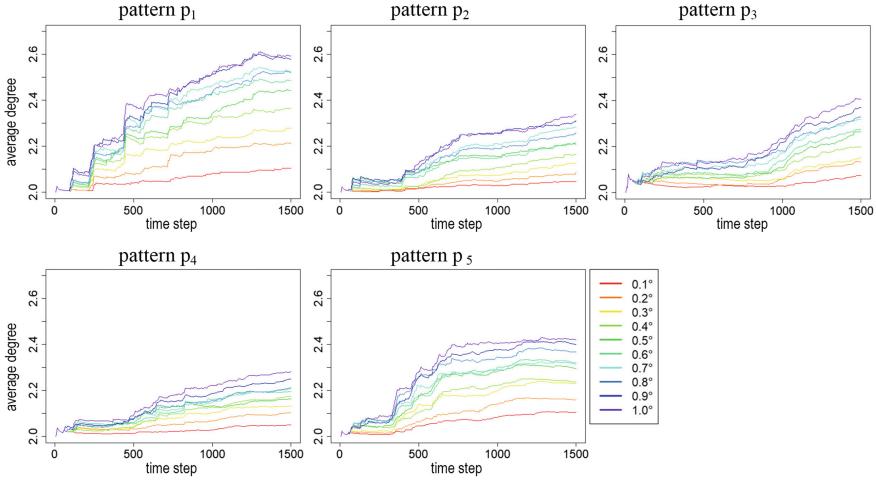


Fig. 7. The evolution of the average degree calculated for all patterns and for all values of $\alpha = [0.1^\circ, 0.2^\circ, \dots, 1.0^\circ]$.

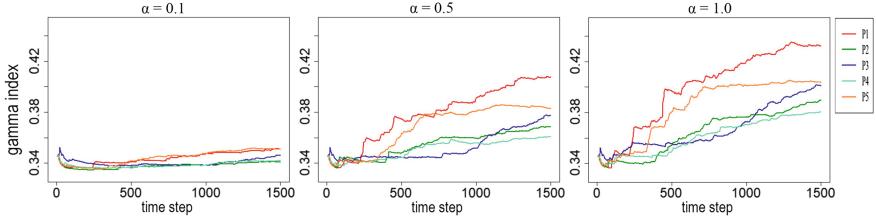


Fig. 8. The evolution of gamma index computed for all patterns and for three values of $\alpha = [0.1^\circ, 0.5^\circ, 1.0^\circ]$.

to vertices with $k = 2$ permits to generate a new seed. Figure 6 depicts the evolution of the number of seeds for three values of $\alpha = [0.1^\circ, 0.5^\circ, 1.0^\circ]$.

In Fig. 7 we show the evolution of average degree $\langle k \rangle = 2e/v$ for all patterns p for all value of α parameter. As expected, parameter α and pattern p play crucial role on the evolution of the average degree $\langle k \rangle$. We observe a similar behaviour for all patterns. The α parameter impacts the connectivity at global scale and a sudden variation is detected at the same time step for each graph. Results suggest that the evolution of morphogens concentration governs the growth of the network and the parameter α amplifies graph average degree.

In order to characterize the evolution of graph density, we calculate the gamma index. It is defined as $\gamma = e/e_{MAX}$ and compares the actual number of edges e to theoretical maximum number of edges $e_{MAX} = 3(v - 2)$ (in the case of undirected planar graph) [32]. In Fig. 8 we observe ample fluctuations during the simulation. Like for average degree $\langle k \rangle$ evolution, we observe non monotone curves for all patterns, as a consequence of an unstable process. Accordingly to a

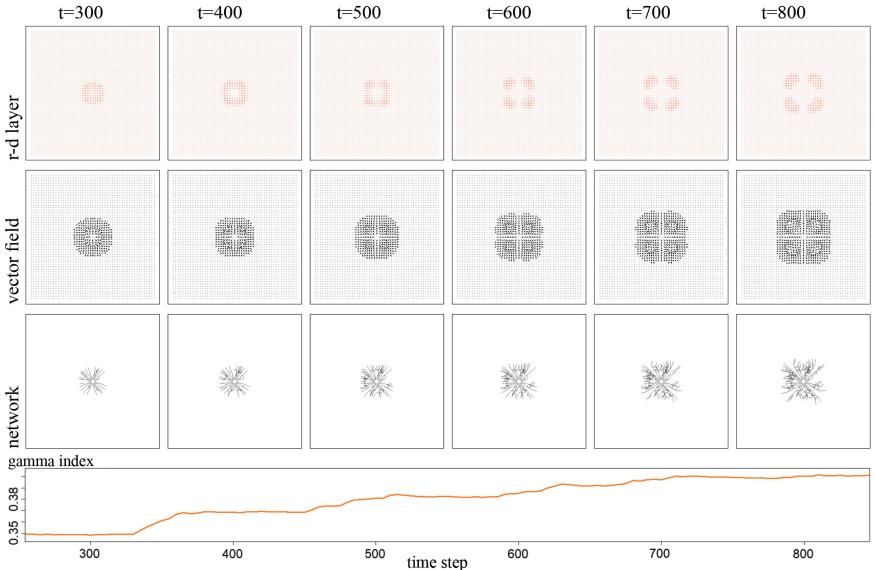


Fig. 9. Snapshots (from step $t = 300$ to $t = 800$, computed each 100 steps) of the three layers (the first three rows) for the pattern p_5 and the corresponding evolution of γ index (the fourth row).

sudden change of morphogen concentration b , graphs grows discontinuously. For instance, observing spatial evolution of concentration b in pattern p_5 , a mitosis process increase number of unstable spots (Fig. 9). In the reaction diffusion layer, this process causes the interaction showed in Fig. 3 and increases the number of seeds. The network suddenly grows and the γ index shows the rise of the graph density. This clearly demonstrates that growing network mechanisms proposed in this paper consider at all times the evolution of morphogens concentration and the spatial properties of the network.

4 Conclusion

In this paper we have exposed a methodology able to generate graphs embedded in two-dimensional space. A reaction-diffusion system, applied to a regular grid, governs a dynamic vector field, which impacts a set of moving seeds. Growing graphs respect the planarity condition, connecting the new vertices to elements in an evolving neighbourhood. The parameter α affects network growth, through local interactions between graph elements and the vector field.

Thanks to simple local connection rules, which govern spatial interactions between moving seeds and the growing network, we have developed a parsimonious model, where spatial and temporal coherence is ensured during the simulation and generate only one connected graph. Combined with the planarity condition our model can be used in a wide range of applications. The evolution

of technical spatial networks such as street networks should be an interesting application for a framework governed by elementary local interactions.

Although these preliminary and not exhaustive evidences, several questions remain unanswered and required an in-depth investigation. Our model allows to define different levels of abstraction. For instance, in urban studies, it may represent sprawl of a single city at microscopic scale or the densification of connections between cities at macroscopic scale. Our minimal approach integrates few essential physical processes and permits to investigate elementary mechanisms of spatial graph generation.

These early results should be integrated in the next works by a systematic evaluation of spatial effects of graph growth to evolving concentration of morphogens, in order to take into account the feedback of the graph to the reaction-diffusion layer. In a complex system, like a growing city, the network plays a crucial role in the spatial evolution of forces acting on the development of the city.

Acknowledgements. This work is supported by the project “AMED” co-funded by ERDF and the region Normandy.

References

1. Achibet, M., Balev, S., Dutot, A., Olivier, D.: A model of road network and buildings extension co-evolution. *Procedia Comput. Sci.* **32**, 828–833 (2014)
2. Adamatzky, A.: Generative complexity of Gray-Scott model. *Commun. Nonlinear Sci. Numer. Simul.* **56**, 457–466 (2018)
3. Barthelemy, M.: Spatial networks. *Phys. Rep.* **499**(1), 1–101 (2011)
4. Barthelemy, M.: Morphogenesis of spatial networks. In: *Lecture Notes in Morphogenesis*. Springer International Publishing (2018)
5. Barthelemy, M., Flammini, A.: Modeling urban street patterns. *Phys. Rev. Lett.* **100**(13), 138,702 (2008)
6. Barthélémy, M., Flammini, A.: Optimal traffic networks. *J. Stat. Mech. Theory Exp.* **2006**(07), L07,002 (2006)
7. Batty, M.: *Cities and Complexity: Understanding Cities With Cellular Automata, Agent-Based Models, and Fractals*. MIT Press, Cambridge, MA (2007)
8. Bird, R.B.: Theory of diffusion. In: T.B. Drew, J.W. Hoopes (eds.) *Advances in Chemical Engineering*, vol. 1, pp. 155–239. Academic Press (1956)
9. Boccaletti, S., Latora, V., Moreno, Y., Chavez, M., Hwang, D.U.: Complex networks: structure and dynamics. *Phys. Rep.* **424**(4), 175–308 (2006)
10. Brede, M.: Coordinated and uncoordinated optimization of networks. *Phys. Rev. E* **81**(6), 066,104 (2010)
11. Buhl, J., et al.: Topological patterns in street networks of self-organized urban settlements. *Eur. Phys. J. B Condens. Matter Complex Syst.* **49**(4), 513–522 (2006)
12. Courtat, T., Gloaguen, C., Douady, S.: Mathematics and morphogenesis of cities: a geometrical approach. *Phys. Rev. E* **83**(3), 036,106 (2011)
13. Diestel, R.: *Graph Theory*, 4 edizione edn. Springer, Heidelberg; New York (2010)
14. Gastner, M.T., Newman, M.E.J.: The spatial structure of networks. *Eur. Phys. J. B Condens. Matter Complex Syst.* **49**(2), 247–252 (2006)

15. Gray, P., Scott, S.K.: Autocatalytic reactions in the isothermal, continuous stirred tank reactor: isolas and other forms of multistability. *Chem. Eng. Sci.* **38**(1), 29–43 (1983)
16. Guillier, S., Muñoz, V., Rogan, J., Zarama, R., Valdivia, J.A.: Optimization of spatial complex networks. *Phys. A Stat. Mech. Its Appl.* **467**, 465–473 (2017)
17. Holme, P.: Modern temporal network theory: a colloquium. *Eur. Phys. J. B* **88**(9), 1–30 (2015)
18. Jiang, B., Claramunt, C.: Topological analysis of urban street networks. *Environ. Plan. B Plan. Des.* **31**(1), 151–162 (2004)
19. Kari, J.: Theory of cellular automata: a survey. *Theor. Comput. Sci.* **334**(1), 3–33 (2005)
20. Katifori, E., Magnasco, M.O.: Quantifying loopy network architectures. *PLOS ONE* **7**(6), e37,994 (2012)
21. Kondo, S., Miura, T.: Reaction-diffusion model as a framework for understanding biological pattern formation. *Science (New York, N.Y.)* **329**(5999), 1616–1620 (2010)
22. Latora, V., Nicosia, V., Russo, G.: Complex Networks: Principles, Methods and Applications. Cambridge University Press, Cambridge, United Kingdom; New York, NY (2017)
23. Lion, B., Barthelemy, M.: Central loops in random planar graphs. *Phys. Rev. E* **95**(4), 042,310 (2017)
24. Masucci, A.P., Smith, D., Crooks, A., Batty, M.: Random planar graphs and the London street network. *Eur. Phys. J. B* **71**(2), 259–271 (2009)
25. Nicolaides, C., Juanes, R., Cueto-Felgueroso, L.: Self-organization of network dynamics into local quantized states. *Sci. Rep.* **6**, 21,360 (2016)
26. Perna, A., Kuntz, P., Douady, S.: Characterization of spatial network like patterns from junction geometry. *Phys. Rev. E* **83**(6), 066,106 (2011)
27. Pigne, Y., Dutot, A., Guinand, F., Olivier, D.: GraphStream: a tool for bridging the gap between complex systems and dynamic graphs. Emergent properties in natural and artificial complex systems. In: Satellite Conference within the 4th European Conference on Complex Systems (2008)
28. Porta, S., Crucitti, P., Latora, V.: The network analysis of urban streets: a primal approach. *Environ. Plan. B Plan. Des.* **33**(5), 705–725 (2006)
29. Rui, Y., Ban, Y., Wang, J., Haas, J.: Exploring the patterns and evolution of self-organized urban street networks through modeling. *Eur. Phys. J. B* **86**(3), 74 (2013)
30. Turing, A.M.: The chemical basis of morphogenesis. *Philos. Trans. R. Soc. London. Ser. B Biol. Sci.* **237**(641), 37–72 (1952)
31. Viana, M.P., Strano, E., Bordin, P., Barthelemy, M.: The simplicity of planar networks. *Sci. Rep.* **3**, 3495 (2013)
32. Xie, Y.B., Zhou, T., Bai, W.J., Chen, G., Xiao, W.K., Wang, B.H.: Geographical networks evolving with an optimal policy. *Phys. Rev. E Stat. Nonlinear Soft Matter Phys.* **75**(3 Pt 2), 036,106 (2007)



Multilayer Network Model of Movie Script

Youssef Mourchid^{1(✉)}, Benjamin Renoust², Hocine Cherifi³, and Mohammed El Hassouni^{1,4}

¹ Faculty of Sciences, LRIT-CNRST URAC 29, Rabat IT Center, Mohammed V University, 1014 Rabat, Morocco

{youssefmour,mohamed.elhassouni}@gmail.com

² Institute for Datability Science, Osaka University, Osaka, Japan

renoust@ids.osaka-u.ac.jp

³ LE2I UMR 6306 CNRS, University of Burgundy, Dijon, France

hocine.cherifi@u-bourgogne.fr

⁴ DESTEC, FLSH, Mohammed V University in Rabat, Rabat, Morocco

Abstract. Network models have been increasingly used in the past years to support summarization and analysis of narratives, such as famous TV series, books and news. Inspired by social network analysis, most of these models focus on the characters at play. The network model well captures all characters interactions, giving a broad picture of the narration’s content. A few works went beyond by introducing additional semantic elements, always captured in a single layer network. In contrast, we introduce in this work a multilayer network model to capture more elements of the narration of a movie from its script: people, locations, and other semantic elements. This model enables new measures and insights on movies. We demonstrate this model on two very popular movies.

Keywords: Script · Multilayer networks · Narration · Movie

1 Introduction

Whether narrated through books or movies, stories create their own universe, putting at play characters in a rich world. The intertwining of the different elements of a story forms the imaginary world that may catch the readers or viewers’ full attention. Books leave to their readers the assembly of story elements glued with their own imagination for each to construct their own vision of the world depicted. Movies are very different in that, they serve viewers a fully constructed world for them only to enjoy with wonder. When some movie directors like to play with the viewers’ progressive perception of their created universe, others really present rich imaginary worlds like in science-fiction movies. One could wish to understand if the interactions of the story elements can make a fingerprint of a story, characterizing a genre or a director.

The interactions of stories elements have often been captured through the use of network modeling [1, 5, 7, 12]. It has been used to support narration of a wide range of stories, from books [7], TV series [1], stories of news events [5], and movies [12], which make our focus. Networks are visual objects that can not only be cleverly visualized to explicit the stories [5], but also their structure can be interrogated in matter of *topology* [7, 20].

These works are limited. Indeed, they mostly focus on a single facet of the stories, mainly the characters at play. Journalists often investigate a story by articulating the 5 W-questions: *Who?*, *Where?*, *What?*, *When?* and *How/Why?* [13, 22]. The network analysis of stories attempts to answer *How/Why?* by articulating the other 4 elements in a network. Since *When?* is provided with data (tight with the narration), the previous works have mostly focused on *Who?* in a single-layer network. We wish to introduce a more holistic approach on tackling also *Where?* and *What?* with a multi-layer network modeling of the stories.

Of course, the network may be manually created [3], but automated approaches may even scale to larger archives [5, 7]. Since it is still hard to get all the necessary information from the video data itself [21, 23], we may rely on text analysis. Fortunately, in their very early stages of creation, movies are *written* in form of scripts. A script is usually extremely well structured, and contains all the necessary components to automatically analyze a movie [8] (*e.g.* scenes, dialogues, characters, *etc.*).

We propose in this paper to exploit this textual information to automatically extract the movie networks from the scripts. We contribute with a novel multilayer model of a movie script enabling to articulate characters, places, and themas. The multilayer network captures a richer structure of a movie. It completes the single character network analysis, and brings new topological analysis tools [2].

After discussing the related work in the next section, we introduce the proposed model in Sect. 3. We describe the movie script processing in Sect. 4, before deploying the analysis in 5 on two popular science-fiction movies. We finally conclude in Sect. 6.

2 Related Works

Networks have been explored to analyze videos data [6] and shown efficient for topics and concept analysis [4]. Particularly actor networks have been broadly analyzed from literature [7], from TV news videos [5], and even a website is dedicated to the social analysis of Game of Thrones [3].

Many techniques have been proposed to analyze movies using a graph structure. Based on story structure analysis, they support the inference of the meaning of interactions between movie objects or scenes through structural analysis. These approaches consist of two major steps: The first one is the construction of the story model, and the second one is the extraction of the information from the model. Once the network that connects entities such as characters, scenes or shots is constructed, semantic information is extracted from this network.

Yeung et al. [9] proposed a scene transition graph and an analysis method for movie browsing and navigation. Each node in the scene transition graph denotes a cluster of shots. There is a link between two nodes i and j if a shot represented by node i immediately precedes any shots represented by node j . This approach builds a network with interactions between shots and then analyzes it in order to extract the story units of scenes. This model uses only a hierarchical clustering of shots with visual primitives for browsing, but it cannot retrieve movie story elements such as actors, scenes, and dialogues.

Jung et al. [10] proposed a narrative structure graph to summarize a movie. The graph is composed of scene nodes which are narrative elements with character interactions, and connections between scenes decided by editorial relations. Using only scenes to construct a narrative structure graph for movie summarization is not sufficient. Indeed, story elements such as major characters and their interactions cannot be retrieved from this graph.

Tan et al. [1] proposed an analysis of the character networks in two science fiction television series. These networks are constructed based on the scene co-occurrence between characters to indicate the presence of a connection. Global network topological measures such as the average path length, graph density, network diameter, average degree, are computed and found to be similar between the two series. Furthermore, various node centrality scores are computed and used to reflect on the interplay between the central characters and the overall narrative.

Some studies have been conducted to apply social network analysis (SNA) for movie story analysis. RoleNet [11] is an SNA-based approach that was proposed to analyze movie stories. It can identify automatically leading roles and corresponding communities by investigating the social interactions between characters using a weighted graph where nodes represent characters, and edges represent co-appearance relationships, *i.e.* two characters appear in the scene. Edge weight represents the number of co-appearances of two characters in the same scene. However, using only scenes to model the movie story is not enough: some scenes may be very long, and others are short. Using an additional source such as the dialog would be more adaptable than this assumption.

Character-net [12] is another interesting work that proposes a story-based movie analysis method via social network analysis. While RoleNet uses co-appearance as relationship between characters, Character-net utilizes dialog. Edges are weighted by the quantity of dialogue exchanged by the characters. Once the weighted network is built, characters are classified according to their degree centrality value as major, minor or extra role. Finally, the classification result is used to detect the movie sequences through clique clustering, and major and minor role clustering.

All of these works focus mostly on a single facet of the stories, either the characters or the scenes. We propose to extract more elements of the narration of a movie such as characters, locations and themes from its scripts. Through a multilayer network, all the extracted elements with their interactions are grouped together in order to form the story of the movie. By articulating all these elements together, the network enables us to respond to the main questions

about the story: Who (W1), Where (W2) and What (W3). It allows also to investigate the relative importance of the various types of nodes.

3 Modeling Stories with a Multilayer Network

In investigative disciplines, the *Four Ws* (*Who?*, *Where?*, *What?*, *When?*) [13] are the fundamental questions that constitute the formula describing a complete story [14]. Inferring *How/Why?* can be done while articulating the other *Ws* making them essential bricks of analysis:

Given our context of movie understanding, we reformulate the four *Ws* as follows:

- *Who?* refers to the **characters** of a movie;
 - *Where?* refers to the **locations** the action of a movie takes place;
 - *What?* refers to the **subjects** the movie talks about.
 - *When?* refers to the **time** guiding the succession of actions taught by the movie.

Except for *time*, each of answers to the *Ws – characters, locations, and subjects* – form the entities grounding our study.

Our goal is to help formulate movie understanding by articulating these four Ws. A multilayer graph model is used to put these elements together in relationship to form a story. This graph is made of three node layers in order to represent each type of entity (characters, locations, and subjects – represented by *keywords*) with multiple relationships between them.

We model two main classes of relationships: intra-layer relationships between nodes of a same category (*e.g.* two people conversing); and inter-layer relationships between nodes of different categories (*e.g.* a person being at a specific place); Put together, the multiple families of nodes and edges form a multilayer graph as illustrated in Fig. 1.

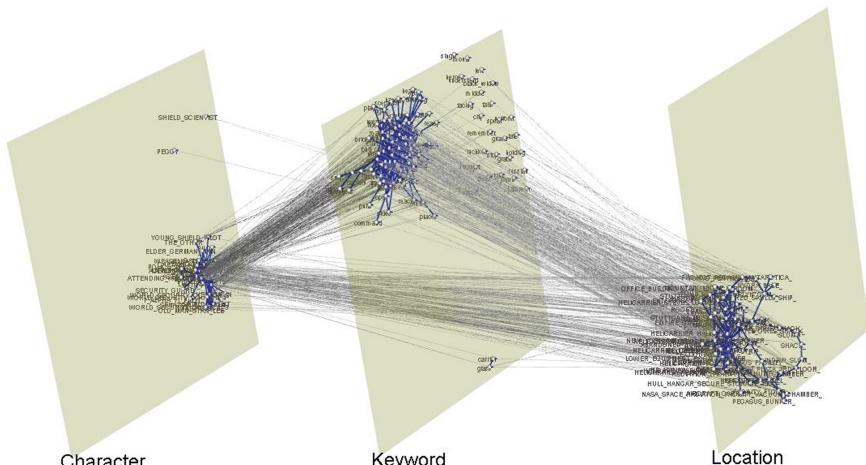


Fig. 1. Multilayer network extracted from Avengers script with the three layers of *Character*, *Location*, *Keyword* and their interactions (realized with MuxViz [24])

We now define our multilayer graph $\mathbb{G} = (\mathbb{V}, \mathbb{E})$ such that:

- $V_C \subseteq \mathbb{V}$ represents the set of characters $c \in V_C$,
- $V_L \subseteq \mathbb{V}$ represents the set of locations $l \in V_L$,
- $V_K \subseteq \mathbb{V}$ represents the set of keywords $k \in V_K$.

The different families of relationships can then be defined as:

- $e \in E_{CC} \subseteq \mathbb{E}$ between two characters such that $e = (c_i, c_j) \in V_C^2$, when a character $c_i \in V_C$ is conversing with another character $c_j \in V_C$.
- $e \in E_{LL} \subseteq \mathbb{E}$ between two locations such that $e = (l_i, l_j) \in V_L^2$, when there is a temporal transition from one location $l_i \in V_L$ to the other $l_j \in V_L$.
- $e \in E_{KK} \subseteq \mathbb{E}$ between two keywords such that $e = (k_i, k_j) \in V_K^2$, when $k_i \in V_K$ and $k_j \in V_K$ belong to the same subject.
- $e \in E_{CK} \subseteq \mathbb{E}$ between a character and keyword such that $e = (c_i, k_j) \in V_C \times V_K$, when the keyword $k_j \in V_K$ is pronounced by the character $c_i \in V_C$.
- $e \in E_{CL} \subseteq \mathbb{E}$ between a character and a location such that $e = (c_i, l_j) \in V_C \times V_L$, when a character $c_i \in V_C$ is present in location $l_j \in V_L$.
- $e \in E_{KL} \subseteq \mathbb{E}$ between a keyword and a location such that $e = (k_i, l_j) \in V_K \times V_L$, when a keyword $k_i \in V_K$ is mentioned in a conversation taking place in the location $l_j \in V_L$.

Note that for the sake of simplicity, we do not consider edge direction and weight. Furthermore, as we do not intend to study the network dynamics, time is not taken into account. However, time supports everything: the existence of a node or an edge is defined upon time, unrolled by the order of movie scenes.

As a shortcut, we can now refer to subgraphs by only considering one layer of links and its induced subgraph:

- $G_{CC} = (V_C, E_{CC}) \subseteq \mathbb{G}$ refers to the subgraph of character interaction;
- $G_{LL} = (V_L, E_{LL}) \subseteq \mathbb{G}$ refers to the subgraph of location transitions;
- $G_{KK} = (V_K, E_{KK}) \subseteq \mathbb{G}$ refers to the subgraph of keyword co-occurrence;
- $G_{CK} = (V_C \cup V_K, E_{CK}) \subseteq \mathbb{G}$ refers to the subgraph of characters speaking keywords;
- $G_{CL} = (V_C \cup V_L, E_{CL}) \subseteq \mathbb{G}$ refers to the subgraph of characters standing at locations;
- $G_{KL} = (V_K \cup V_L, E_{KL}) \subseteq \mathbb{G}$ refers to the subgraph of keywords mentioned at locations.

Now that we have set the model, we need to extract its elements from the script. This allows analyzing various topological properties of the network in order to gain a better understanding of the story.

4 Extracting the Multilayer Network from the Script

We now describe the methodology used to build the network of the movie. Fortunately, most feature-length movies are produced using scripts that happen to be very well-structured textual documents [8]. They organize the film in scenes, each placing characters in locations, describing their context, the actions, and most importantly the characters dialogues. As the script brings at play numerous but ambiguous notions, we first give essential information about its structure and a few important definitions. Then, we present the methodology used to extract the various entities and interactions. Finally, we explain how to build the network based on this information.

4.1 Script Structure and Definitions

A script details all elements of a story: location and setting, characters with their situation and actions, and most importantly dialogues. The actual content of a script often follows a semi-regular format [8] such as depicted in Fig. 2.

A script usually starts with a heading describing the location and time of the scene. Specific keywords give important setting information (such as inside or outside scene) and character and key objects are often emphasized. The script then follows in a series of dialogues and setting descriptions. In order to remove any ambiguity, we first define the following dedicated glossary.

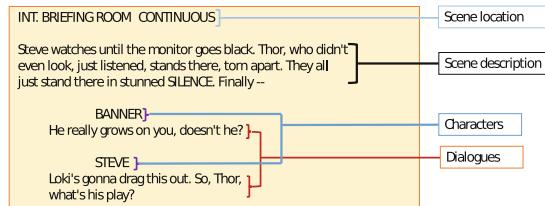


Fig. 2. Snippets of a resource script describing the movie *The Avengers 2012* [26], displaying different elements manipulated (characters, dialogues and locations).

- Script: A text source of the movie which has descriptions about scenes, with setting and dialogues.
- Scene: Chunk of a script, temporal unit of the movie. The collection of all scenes form the movie script.
- Setting: The location a scene takes place, and its description.
- Character: Denotes a person/animal/creature who is present in a scene, often impersonated by an actor.
- Dialogues: A collection of utterances, what all characters say during a scene.
- Utterance: An uninterrupted block of a dialogue pronounced by one character.
- Conversation: A continuous series of utterances between two characters.

- Speaker: A character who pronounced an utterance.
- Description: A script block which describes the setting.
- Location: Where does a scene take place, or mentioned by a character.
- Keyword: Most relevant information from an utterance, often representative of its topic.

4.2 Script Preprocessing

Figure 3 illustrates the script processing pipeline. Although this process is language-dependent, we restrict our attention to scripts written in English. Whatever, generally, script content follows a semi-regular format as shown in Fig. 2.

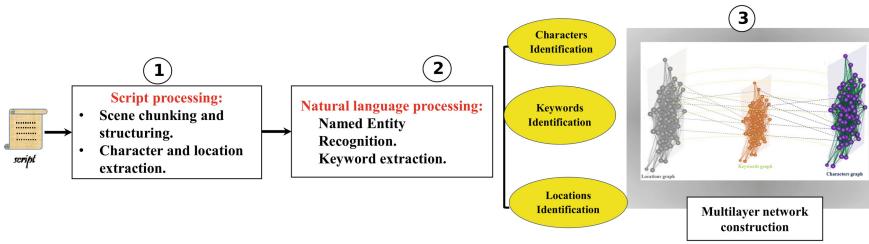


Fig. 3. Schematic view of the model construction process.

Scene chunking: The first task is to chunk the script into scenes. Indeed, they are the main subdivisions of a movie, and consequently the main unit of analysis. To each scene is attached a set (actions, location and characters). Each scene starts with a technical description line written in capital letters, that establishes the physical context of the action that follows. It starts by indicating whether a scene takes place inside or outside (*INT* or *EXT*), the name of the location, and can potentially specify the time of day (e.g. *DAY* or *NIGHT*). These are markers we detect to chunk the script into scenes.

Scene structure: Sets are attached to locations which are always included in the scene header, that we can easily parse. Important people and key objects are usually highlighted in capital letters that we harvest while analyzing the text. The rest of a scene is made of dialogue and description. Character names and their actions are always depicted before the actual dialogue lines. A line indent also helps to identify characters and dialogue parts in contrast to scene description.

By structuring each scene into its list of description and dialogues, we can harvest set locations and utterance speakers. We can then identify scene conversations and characters present at a set. Specific descriptions can then be associated to locations, and dialogues to characters.

4.3 Text Processing

The next step is to process the actual text content that is already associated either to a set location, or a to speaker. To do so, we use natural language processing tools to extract named entities and keywords.

Named Entity Recognition: Named Entity Recognition (NER) tag important words identified in a text content (such as people, organizations, cities, *etc.*). We use the spaCy library [15] for its efficiency. We apply NER to scene description blocks and discard irrelevant categories such as quantities, ordinals, money *etc.*. Because many words may end up mislabelled (especially due to the ambiguous context of a sci-fi movie), we manually curate the resulting list of words. We combine the categories to three larger categories: characters, location, and keywords, and we assign a unique class for ambiguous names referring to the same concept (*i.e.* $\{TONY, STARK, IRONMAN\} \rightarrow TONYSTARK$). NER mainly helps us identifying characters present at a scene who are pronouncing utterances in conversations.

Keyword extraction: Keywords are identified from dialogues. We measure the relevance of keywords using three methods: TF-IDF [17], LDA [16] and Word2Vec [18]. Script texts are made of short sentences (even shorter after stop-words removal), so Word2Vec and TF-IDF render either too few words or too much of words without semantic content. Therefore, we mainly rely on LDA, which bring the best trade-off, and manually curate the resulting words by removing the semantic-less keywords (such as *can*, *have*, and so on).

4.4 Network Construction

As a result of the previous two steps, for each scene, location and their description, characters, utterances and their speakers, and keywords extracted from description and utterances are identified. We now use this structure to construct the multilayer graph.

Let us revisit our investigative questions in the context of a scene: *Who is involved in a scene?* may be tackled by characters, *Where does a scene take place?* is identified through locations, and *What is a scene about?* is identified through keywords. These three families of entities naturally form the three categories of nodes, respectively V_C , V_L and V_K .

We now wish to infer the relationships we described in Sect. 3. Two characters c_i , c_j can be connected when they participate in a same conversation, hence forming an edge $e_{c_i,c_j} \in E_{CC}$. We connect two locations $e_{l_i,l_j} \in E_{LL}$ when there is a temporal transition between the locations l_i and l_j (analogous to geographical proximity), *i.e.* following the succession of two scenes. Keywords k_i , k_j co-occurring in a same conversation create an edge $e_{k_i,k_j} \in E_{KK}$.

Using the structure, extracted from the script, we can add additional links between categories. An edge $e_{c_i,l_j} \in E_{CL}$ associates a character c_i with a location l_j when the character c_i appears in a scene taking place at location l_j . When a character c_i speaks an utterance in a conversation, for each keyword k_j that is detected in this utterance, we will create an edge $e_{c_i,k_j} \in E_{CK}$. Finally, we can

associate the keywords k_i extracted in conversation placed in a location l_j to form the edge $e_{k_i, l_j} \in E_{KL}$. A resulting graph combining all layers is visualized in Fig. 1.

5 Analyzing the Multilayer Network

5.1 Data and Method

We now wish to apply our methodology onto two popular science-fiction movie scripts, *Star Wars: Episode IV-A New Hope* [25] (hereafter *SW*) and *Marvel’s the Avengers* [26] (hereafter *AV*). *Star Wars*, 1977, is iconic of the science-fiction genre. It tells the story of a young man awaking to the Force (a religion as much as a source of power) in an epic battle between the Light (the Jedi) and Dark (the Empire) side of the Force dominating a galaxy. *The Avengers*, 2012, puts at play a team of superheroes (the Avengers) from the Marvel universe against an extraterrestrial villain trying to invade Earth in search for the Tesseract, a powerful artifact.

the scripts are collected from (www.imsdb.com), an online repository database made available by filmmakers. Once the graphs are extracted for each movie, we first investigate their global topological properties. Then, we investigate the node *influence* using classical centrality measures (Degree, Betweenness, Eigenvector) and a measure recently introduced that combine both local and global features (Influence Score) [19].

5.2 Basic Topological Properties

Basic topological properties of the networks are reported in Table 1. Note that the character and the location layers are made of a single connected component. The keyword layers have a few isolated nodes. We report the statistics of the “giant” component in this case. The multilayer networks of both movies are quite small. The Character layers are very dense as compared to location and keyword. Indeed, almost all characters have been talking together. In addition, all layers have a high diameter, especially for the location layer. This is due to a few temporal transitions between not very frequent locations that introduce long paths. In both movies, the clustering coefficient is very high for the character layer. The keywords layers are also well-clustered. There is no triangle in the inter-layer interactions (as in G_{CL} , G_{CK} and G_{KL}) because those are bipartite graphs linking two sets of objects. We can observe that the multilayer networks are assortative. Indeed, major objects in the movie, especially characters and keywords tend to connect together. For example, tesseract and Hellicarrier bridge which are major keywords tend to be connected with Nick Fury a main character in the movie. Both multilayer networks exhibit a small average shortest path.

Table 1. Global topological properties of the various networks: number of nodes (V), number of edges (E), density (ρ), diameter (d), average clustering coefficient (C), assortativity coefficient (τ), and average shortest path (l_G).

	Avengers					Star									
	V	E	ρ	d	C	τ	l_G	V	E	ρ	d	C	τ	l_G	
G	187	1391	0,06	5	0,53	0,5	2,42	269	2586	0,06	6	0,56	0,52	2,61	
G_{CC}	38	276	0,46	4	0,78	0,18	2,08	62	650	0,32	4	0,84	0,07	2,02	
G_{KK}	81	221	0,07	7	0,49	0,05	2,83	74	701	0,08	7	0,57	$2 * 10^{-2}$	3,46	
G_{LL}	68	240	0,07	9	0,29	0,03	3,90	133	566	0,05	10	0,24	0,17	3,81	
G_{CL}	81	159	0,04	6	0	-0,01	3,36	143	316	0,04	7	0	0,07	3,24	
G_{CK}	96	228	0,07	6	0	-0,02	2,78	85	148	0,06	5	0	-0,1	2,91	
G_{KL}	115	267	0,04	6	0	-0,02	3,14	96	205	0,06	10	0	-0,02	3,27	

5.3 Node Influence

We report in Table 2 the top 5 nodes sorted by their centrality score computed in the three layers considered independently for *SW* and *AV*.

Ranking characters: All top characters are the main ones in *SW* with *Luke Skywalker* dominating all centralities. *Vader*, *Leia* and *Han Solo* are accompanying the main character during all the movie. *Leia* is the princess that is saved leading the rebels. *Ben* the captain of the Millennium Falcon, work together with *C-3PO* to rescue the beautiful princess. *Darth Vader* is the main villain. In *AV*, the top characters play an important role in the movie and the Marvel universe. Note that except for three characters they are superheroes (, also having their own movie (including the top-billed and occurring one, *Tony Stark*). *Nick Fury*, the top influential character is basically the liaison between all superheroes, which team-up with *Tony Stark*, *Captain America* and *Natasha* against *Loki* the last of our rank, which is the main villain.

Ranking keywords: In *SW*, there is an extremely uneven distribution in the keywords centralities. In the top rankings we still find key concepts such as the *Empire*, *Jedi*, *side*. The name of *Luke* the main character is notably the most occurring by far. In *AV*, the top keyword overall is the *Tesseract*, which is the object motivating the villains to attack. The following keywords explain why so: *world* and *control* supports that this object is a powerful source which can control the world. *Phil Coulson* is the one send by *Nick fury* to find the heroes he needs to oppose *Loki* with the power of *Tesseract*.

Ranking locations: In *SW*, the top location is the *Space*, where almost all the scenes take place. *Surface of the Death Star* – in which another major part of the action takes place – destroys it. The *Lukes X-wing Fighter Cockpit* is Luke's ship which helps him in the war against *Darth Vaders Cockpit*. In *AV*, the top three locations are the places where all main characters regroup or fight, and in which most action occurs (note that the *Helicarrier Bridge* and *Helicarrier Detention Section* are two places of the *Quinjet*, which is the vehicle transporting the main characters everywhere). *Stark Tower* is *Tony Stark*'s home which is another key place in the movie. *Sky* is a transitory scene. *Manhattan* is the city where the fight begins between superheroes and *Loki*. We can observe that for both movie, the Influence score ranking is globally well correlated with the fraction of appearance of the objects per scene (Characters, Keywords, Locations).

Multilayer ranking: Table 3 reports the centrality scores computed in the multilayer networks. The top nodes are mainly the main characters of the movie. However, major Locations and keywords are also in the top ranking list. We can observe that while degree centrality tends to bring forward characters, betweenness and eigencentrality measures tend to bring forward locations.

To summarize, these results show that even a very succinct topological analysis of the multilayer networks can give a pretty good idea about the movie content.

Table 2. First 5 nodes sorted by relevance according to their centrality score: Degree (D), Betweenness(B), Eigencentrality (Ei), Influence score (I.S). (0) is the fraction of scenes where the objects appears. In the location layer abbreviations mean S: Space; A: Alderaan; Dsh: Death Star Hallway; Td: Tatooine Desert; Dvc: Darth Vader's Cockpit; Sds: Surface of The Death Star; Lxfc: Luke's X-wing Fighter Cockpit; Sds:Space Around The Death Star; Mfc: Millennium Falcon Cockpit; M: Manhattan; St: Sark Tower; Sk: Sky; Hds: Helicarrier Detention Section.

CHARACTERS					KEYWORDS					LOCATIONS					
	D	B	Ei	I.S	O	D	B	Ei	I.S	O	D	B	Ei	I.S	O
SW	Luke	Luke	Luke	89	Empire	Side	Luke	Empire	7	S	S	Sds	S	31	
	C-3PO	Han	C-3PO	41	Luke	Empire	Side	Luke	180	Sds	Mfc	Lxfc	Sds	29	
	Han	Vader	C-3PO	42	Jedi	Sandpeople	Jedi	Side	61	Lxfc	A	Dyc	Lxfc	36	
	Vader	Han	Leia	33	Side	Luke	System	Long	38	Sads	Dsh	Sads	Sads	26	
	Leia	Red leader	Ben	24	System	Stay	Back	Jedi	11	Mfc	Td	Rlc	Dvc	21	
AV	N.Fury	N.Fury	C.America	36	Tesseract	Tesseract	Tesseract	Tesseract	31	Hb	Hb	Hb	Hb	19	
	Tony	Banner	Tony	48	P.Coulson	P.Coulson	P.Coulson	P.Coulson	17	M	Hds	M	Hds	10	
	C.America	C.America	N.Fury	69	World	Tony	World	World	17	St	St	St	St	11	
	Natasha	Loki	Natasha	36	Tony	World	Tony	Tony	2	Sk	Sk	Sk	Sk	9	
	Banner	Natasha	Banner	35	Control	Control	Control	Control	17	Hds	M	Hds	M	13	

Table 3. First 12 nodes sorted by centrality scores for the multilayer networks. In the location layer abbreviation mean, Bg: Brooklyn Gym; Bl: Banner's Lab; Bs: Bridge Street; Mfc: Millenium Falcon Cockpit; Td: Tatooine Desert; Tmes: Tatooine Mos Eisley Street.

AV	D	N.Fury	Natasha	Tony	Banner	C.America	Loki	Thor	P.Coulson	Barton	Tesseract	Hb	Barton
B	N.Fury	Banner	Natasha	Hb	Loki	Tony	Thor	C.America	Bg	Bl	Tesseract	Bs	
Ei	C.America	Tony	N.Fury	Natasha	Thor	Banner	Loki	P.Coulson	Bl	Bs	Loki	Hb	
I.S	N.Fury	Natasha	Tony	Banner	C.America	Loki	Thor	P.Coulson	Barton	Tesseract	Hb	Barton	
SW	D	Luke	C-3PO	Han	Ben	Leia	Vader	Biggs	Tarkin	R.Leader	S	Sds	Lxfc
B	Luke	C-3PO	S	Han	A		Leia	Vader	Ben	Sds	Biggs	Mfc	Td
I.S	Luke	C-3PO	Han	Ben	Leia	Vader	Biggs	Tarkin	R.Leader	S	Sds	Lxfc	

6 Conclusion

In this paper, we introduce a multilayer model that relates *characters*, *locations* and *keywords* in movies. This model is much more informative than single layer networks which usually concentrate on *characters* or *scenes*. We also propose a methodology to extract the networks elements from the script. We have deployed the model on two popular movies. At the moment, the processing requires to clean the results. In order to get a fully automatic network extractor, we need to remove any ambiguity by including multimedia elements, such as scene segmentation, face tracking, and subtitle information. Due to space constraint, we reported the results of a brief analysis of the extracted networks that confirmed the effectiveness of the model. Note that much more information can be gained by a deeper topological analysis. To be the less ambiguous possible, we have considered only primitive interactions that are easy to interpret, but we could easily project different views. The model can be extended to derive a co-occurrence network of characters in the same location, a directed network of conversations, or mention of characters, etc.. Future work implies deploying our tool on larger collections, such as the whole Star Wars series, or even a larger collection of movies. We plan to use the multilayer network model to characterize movie period, genres, or directors, and even correlate with acting careers from public databases such as IMDB.

References

1. Tan, M.S.A., Ujum, E.A., Ratnavelu, K.: A character network study of two sci-fi TV series. In: AIP Conference Proceedings, vol. 1588, No. 1. AIP (2014)
2. Kivela, M., et al.: Multilayer networks. *J. Complex Netw.* **2**(3), 203–271 (2014)
3. Mish, B.: Game of Nodes: A Social Network Analysis of Game of Thrones (2016)
4. Kadushin, C.: Understanding social networks: theories, concepts, and findings. OUP, USA (2012)
5. Renoust, B., et al.: A social network analysis of face tracking in news video. In: 2015 11th International Conference on Signal-Image Technology and Internet-Based Systems (SITIS). IEEE (2015)
6. Renoust, B., Melanon, G., Viaud, M.-L.: Entanglement in multiplex networks: understanding group cohesion in homophily networks. In: Cham, Social Network Analysis-Community Detection and Evolution, pp. 89–117. Springer (2014)
7. Waumans, M.C., Nicodme, T., Bersini, H.: Topology analysis of social networks extracted from literature. *PloS one* **10**(6), e0126470 (2015)
8. Jhala, A.: Exploiting: structure and conventions of movie scripts for information retrieval and text mining. Springer, Berlin, Heidelberg (2008)
9. Yeung, M., Yeo, B.-L., Liu, B.: Extracting story units from long programs for video browsing and navigation. In: Proceedings of the Third IEEE International Conference on Multimedia Computing and Systems, 1996. IEEE (1996)
10. Jung, B., et al.: Narrative abstraction model for story-oriented video. In: Proceedings of the 12th Annual ACM International Conference on Multimedia. ACM (2004)
11. Weng, C.-Y., Chu, W.-T., Ja-Ling, W.: Rolenet: movie analysis from the perspective of social networks. *IEEE Trans. Multimed.* **11**(2), 256–271 (2009)

12. Park, S.-B., Kyeong-Jin, O., Jo, G.-S.: Social network analysis in a movie using character-net. *Multimed. Tools Appl.* **59**(2), 601–627 (2012)
13. Kipling, R.: Just So Stories, 1902. Initial letter by (2004)
14. Flint, L.N.: Newspaper writing in high schools: containing an outline for the use of teachers. Pub. from the Department of journalism Press in the University of Kansas (1917)
15. Omran, A., Fouad, N.A., Christoph, T.: Choosing an NLP library for analyzing software documentation: a systematic literature review and a series of experiments. In: *Software Repositories*. IEEE Press (2017)
16. Blei, D.M., Andrew, Y.N., Jordan, M.I.: Latent dirichlet allocation. *J. Mach. Learn. Res.* **3**, 993–1022 (2003)
17. Li, J., Zhang, K.: Keyword extraction based on tf/idf for Chinese news document. *Wuhan Univ. J. Nat. Sci.* **12**(5), 917–921 (2007)
18. Yuepeng, L., Cui, J., Junchuan, J.: A keyword extraction algorithm based on Word2vec. *E-sci. Technol. Appl.* **4**, 54–9 (2015)
19. Bioglio, L., Pensa, R.G.: Is this movie a milestone? identification of the most influential movies in the history of cinema. In: *International Workshop on Complex Networks and their Applications*, Springer, Cham (2017)
20. Rital, S., Cherifi, H., Miguet, S.: Weighted adaptive neighborhood hypergraph partitioning for image segmentation. Springer, Berlin, Heidelberg (2005)
21. Demirkesen, C., Cherifi, H.: A comparison of multiclass SVM methods for real world natural scenes. In: *Concepts for Intelligent Vision Systems*. Springer, Berlin, Heidelberg (2008)
22. Chen, B.-W., Wang, J.-C., Wang, J.-F.: A novel video summarization based on mining the story-structure and semantic relations among concept entities. *IEEE Trans. Multimed.* **11**(2), 295–312 (2009)
23. Pastrana-Vidal, R.R., et al.: Predicting subjective video quality from separated spatial and temporal assessment. In: *Human Vision and Electronic Imaging XI*, vol. 6057. International Society for Optics and Photonics (2006)
24. Domenico, M.D., Porter, M.A., Arenas, A.: Multilayer analysis and visualization of networks. *J. Complex Netw.* **10** (2014)
25. Lucas, G.: Star Wars, Episode IV: A New Hope. Twentieth Century Fox Home Entertainment (2006)
26. Whedon, J., Downey, R., Jr.: The Avengers. Walt Disney Studios Home Entertainment (2012)



Effects of Interaction and Learning Distance on Cooperation in Evolutionary Games on a Multiplex Network

Yasuyuki Nakamura^{1(✉)}, Koichi Yasutake², Keiya Ando³, and Takahiro Tagawa⁴

¹ Graduate School of Informatics, Nagoya University, Nagoya 464-8601, Japan
nakamura@nagoya-u.jp

² Graduate School of Social Sciences, Hiroshima University, Higashi-Hiroshima 739-8525, Japan
ystake@hiroshima-u.ac.jp

³ Faculty of Economics, Hiroshima University, Higashi-Hiroshima 739-8525, Japan
keiyando.0917@gmail.com

⁴ Research Institute for Information Technology, Kyushu University,
Fukuoka 819-0395, Japan
tagawa.takahiro.855@m.kyushu-u.ac.jp

Abstract. We investigated a two-layered multiplex network in which players interact depending on the social distance between them. The aim of this study is to clarify the effects of multi-layered structure on emergence of cooperation in the prisoner's dilemma game including social distances, interactions, and learning distances. We found that the increase of learning distance promotes cooperation even on a multiplex network, and a cooperative strategy tends to vanish away with the parameters, which is contract to the Seltzer's result.

Keywords: Multiplex network · Prisoner's dilemma game
Social distance · Cooperation

1 Introduction

Cooperative behaviors, such as the herd behavior of animals and the colony formation of insects, play an important role in nature. Such relationships are often seen in human communities as well, and cooperation is the key behavior to establish a stable and well-ordered society. While cooperative behavior is non-trivial, and instead a defective behavior is equilibrium as we see in the prisoner's dilemma game, cooperative behavior emerges in real society even if the same situation occurs with the prisoner's dilemma game. To solve this inconsistency, the origin or mechanism of the emergence of cooperation has been investigated. Since Nowak and May revealed that the structures of interactions among agents affect the emergence of cooperation [1], the prisoner's dilemma on

complex networks has been investigated by many authors [2–5]. Although evolutionary games among directly connected players are examined in these studies, it is unrealistic that interactions are restricted to those directly connected. Seltzer and Smirnov analyzed a social network in which players make contributions to others, depending on the social distance between two players which they refer to as ‘Interaction Distance’ or ‘Learning Distance’ [6]. They found that cooperation can be accelerated when players interact with others in ‘long’ social distance.

These studies have mainly investigated cooperative behaviors on a single network; however, in the real world, each player usually belongs to different communities, and community networks affect each other directly or indirectly, thus affecting the cooperative behavior among agents. These interacting networks, known as multilayer networks or multiplex networks, are examples of networks of networks [7], and they have been garnering significant attention in recent years.

Gómez-Gardeñes et al. coupled the evolutionary dynamics of the prisoner’s dilemma in each network of multiplex networks, and they showed that the resilience of cooperation for large values of the temptation to defect is enhanced by the multiplex structure [8]. Wang et al. studied the impact of biased utility functions on interdependent networks in [9] and investigated the emergence of interdependent network reciprocity in evolutionary games in [10]. They also found the optimal interdependence for cooperation to be prompted [11], while it was shown that the interdependence between networks is self-organized to yield optimal conditions for the evolution of cooperation [12]. In [13], two-layer scale-free networks with all possible combinations of degree mixing were investigated, and it was found that degree mixing in multilayer networks impedes the evolution of cooperation. The evolution of cooperation in evolutionary games on multilayer networks was reviewed in [14].

As far as we know, there is no investigation to multiplex network in which players interact depending on the social distance between them. A network model considering the social distance between players is a realistic but previously unexplored model of cooperation, before Seltzer investigated. The social distance, which we discuss in the next section, can be measured by the degrees of separation, and defined as the number of connections between any two agents. For example, two people who know each other are separated by one degree, two people who do not know each other but have a common acquaintance are separated by two degrees, and so on. Furthermore, even if people do not connect directly, they may learn the behavior (strategy) from non-directly-connected people. Therefore, it is important to investigate a network model considering social distance. The aim of this study is to clarify the effects of multi-layered structure on emergence of cooperation in the prisoner’s dilemma game including social distances, interactions, and learning distances, which is a simple extension of Seltzer’s model.

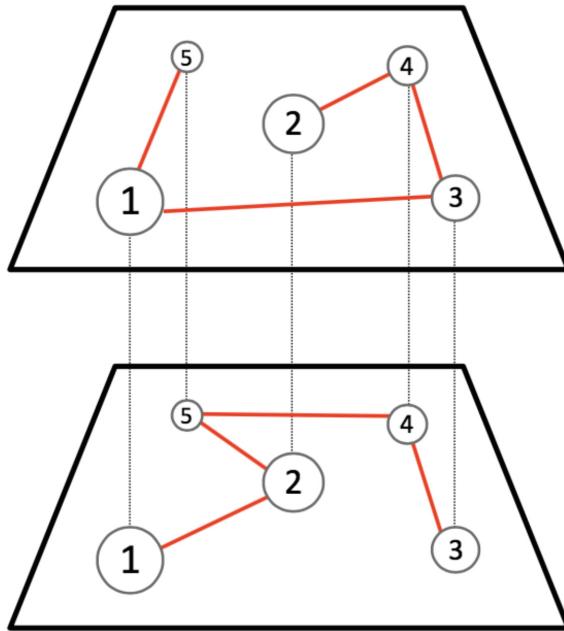


Fig. 1. An example of a two-layered multiplex network in which there are five nodes or players in each network. Red lines represent interactions between players and nodes connected with black dotted lines between two layers are identical players.

2 Model

2.1 Network Structure, Social Distance, Strategy and Payoff

To avoid effects other than those from network structure, we adopted the same rules with Seltzer's model on each network involved in multiplex network. We constructed a two-layered multiplex network (Fig. 1) composed of two small-world networks. We used the Watts–Strogatz model for the small-world network [5] and placed N nodes in circular order, and the k nodes that are the nearest from each node are selected and connected to each other. Each link is selected with probability p and reconnected to a randomly selected second node. In our study, we set $N = 400$, $k = 4$, and $p = 0.1$.

As Seltzer did, we introduced “Interaction Distance (ID)” and “Learning Distance (LD)” as social distances. ID defines a maximum degree of separation between two agents, which means each agent plays the game with agents within the degree of separation ID . For example, in the case of $ID = 4$, each agent plays with an agent who is randomly selected among agents in the distance between degrees 1 and 4. LD is defined as the range of degrees of separation, in which agents learn a strategy of selected agents. In the case of $LD = 4$, each agent updates his/her strategy referring the strategy of agents who are separated at 4th degree of distance. Note that ID and LD are common in all agents.

Each agent i has a ID -dimensional strategy vector \mathbf{S}^i . The d -th element of \mathbf{S}^i , which we call d -th strategy, is an i -th agent's propensity to cooperate with neighbors separated by d degrees of separation. For example, for $ID = 4$, the strategy vector is $\mathbf{S}^i = (S_1^i, S_2^i, S_3^i, S_4^i)$. Each agent's initial propensity to cooperate for all types of neighbors $S_d^i, i = 1, \dots, N, d = 1, \dots, ID$ is drawn randomly from the uniform probability distribution $U[0, 1]$.

When an agent i plays with agent j separated by d degrees, the payoff P_{ij} to an agent i is defined as

$$P_{ij} = bS_d^j - cS_d^i, \quad (1)$$

where b and c are the benefit and cost of cooperation ($b > c$). Therefore, the closer S_d^i is to 1, the more cooperative the agent i is, and the closer S_d^i is to 0, the more defective the agent i is.

2.2 Prisoner's Dilemma Game on a Multiplex Network

The algorithm of the game is as follows:

1. A two-layered multiplex network is constructed and the agent first plays the game on layer L_1 , and then plays the next game on layer L_2 .
2. Each agent determines a ID -dimensional strategy vector \mathbf{S}^i whose elements S_d^i are drawn randomly from the uniform probability distribution $U[0, 1]$ on layer L_1 .
3. Each agent selects an integer d from 1 to ID randomly and the agent plays with one of the agents connected in the d -th degree of distance. Then the payoff of each agent is calculated according to the Eq. (1).
4. After all agents play, each agent updates his/her strategy as follows: Each agent imitates the strategy of the agent whose payoff is highest within LD -th degree of distance. In addition to imitation, each agent strategy is also subject to random mutation. The agent's strategy is regenerated according to the same initial probability distribution with α , which is so-called mutation level.
5. The updated strategy of each agent on layer L_1 is used in layer L_2 in the next game. On the completion of the game, the strategy of each agent is updated in the same way as shown in Step 4.
6. The updated strategy on L_2 is transferred to L_1 .
7. Steps 3, 4, 5, and 6 are repeated for 400 generations.

3 Results

We carried out experiments with the constructed model. We set the number of nodes to $N = 400$. The benefit and cost of cooperation parameter b and c are set to $b = 6$ and $c = 1$. The mutation rate α is assumed to be 0.001. The calculations are averaged after 100 simulations.

Firstly, we carried out simulations with fixed $LD = 1$; Fig. 2 shows the results with different ID from 1 to 4. Seltzer had reported that agents' strategy became cooperative in the case of $LD = ID = 1$ with single layer small-world network [6]. This result is easily explained by the study by Masuda [15]. The main mechanism of emerging of cooperation comes from the emerging of cluster of cooperative agents. When the number of layer becomes two, however, cooperative strategy is transferred to the other layer. An agent with cooperative strategy may play with a defective agent, because their network structures are different from each other. Then the cooperative agent could be exploited by a defective agent. Then the cluster of cooperative agents become disrupted. This is why Fig. 2 shows that cooperative strategy tends to vanish away with $LD = ID = 1$. This is an influence of the multiplex network structure.

In Seltzer's calculation, as ID becomes large, cooperation level of first strategy is decreased. In our simulation, on the contrary, as ID becomes large, the level of cooperation of first strategy is increased asymptotically, as shown in Fig. 2. We could not find any difference in behaviors of 2nd, 3rd, and 4th strategy between the Seltzer's results and ours.

Figure 3 shows that all strategies become cooperative as LD becomes large. This result is consistent with Seltzer's results.

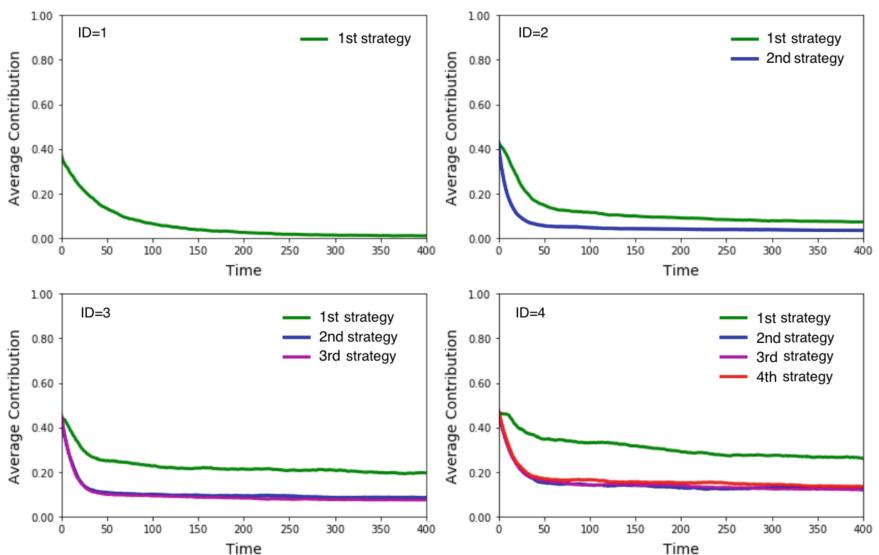


Fig. 2. Time evolution of averages of cooperation with $LD = 1$ as a function of ID . Colored lines are referring to d -th strategy.

4 Conclusion

We constructed a two-layered multiplex network composed of two small-world networks. Our model is a straightforward extension of the Seltzer's model. We examined the effect of the multi-layer by carrying out simulations, and we found two facts: one is that the increase of the learning distance LD promotes cooperation even on multiplex network; the other is that a cooperative strategy tends to vanish away with the parameter $LD = ID = 1$, which is contract to Seltzer's result.

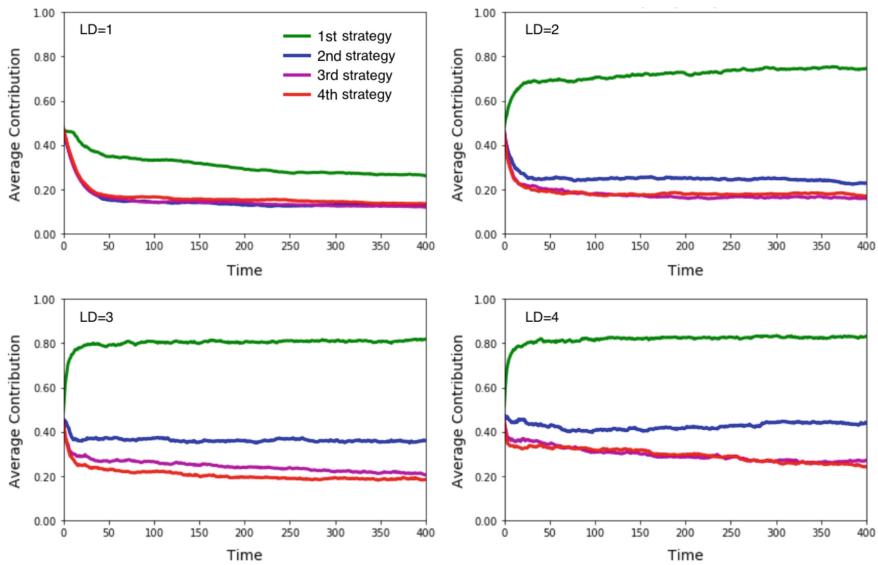


Fig. 3. Time evolution of averages of cooperation with $ID = 4$ as a function of LD . Colored lines are referring to d -th strategy.

Our results are still preliminary, because our simulation is restricted to the special case. However, the case is set in order to compare with Seltzer's result. We fixed the parameters b and c for the calculation of payoff and the parameters of k and p for the nature of small-world network. In the next stage, we should carry out simulation with comprehensive situations. Dependence on the number of layers could also be relevant to explore. Our simulation is restricted to small-world multiplex networks. It would be interesting to investigate what happens if Erdős–Rényi random network or scale-free network is chosen. This could be a future work.

Acknowledgement. This research was financially supported by JSPS KAKENHI grant numbers 18H01052, 18K18653 and 17K01135.

References

1. Nowak, M.A., May, R.M.: Evolutionary games and spatial chaos. *Nature* **359**, 826–829 (1992)
2. Santos, F.C., Pacheco, J.M.: Scale-free networks provide a unifying framework for the emergence of cooperation. *Phys. Rev. Lett.* **95**, 098104 (2005)
3. Szabó, G., Fáth, G.: Evolutionary games on graphs. *Phys. Rep.* **446**, 97–216 (2007)
4. Perc, M., Szolnoki, A.: Coevolutionary games—a mini review. *Biosystems* **99**, 109–125 (2010)
5. Watts, D.J., Strogatz, S.H.: Collective dynamics of ‘small-world’ networks. *Nature* **393**, 440–442 (1998)
6. Seltzer, N., Smirnov, O.: Degrees of separation, social learning, and the evolution of cooperation in a small-world network. *J. Artif. Soc. Soc. Simul.* (2015). <https://doi.org/10.18564/jasss.2851>
7. Kenett, D.Y., Perc, M., Boccaletti, S.: Networks of networks—an introduction. *Chaos Solitons Fract.* **80**, 1–6 (2015)
8. Gómez-Gardeñes, J., Reinare, I., Arenas, A., Floría, L.M.: Evolution of cooperation in multiplex networks. *Sci. Rep.* (2012). <https://doi.org/10.1038/srep00620>
9. Wang, Z., Szolnoki, A., Perc, M.: Evolution of public cooperation on interdependent networks: the impact of biased utility functions, *EPL* (2012). <https://doi.org/10.1209/0295-5075/97/48001>
10. Wang, Z., Szolnoki, A., Perc, M.: Interdependent network reciprocity in evolutionary games. *Sci. Rep.* (2013). <https://doi.org/10.1038/srep01183>
11. Wang, Z., Szolnoki, A., Perc, M.: Optimal interdependence between networks for the evolution of cooperation. *Sci. Rep.* (2013). <https://doi.org/10.1038/srep02470>
12. Wang, Z., Szolnoki, A., Perc, M.: Self-organization towards optimally interdependent networks by means of coevolution. *New J. Phys.* (2014). <https://doi.org/10.1088/1367-2630/16/3/033041>
13. Wang, Z., Wang, L., Perc, M.: Degree mixing in multilayer networks impedes the evolution of cooperation. *Phys. Rev. E* (2014). <https://doi.org/10.1103/PhysRevE.89.052813>
14. Wang, Z., Wang, L., Szolnoki, A., Perc, M.: Evolutionary games on multilayer networks: a colloquium. *Eur. Phys. J. B* (2015). <https://doi.org/10.1140/epjb/e2015-60270-7>
15. Masuda, N., Aihara, K.: Spatial prisoner’s dilemma optimally played in small-world networks. *Phys. Lett. A* **313**, 55–61 (2003)

Resilience and Control



A Genetic Algorithm for Enhancing the Robustness of Complex Networks Through Link Protection

Clara Pizzuti^(✉) and Annalisa Socievole

National Research Council of Italy (CNR), Institute for High Performance Computing and Networking (ICAR), Via Pietro Bucci, 7-9C,
87036 Rende (CS), Italy
{clara.pizzuti,annalisa.socievole}@icar.cnr.it

Abstract. An important challenge in complex networks is the improvement of network robustness. Electrical networks, water/gas networks and telecommunication networks are representative examples of infrastructures distributing critical resources for our society that require high level of robustness. In this paper, we propose a method based on Genetic Algorithms to enhance network robustness focusing on the protection of the link whose removal would severely increase the effective graph resistance. Derived from the field of electric circuit analysis, effective graph resistance is a robustness measure that can be computed as a cumulative sum of the inverses of the $N - 1$ largest eigenvalues of the Laplacian matrix associated with the network. Simulations on real-world and synthetic networks show that our method in most cases equals the exhaustive search and also outperforms other heuristic strategies.

Keywords: Network robustness · Genetic algorithm · Graph resistance

1 Introduction

The improvement of the structural robustness of a network is an important challenge in the research area of complex networks. Many application domains such as telecommunication networks, transportation networks, electric power grids and water/gas management networks require proper strategies to achieve high robustness. Robustness has been defined as the capability of a network to cope with threats and survive to accidental events [13]. Usually, networks are subject to failures that can be accidental or intentional, as random or targeted attacks. These last attacks address specific nodes and/or links in order to decrease the network throughput.

How to measure the robustness of a network is still an open issue. Several measures have been proposed so far. Early measures were based on the concept of graph connectivity [5], such as vertex and edge connectivity, size of the largest

connected component, and on the spectral properties of a graph [4]. Under node or link removal, however, the former measures do not well express the capability of a network to preserve a given connectivity. Other improved measures have been proposed including super connectivity, fault diameter, tenacity and isoperimetric number [13]. The problem of computing these measures in general graphs, however, is computationally expensive and thus of no practical use in complex networks.

An alternative formulation of robustness emerged from graph spectra theory. Fielder [4] introduced the concept of algebraic connectivity, defined as the second smallest eigenvalue of the Laplacian matrix of a graph, which measures how well connected a graph is. If the algebraic connectivity is large, the graph usually remains connected. Wu et al. [13] proposed another measure based on graph spectra, called natural connectivity, which characterizes the redundancy of alternative routes in a network by computing the weighted number of closed walks of all the lengths. This number can be obtained as a sort of average eigenvalue of the graph adjacency matrix. Another spectral measure is the *effective graph resistance* proposed by Ellens et al. [3]. Denoted with R_G , the effective graph resistance is a robustness measure derived from the field of electric circuit analysis. Differently from the aforementioned robustness measures, the effective graph resistance has been shown to be a suitable index for evaluating the robustness of a network subject to link addition or removal for different reasons. First, R_G is able to well measure the closeness of two nodes i and j and the ease of communication in a graph [6]. For low R_G values, in fact, there are many paths between nodes, while for high R_G values the number of paths is low. Abbas and Egerstedt [1] observed that the existence of multiple paths between nodes means that these nodes are highly interconnected, thus they are resilient to node/edge failures. Second, R_G takes also into account the quality of the paths. R_G is related to random walks being proportional to the expected time to reach any vertex j from a vertex i , averaged over all the node pairs. As such, when the effective graph resistance is low, the paths are shorter and hence the connectivity is better as well as the end-to-end delay in communication. In other words, the network with these paths is less affected by failures. It has been shown that the R_G value decreases when edges are added to a graph and increases when edges are removed. In the first case, adding a link for infrastructural investments would increase the system lifetime. R_G would thus suggest where adding a link. In the second case, R_G would be helpful in finding which link to protect when having an infrastructure subject to cyber-physical targeted attacks, for example.

In this paper, we focus on the problem of finding the link to protect whose removal would maximize the effective graph resistance. We formulate this problem as an optimization problem solved through a Genetic Algorithm (GA) [7], a powerful search technique for solving combinatorial problems. Our method, named *RobLPGA*, represents each possible solution/individual with vectors of size 2 denoting the two edge nodes of the link to be protected. This simple representation is efficient and effective in finding the solution that gives the maximum R_G , used as fitness function. Proposing specialized crossover and mutation

operators, the method is able to find solutions almost always coinciding with the solution provided by the exhaustive search. By comparing *RobLPGA* to a set of heuristics investigated in [12] over real-world and synthetic networks, our method shows its superiority in 11 out of the 12 considered networks.

The paper is organized as follows. In Sect. 2, we describe the most recent approaches related to the improvement of robustness in networks. Section 3 defines the concept of effective graph resistance and formalizes the problem to solve. Section 4 describes the proposed approach. Section 5 presents the experimental results. Finally, Sect. 6 concludes the paper and discusses future works.

2 Related Work

The improvement of robustness in networks is usually related to the study of topological perturbations, like the addition or removal of nodes or links, or the rewiring of links [9]. Through proper perturbations, in fact, the robustness of a network can be preserved or improved. For example, Wang et al. [12] provide theoretical and experimental findings on network robustness under two scenarios: (i) addition of a link that maximally decreases the effective graph resistance and (ii) protection of a link whose removal would maximally increase the effective graph resistance. Four strategies for adding or removing links are evaluated on both real-world and synthetic traces showing the high impact that the perturbations have on network robustness. Buesser et al. [2] optimize the robustness of scale-free networks subject to malicious attacks to highly connected nodes by using the simulated annealing optimization heuristic. The network is properly rewired in order to maintain node connectivity and degree distribution, while optimizing the robustness computed through the R value, defined by Herrmann et al. [8] as the sums of the fraction of nodes in the largest connected cluster after removing nodes.

Regarding evolutionary approaches for improving robustness, Zhou and Liu [14] propose a memetic algorithm that optimizes the R value over scale-free networks. Similarly to [2], this work optimizes robustness in situations where there are attacks to important nodes having high degree without changing the degree distribution of the network. Through an ad hoc crossover operator performing global search and a local search operator, the proposed algorithm is able to search the most robust network structure. In another work [11], Wang and Liu focus on enhancing the robustness of scale-free and Erdős-Rényi networks under attacks to edges causing cascading failures. The authors define a memetic algorithm exploiting the same chromosome representation and crossover used in [14] with a local search operator employing simulated annealing as in [2]. As robustness measure, a new measure, based on the aforementioned R measure and adapted to cascading failures, is proposed. Note that, differently from [14] and [11], we improve robustness by protecting the link whose removal would result in the highest effective graph resistance and not through rewirings. Moreover, we perform simulations on a more heterogeneous set of complex networks.

3 Effective Graph Resistance and Optimization Through Link Removal

Effective graph resistance is a robustness measure derived from the field of electric circuit analysis. Considering the graph as an electrical circuit where edges are resistances with value 1, the effective graph resistance is defined as the sum of the effective resistances between all the pairs of nodes. Given an undirected graph $G = (V, E)$ with N nodes, the adjacency matrix A of the graph is a symmetric $N \times N$ matrix with elements $a_{ij} = 1$ if there is an edge between i and j , 0 otherwise. The $N \times N$ diagonal degree matrix $\Delta = \text{diag}(d_i)$, where $d_i = \sum_{j=1}^N a_{ij}$, is used to define the *Laplacian* Q of G which is an $N \times N$ symmetric matrix $Q = \Delta - A$ with elements

$$Q_{ij} = \begin{cases} d_i & \text{if } i = j \\ -1 & \text{if } (i, j) \in E \\ 0 & \text{otherwise.} \end{cases} \quad (1)$$

The effective graph resistance R_G is defined in terms of graph spectra as

$$R_G = N \sum_{k=1}^{N-1} \frac{1}{\mu_k} \quad (2)$$

where μ_k is the k -th largest eigenvalue of Q .

The effective graph resistance strictly increases if a link is removed from the graph [3]. As a consequence, for improving the robustness of a network focusing on link removal as network perturbation, it is necessary to protect the link $e = (i, j)$ whose removal would maximize the effective graph resistance of $G - \{e\}$. More formally, we formulate the problem of enhancing robustness as follows.

Problem: Given an undirected graph $G = (V, E)$ with N nodes and L links, find the link $e = (i, j) \in E$ such that

$$R_{G-\{e\}} > R_{G-\{e'\}} \quad (3)$$

for any other edge $e' = (k, l) \in E$.

In the next section, we propose a method based on GAs for solving this optimization problem.

4 Method Description

The *Robustness Link Protection* method *RobLPGA* is a Genetic Algorithms based method which evolves a population of individuals by performing variation and selection operators to increase the value of the effective graph resistance as objective function, while exploring the research space during the optimization process. In the following, we specify how *RobLPGA* represents the individuals of

the population and how they evolve through crossover and mutation operators. Finally, we describe the various steps of the algorithm.

Representation: an individual of the population consists of a vector of 2 genes where each gene can assume a value in the range $\{1, \dots, N\}$. A value (i, j) for an individual represents the link between the nodes i and j to be removed from the graph G . Initially, the population of individuals is initialized with pairs of nodes chosen between all the existing links of the graph.

Crossover: given two parents $e_1 = (i, j)$ and $e_2 = (k, l)$, the crossover operator generates a child e_3 as follows. To avoid redundant checks, the couples of nodes are initially sorted in order to have $i < j$, for each couple (i, j) . If $e_1 = e_2$, no child is generated. If $i \neq j \neq k \neq l$ (Fig. 1(a)), e_3 is randomly chosen between the links $\{(i, k), (i, l), (j, k), (j, l)\}$, provided that these links belong to the set of edges E . If none of the four links are edges of G , the crossover operator does not generate any child. Figs. 1(b–e) show the cases in which two links share one node. There are four cases to consider: (1) $i = k$, (2) $i = l$, (3) $j = k$ and (4) $j = l$. In any case, the child that will be generated is depicted with dotted lines. If the links with the dotted lines do not belong to the set of edges of G , the child will be one of the two parents.

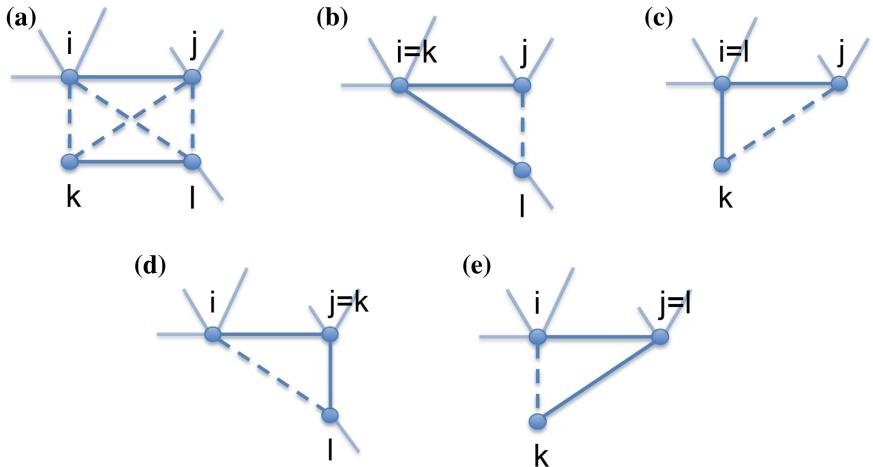


Fig. 1. Crossover operator: (a) the two parents (i, j) and (k, l) have distinct nodes. (b)–(e) the two parents share a node.

Mutation: given an individual (i, j) , the mutation operator disconnects i from j and connects it to another node k randomly chosen among its neighbors.

RobLPGA pseudo-code: Figure 2 shows the various steps of the algorithm. Given a input graph $G = (V, E)$ and a maximum number T of generations, *RobLPGA* initializes the population at random by choosing individuals representing existing edges. Then, for a number T of generations, for each individual e of the population, *RobLPGA* computes the Laplacian of $G - \{e\}$ and evaluates

the fitness function $R_{G-\{e\}}$. A new population of individuals is then created by applying the variation operators. Finally, the link e maximizing $R_{G-\{e\}}$ is returned as solution to the problem.

Computational Complexity: the computational complexity of the algorithm is $O(T \times P \times N^3)$, where T is the number of generations and P the population size. In fact, the computation of the effective graph resistance R_G needs the computation of the eigenvalues of the adjacency matrix of G , which is of the order $O(N^3)$. It is worth pointing out that R_G can also be computed by using the Moore-Penrose pseudo-inverse Q^+ of the Laplacian matrix Q of A as N times the trace of Q^+ [3]. As highlighted by Ranjan et al. [10], Q^+ can be computed as

$$Q^+ = (Q + \frac{1}{N} J)^{-1} - \frac{1}{N} J \quad (4)$$

The *RobLPGA* Method:

Input: A graph $G = (V, E)$ representing a network, maximum number of generations T

Output: A link $e = (i, j) \in E$

```

1 Initialize a population of random individuals choosing edges from  $E$ 
2 while termination condition is not satisfied do
3   for each individual  $e = (i, j)$  of the population,
     compute the Laplacian of  $G - \{e\}$ 
4   evaluate the objective function  $R_{G-\{e\}}$ 
5   create a new population of individuals by applying the variation operators
6 end while
7 Return the individual  $e$  which removed from  $G$  gives the highest value
     of the effective graph resistance

```

Fig. 2. The pseudo-code of the *RobLPGA* algorithm.

where $J \in R^{N \times N}$ is a matrix of all 1's. Though Eq. (4) requires also $O(N^3)$ computation time, in [10] it has been shown experimentally that, in practice, it is much faster than computing Q^+ through an eigen-decomposition of Q . Moreover, these authors propose an incremental computation of the pseudo-inverse of Laplacian when adding or deleting an edge. Thus, after computing the pseudo-inverse Q^+ of Q , once at the beginning of the algorithm, the fitness function can be evaluated in linear time since the pseudo-inverse associated with $G - \{e\}$ can be obtained by Q^+ as linear combination of the elements of Q^+ (for details refer to [10]). Finally, it is worth pointing out that *RobLPGA* could be used with large networks by implementing it on a parallel machine, considering the intrinsic natural parallelization of population-based methods.

5 Experimental Evaluation

In this section, we describe the experiments performed over *RobLPGA* to obtain the edge that ensures the best graph robustness. Specifically, we detail the features of the datasets considered and the results obtained by comparing *RobLPGA*

to other strategies. All the algorithms have been written in Matlab, version 2015b. For *RobLPGA* we exploited the Genetic Algorithm solver implemented in the Global Optimization Toolbox. As genetic parameters, we set the crossover fraction to 0.9, the mutation rate to 0.2, the maximum number of generations to 200 and the population size to 100. These parameters have been obtained by applying a trial-and-error procedure. In the experiments, we first compare *RobLPGA* to the case in which no link is removed. Then, to test its effectiveness, we compare the robustness obtained by *RobLPGA* and other strategies maximizing the effective graph resistance when a link is removed with the effective graph resistance obtained by the exhaustive search of the optimal link to remove.

5.1 Datasets

We consider an heterogeneous set of real-world (Internet backbones, Facebook Ego networks and a power grid) and synthetic networks (Erdős-Rényi, Watts-Strogatz and Bárabasi-Albert networks) whose topological characteristics are described in Tables 1 and 2. In the following, we discuss their main features.

- **Internet Backbones.** We selected from the repository *Internet Topology Zoo*¹, which contains hundreds of Internet backbones topologies, 5 networks with different number of nodes and densities, namely *Bell South*, *ASNET-AM*, *ITC Deltacom*, *ION*, and *US Carrier*. Each of these networks has been converted from a .gml format into an adjacency matrix through a Python script. On average, each node which corresponds to a BGP (Border Gateway Protocol) router, is connected to one or two routers. Usually these networks are characterized by a core backbone with the highest degree to which are connected peripherals nodes with lower degree. We study such networks since they are often subject to several attacks like blackholing or traffic redirection that severely influence their stability.
- **Facebook Ego Networks.** This set of real-world networks represents the egos (or social profiles) of Facebook users. Typically, Facebook networks are composed by many egos having several friends connected to other egos having friends in common. We analyze three egos, the *Ego 3980*, the *Ego 686*, and the *Ego 3437*. The *Ego 3980* has a small number of nodes and low average degree, while the *Ego 686* and *Ego 3437* are larger egos with a higher and similar average degree, and a significantly higher number of links. For each network, we compute the largest connected component since some nodes (8 nodes in *Ego 3980*, 2 nodes in *Ego 3437* and *Ego 686*) are isolated nodes. We thus report in Table 1 the statistics of the largest connected components. We consider these networks in our analysis since Facebook profiles may be hacked causing the spreading of fake news/comments.
- **US Power Grid.** The *Western States Power Grid*² is composed by 4941 nodes representing either a generator, a transformer or a substation, and edges

¹ <http://www.topology-zoo.org/>.

² <http://konect.uni-koblenz.de/networks/opsahl-powergrid>.

representing power supply lines. Such networks may experience cascading failures causing large-scale blackouts.

- **Erdős-Rényi random networks.** Erdős-Rényi networks are synthetic networks where nodes are connected between them with probability p_c . Starting with isolated nodes, edges between pairs of nodes are added at random. In our simulations, we generated a network of 128 nodes with $p_c = 0.03$. With this connection probability, we obtain a connected graph. Erdős-Rényi networks are the simplest possible model for peer-to-peer networks and ad-hoc networks.
- **Watts-Strogatz small-world networks.** This class of synthetic networks are clustered random networks with the feature of having nodes easily reachable in few hops by the other nodes. Watts-Strogatz network usually model Bluetooth or Wi-Fi contact networks. Starting from a ring lattice of N nodes, each node is then connected to k nodes by rewiring each edge with probability p . Here, we generated a network with $k = 6$ and $p = 0.5$.
- **Bárbasi-Albert power law networks.** Bárbasi-Albert networks are strategic network models where the choices of connecting nodes depend from the current network structure. Having the *preferential attachment* feature, nodes tend to connect to high degree nodes. Such networks are generated from an initial connected graph of m_0 nodes where subsequently, a new node is connected to $m \leq m_0$ nodes with a probability proportional to the degree of the existing nodes. The Bárbasi-Albert network analyzed has been generated setting $m_0 = 5$ and $m = 2$. We study this network class for its feature of well modeling social networks, the Internet and the World Wide Web.

Table 1. Topological features of real-world networks: number of nodes (N), number of links (L), average degree ($\langle k \rangle$), average clustering coefficient ($\langle C \rangle$) and density (D).

Network	ID	N	L	$\langle k \rangle$	$\langle C \rangle$	D
Bell south	BS	51	66	1.294	0.081	0.052
ASNET-AM	AA	65	77	1.184	0.063	0.037
ITC Deltacom	ITD	113	161	1.425	0.053	0.025
ION	ION	125	146	1.168	0.006	0.019
US carrier	USC	158	189	1.196	0.002	0.015
Ego 3980	3980	44	138	3.136	0.227	0.072
Ego 686	686	168	1656	9.8572	0.266	0.059
Ego 3437	3437	532	4812	9.045	0.272	0.017
US power grid	USPG	4941	6594	2.669	0.103	5.403e-04

Table 2. Topological features of synthetic networks.

Network	ID	N	L	$\langle k \rangle$	$\langle C \rangle$	D
Erdős-Rényi	ER	128	625	5.23	0.08	0.076
Watts-Strogatz	WS	128	384	6	0.114	0.047
Bárbasi-Albert	BA	128	253	3.953	0.171	0.031

5.2 Strategies for Link Removal

The best strategy for finding the link to remove is the exhaustive search since it checks all the possible links and computing the corresponding effective graph resistance is able to find the link maximizing it. Its computational cost, however, is high having a complexity $O(N^5)$. Alternative and lower-cost strategies that are able to determine the link to remove have been proposed by Wu et al. [12]. In the following we describe those used in our work for performance comparison.

- S_1 (semi-random): the link $e(i, j) \in E$ has i with the minimum degree and j is randomly chosen. The complexity of the strategy is $O(N^2 - N + L + 1)$.
- S_2 (degree product): the link $e(i, j)$ is chosen among the links in E having the minimum degree product $d_i d_j$. This strategy has complexity $O(N^2 - N + 2L)$.
- S_3 (Fiedler vector): the link $e(i, j)$ is chosen using the Fiedler vector y corresponding to the eigenvector of the second smallest eigenvalue of the Laplacian matrix of G . Nodes i and j are chosen as the couple having the maximum difference $|y_i - y_j|$, where y_i and y_j are the components i and j of the Fiedler vector, respectively. The complexity of S_3 is $O(N^3 + 2L)$.

5.3 Results

As first experiment, we compared *RobLPGA* and the other algorithms on the real-world networks by considering as possible links to be removed all the links that are present in the original network graph. Table 3 shows the effective graph resistance values computed in the original network (R_G) and those obtained by removing the link $e = (i, j)$ and applying the exhaustive search, denoted by $R_{G-\{e\}}^*$, the three strategies S_1 , S_2 , S_3 , namely $R_{G-\{e\}}^{S1}$, $R_{G-\{e\}}^{S2}$ and $R_{G-\{e\}}^{S3}$, respectively, and our approach, $R_{G-\{e\}}^{RobLPGA}$. We indicate the link selected by each algorithm with the IDs of the two nodes between square brackets. We can first observe that the removal of a link, in general, severely increases the effective graph resistance. As expected, if comparing R_G to the values obtained by all the other algorithms, we obtain effective graph resistance values significantly higher. The exhaustive search is able to find the link having the maximum effective graph resistance. The infinite value over all the networks indicates that the removal of the selected link disconnects the network. Similarly, *RobLPGA* is able to find the link resulting in the highest effective graph resistance over all the networks. Considering the group of Internet Backbones, strategies S_1 and S_2 are also

Table 3. Comparison of effective graph resistance in the original network (R_G) and in the network with a link removed resulting from the exhaustive search ($R_{G-\{e\}}^*$), heuristic $S1$ ($R_{G-\{e\}}^{S1}$), $S2$ ($R_{G-\{e\}}^{S2}$), heuristic $S3$ ($R_{G-\{e\}}^{S3}$) and $RobLPGA$ ($R_{G-\{e\}}^{RobLPGA}$) over real-world networks. The link selected by each algorithm with the IDs of the two nodes is reported in square brackets.

ID	R_G	$R_{G-\{e\}}^*$	$R_{G-\{e\}}^{S1}$	$R_{G-\{e\}}^{S2}$	$R_{G-\{e\}}^{S3}$	$R_{G-\{e\}}^{RobLPCA}$
BS	3.22e+03	∞ [3 43]	9.989e+16 [1 49]	∞ [3 43]	3.793e+03 [11 36]	∞ [37 47]
AA	6.005e+03	∞ [6 8]	1.082e+17 [1 2]	∞ [46 49]	1.189e+17 [23 37]	∞ [37 40]
ITC	1.984e+04	∞ [79 102]	1.862e+16 [39 40]	1.862e+16 [39 40]	2.316e+04 [25 28]	∞ [79 102]
ION	3.67e+04	∞ [4 5]	∞ [4 5]	1.505e+17 [54 124]	4.08e+04 [101 106]	∞ [35 105]
USC	8.361e+04	∞ [1 86]	∞ [1 86]	2.735e+17 [41 44]	1.051e+05 [31 132]	∞ [73 77]
3980	793.399	∞ [4 28]	∞ [4 28]	∞ [14 36]	917.06 [16 17]	∞ [35 39]
686	4.942e+03	∞ [12 145]	∞ [12 145]	1.618e+16 [62 70]	4.807e+15 [52 70]	∞ [12 145]
3437	5.284e+04	∞ [69 513]	2.7417e+16 [154 157]	∞ [360 440]	5.375e+04 [25 371]	∞ [430 442]
USPG	6.377e+07	-	∞ [10 11]	∞ [4070 4164]	7.035e+07 [559 2824]	∞ [2774 2775]

effective and outperform $S3$. Even if some resistance values are not infinite, the high values (e.g. $9.989e + 16$ in the case of $S1$ over the *Bell South* network) of effective graph resistance are related to links that disconnect the network in two components. In general the number of zeros in the eigenvalues of the Laplacian matrix of the network graph denotes the number of network components. For these high values of effective graph resistance, we observed that, because of approximation errors, the eigenvalues of the Laplacian matrix are zero or a very low value near to zero. On the contrary, low effective graph resistance values such as $3.793e + 03$ for $S3$ in the *Bell South* network indicate that the network remains connected after the link removal. The effective graph resistance measure thus suggests to mainly protect the link that may disconnect the network.

Looking at the Facebook networks, on *Facebook Ego 3980*, for example, all the strategies except $S3$ are able to identify the links that maximize the effective graph resistance. On the other two networks, the results are similar to the Internet Backbones where *RobLPGA* equals the exhaustive search and $S1$ and $S2$ have also a good performance. In the *USPG* network, where the number of nodes is significantly higher, we could not compute $R_{G-\{e\}}^*$ due to the high computing time, but observing that $S1$, $S2$ and *RobLPGA* have infinity values for $R_{G-\{e\}}$, we can conclude that also the exhaustive search would have obtained an infinite value. Overall, considering the real-world networks, the best strategy (excluding the exhaustive search) is *RobLPGA*, followed by $S2$, $S1$ and $S3$.

As second experiment, we evaluated the class of synthetic networks. In all the cases, since these networks are denser than the real-world networks considered, each strategy finds links that do not disconnect the network. *RobLPGA* performance is the best over Erdős-Rényi and Bárabasi-Albert networks finding the same link of the exhaustive search and achieving an equal value of effective graph resistance. In the Watts-Strogatz network, $S3$ performs the best finding the link [10 126] which has been also selected by the exhaustive search. Here, node 126 is an example of important node for network robustness, also $S2$ selects this node when looking for the link to remove. *RobLPGA* performance in this case is comparable to $S2$, with $S1$ performing the worse. Compared to the real-world networks, we observe that $S3$ performs slightly better than $S2$, with $S1$ performing the worse (Table 4).

Table 4. Robustness comparison over synthetic networks. Each robustness value has been normalized by $10e+02$.

ID	R_G	$R_{G-\{e\}}^*$	$R_{G-\{e\}}^{S1}$	$R_{G-\{e\}}^{S2}$	$R_{G-\{e\}}^{S3}$	$R_{G-\{e\}}^{RobLPGA}$
ER	2.031	2.046 [36 119]	2.045 [9 36]	2.045 [9 36]	2.045 [36 108]	2.046 [36 119]
WS	3.672	3.727 [101 126]	3.702 [31 34]	3.7081 [126 128]	3.727 [101 126]	3.7088 [6 42]
BA	7.781	7.9347 [11 122]	7.854 [4 15]	7.905 [89 112]	7.9341 [63 116]	7.9347 [11 122]

6 Conclusion

This paper showed that GAs are effective in finding the link to protect when dealing with the optimization of network robustness in terms of effective graph resistance. By using specialized crossover and mutation operators, and a particular representation for the individuals of the population, the GA is able to find a link that would result in the maximum effective graph resistance, if removed, thus suggesting to protect the chosen link. Experiments on real-world networks showed that the method always finds solutions having the same value of the objective function obtained by the exhaustive search, which is also an upper-bound for the other heuristic strategies evaluated in this work. On synthetic networks, on Erdős-Rényi and Bárabasi-Albert networks in particular, our method again equals the exhaustive search finding coinciding solutions. Finally, over Watts-Strogatz networks, the method outperforms two of the three heuristics. Future work will study how optimizing robustness when the network is subject to different and multiple perturbations.

References

1. Abbas, W., Egerstedt, M.: Robust graph topologies for networked systems. In: 3rd IFAC Workshop on Distributed Estimation and Control in Networked Systems, pp. 85–90 (2012)
2. Buesser, P., Daolio, F., Tomassini, M.: Optimizing the robustness of scale-free networks with simulated annealing. In: International Conference on Adaptive and Natural Computing Algorithms, pp. 167–176. Springer (2011)
3. Ellens, W., Spiels, F., Mieghem, P.V., Jamakovic, A., Kooij, R.: Effective graph resistance. *Linear Alg. Appl.* **435**(10), 2491–2506 (2011)
4. Fiedler, M.: Algebraic connectivity of graphs. *Czechoslovak Math. J.* **23**(2), 298–305 (1973)
5. Frank, H., Frisch, I.: Analysis and design of survivable networks. *IEEE Trans. Commun. Technol.* **8**(5), 501–519 (1970)
6. Ghosh, A., Boyd, S., Saberi, A.: Minimizing effective resistance of a graph. *SIAM Rev.* **50**(1), 37–66 (2008)
7. Goldberg, D.E.: *Genetic Algorithms in Search. Optimization and Machine Learning*. Addison-Wesley Longman Publishing Co., Inc, Boston, MA, USA (1989)
8. Herrmann, H.J., Schneider, C.M., Moreira, A.A., Andrade Jr., J.S., Havlin, S.: Onion-like network topology enhances robustness against malicious attacks. *J. Stat. Mech. Theory Exp.* **2011**(01), P01,027 (2011)
9. Mieghem, P.V., Doerr, C., Wang, H., Hernandez, J.M., Hutchison, D., Karaliopoulos, M., Kooijt, R.: A framework for computing topological network robustness. Delft University of Technology, Report 20101218 (2010)
10. Ranjan, G., Zhang, Z., Boley, D.: Incremental computation of pseudo-inverse of Laplacian. In: Combinatorial Optimization and Applications. COCOA, pp. 730–749. Springer International Publishing, Switzerland (2014)
11. Wang, S., Liu, J.: Enhancing the robustness of complex networks against edge-based-attack cascading failures. In: 2017 IEEE Congress on Evolutionary Computation (CEC), pp. 23–28. IEEE (2017)

12. Wang, X., Pournaras, E., Kooij, R.E., Van Mieghem, P.: Improving robustness of complex networks via the effective graph resistance. *Europ. Phys. J. B* **87**(9), 221 (2014)
13. Wu, J., Barahona, M., Tan, Y.J., Deng, H.Z.: Spectral measure of structural robustness in complex networks. *Trans. Sys. Man Cyber. Part A* **41**(6), 1244–1252 (2011)
14. Zhou, M., Liu, J.: A memetic algorithm for enhancing the robustness of scale-free networks against malicious attacks. *Phys. A Stat. Mech. Appl.* **410**, 131–143 (2014)



Numerical Assessment of the Percolation Threshold Using Complement Networks

Giacomo Rapisardi¹(✉), Guido Caldarelli^{1,2}, and Giulio Cimini^{1,2}

¹ IMT School for Advanced Studies, 55100 Lucca, Italy
giacomo.rapisardi@imtlucca.it

² Istituto dei Sistemi Complessi (ISC)-CNR, 00185 Rome, Italy

Abstract. Models of percolation processes on networks currently assume locally tree-like structures at low densities, and are derived exactly only in the thermodynamic limit. Finite size effects and the presence of short loops in real systems however cause a deviation between the empirical percolation threshold p_c and its model-predicted value π_c . Here we show the existence of an empirical linear relation between p_c and π_c across a large number of real and model networks. Such a putatively universal relation can then be used to correct the estimated value of π_c . We further show how to obtain a more precise relation using the concept of the complement graph, by investigating on the connection between the percolation threshold of a network, p_c , and that of its complement, \bar{p}_c .

Keywords: Percolation theory · Complement graphs

1 Introduction

Percolation theory is a widely used concept in statistical physics [1], in particular in the field of complex networks to study critical phenomena, resilience and spreading processes [2–4]. However, percolation properties in network models (be they sparse treelike graphs [5] or clustered networks [6, 7]) are often considerably different from those of real-world networks—which feature a highly more complex topology. Recently, percolation has been reformulated as a message passing process which takes as input the detailed topology of a given network to predict percolation-related observables [8, 9], and which implies that the bond percolation threshold π_c of the network is bounded from below by the leading eigenvalue of its non-backtracking matrix [10]. This approach has been then generalized to clustered networks [11], in order to go beyond the locally treelike approximation which is not reliable for networks with high density of edges and short loops [12]. However, the method comes at a price of much higher computational complexity, and is not particularly satisfactory for bond percolation processes. Another important aspect of the message passing approach is that it describes network percolation in the thermodynamic limit, and as such cannot be precisely applied to finite graphs [13]. Indeed, the very percolation transition is ill defined in finite systems.

2 Numerical Methods and Results

Numerical simulations of the percolation process obtain the value of the percolation threshold p_c using Monte Carlo techniques [14]. Given Q independent realizations of the process at fixed percolation probability p , and the relative size of the largest cluster in the network $s_q(p)$, $q = 1, \dots, Q$ in the q -th realization, the percolation strength at p is estimated as $S(p) = \frac{1}{Q} \sum_q s_q(p)$, and the susceptibility as $\chi(p) = \frac{1}{Q \cdot S(p)} \sum_q [s_q(p)]^2 - S(p)$. The best estimate of the percolation threshold is then the value of p at which the susceptibility is maximal. As the simulated system is finite, such defined pseudo-critical threshold $p_c(N)$ decays towards the percolation threshold as $p_c(N) - p_c \sim N^{-1/\nu}$, where N is the (finite) size of the network [15].

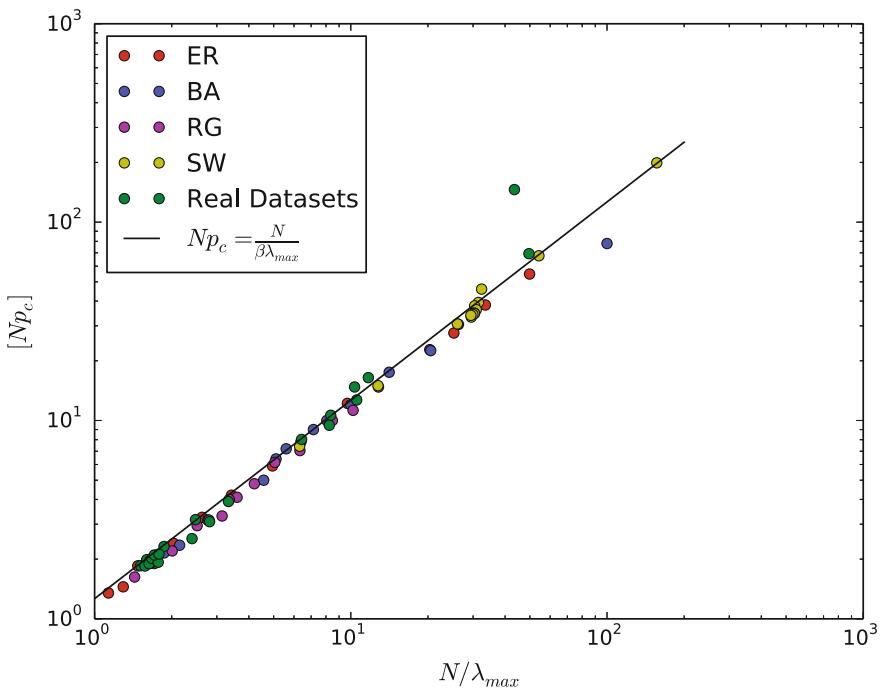


Fig. 1. Plot of Np_c versus $N\pi_c = N/\lambda_{\max}$, where λ_{\max} is the leading eigenvalue of the non-backtracking matrix, for several model and empirical networks. Note that accounting for the factor N allows to compare networks of different size. The black solid line is the linear fit $p_c = \pi_c/\beta$.

Figure 1 shows, for the bond percolation problem, the relation between the value p_c obtained in numerical simulations and the theoretical π_c given by the inverse of the leading eigenvalue of the non-backtracking matrix (λ_{\max}). The plot is obtained by considering a total of 79 networks of different sizes N (varying

approximately from 20 to 890), 23 of which are empirical while the remaining 56 are artificially generated according to four different random network models: Erdős-Rényi (ER), Regular (RG), Barabási-Albert (BA) and Watts-Strogatz (SW) [16]. Points are well fitted by a linear relation $p_c = \pi_c/\beta$ with $\chi^2/\nu = 4.34$, where however the value of $\beta = 0.791 \pm 0.019$ is different from unity: numerical and theoretical percolation thresholds do not coincide, yet their ratio appears to be constant across a variety of empirical and model networks of different size. While assessing the general validity of such an empirical evidence needs further statistical analysis, this relation can be quite valuable for correcting the theoretical value of π_c for finite, non-treelike networks.

In this work we explore the possibility to improve such an empirical relation using the concept of *complement graph*. The complement of a graph G is the graph \bar{G} with the same vertex set, but whose edges are those which are not present in G [17, 18]. The union graph of G and \bar{G} is therefore a complete graph. Complement graphs are found since long in the mathematic literature, for instance to address the graph coloring problem [19], to develop graph compression schemes [20] and search algorithms [21], to study network synchronizability [22], to assess graph hyperbolicity [23] and domination numbers [24]. The common approach of these studies is to prove rigorous results for graphs with a

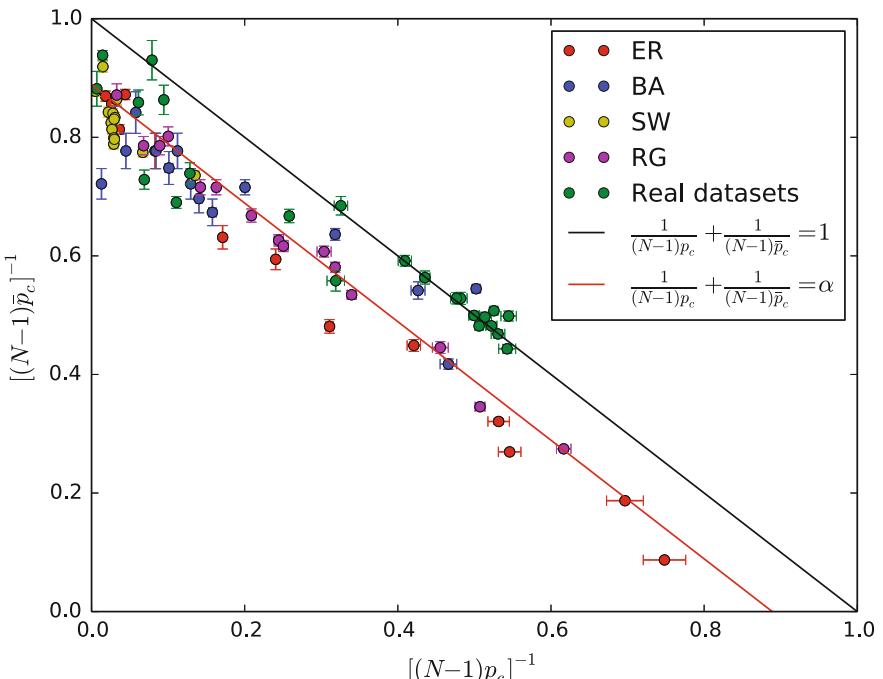


Fig. 2. Plot of p_c^{-1} versus \bar{p}_c^{-1} for various networks of different sizes. The solid black line is Eq. (1), while the solid red line is the linear fit of data.

small number of vertices [25–27]. Here, for the first time to our knowledge, we use complement graphs in the context of percolation on large-scale complex networks. In particular, we investigate on the existence of a complement relation for the percolation threshold p_c of a given graph G and the complement percolation threshold \bar{p}_c of \bar{G} .

Now, since the complement of a sparse network is dense, in the thermodynamic limit the percolation threshold of \bar{G} converges to the inverse of the leading eigenvalue of the adjacency matrix of \bar{G} [28]. In the simple case of ER networks, for $N \rightarrow \infty$ it is $p_c \simeq 1/\langle k \rangle = 1/[(N-1)f]$ (where f is the probability of existence of an edge), and thus the following relation should hold:

$$\frac{1}{(N-1)p_c} + \frac{1}{(N-1)\bar{p}_c} \simeq 1 \quad (1)$$

(an analogous complement relation of the two critical points also holds for regular graphs). As Fig. 2 shows, Eq. (1) slightly overestimates the relation between p_c and \bar{p}_c , as they do not add up to unity. In particular, the theoretical curve seems to constitute a boundary in the (p_c, \bar{p}_c) plane, and data are better fitted by a shifted linear relation

$$\frac{1}{(N-1)p_c} + \frac{1}{(N-1)\bar{p}_c} = \alpha < 1, \quad (2)$$

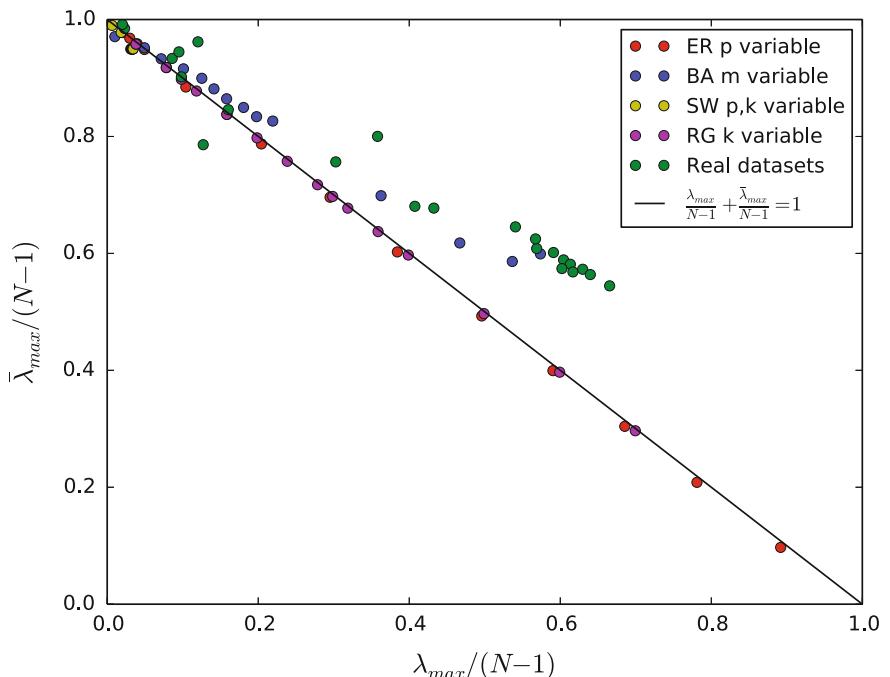


Fig. 3. Plot of λ_{\max} versus $\bar{\lambda}_{\max}$ for various networks of different sizes.

with $\alpha = 0.889 \pm 0.008$ and $\chi^2/\nu = 3.68$. The same behavior is observed in Fig. 3 for theoretical values of the percolation threshold, obtained as the leading eigenvalue of the non-backtracking matrices.

Building on the analysis of Fig. 1, we now study the relation

$$p_c + \bar{p}_c = \frac{1}{\beta'} \left(\frac{1}{\lambda_{\max}} + \frac{1}{\bar{\lambda}_{\max}} \right). \quad (3)$$

As shown in Fig. 4, Eqn. (3) fits the data quite well, and much better than the fit of Fig. 1. From the fit we obtained $\beta' = 0.856 \pm 0.010$ and $\chi^2/\nu = 1.16$. This factor can therefore be used to improve the estimate of the percolation threshold on finite, non treelike networks.

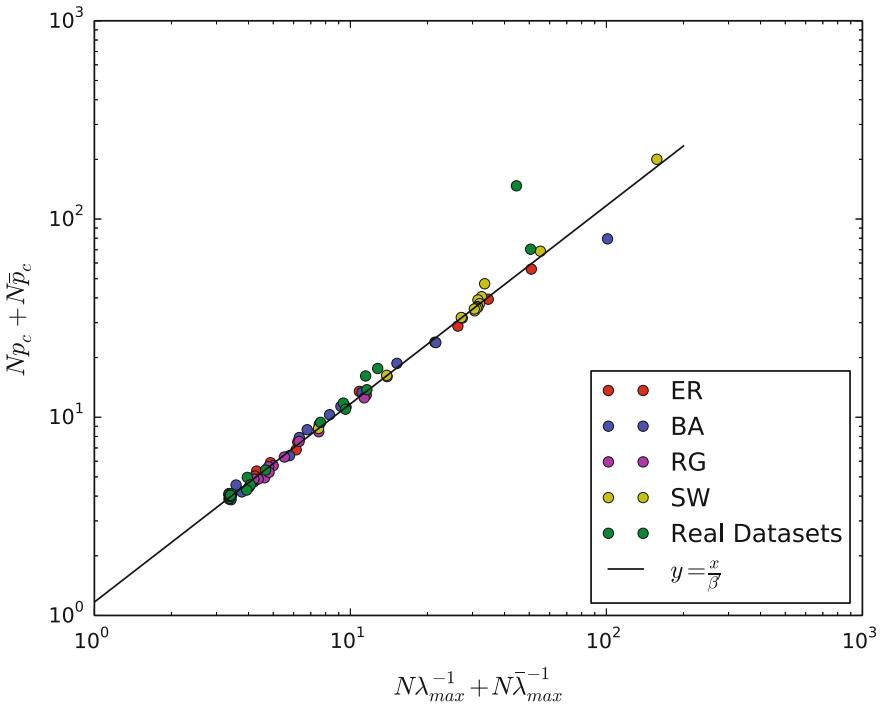


Fig. 4. Plot of $Np_c + N\bar{p}_c$ vs $\left(\frac{N}{\lambda_{\max}} + \frac{N}{\bar{\lambda}_{\max}}\right)$, where λ_{\max} is the leading eigenvalue of the non-backtracking matrix, for several model and empirical networks. The black solid line is the linear fit of Eq. (3).

To show that this is the case, in Fig. 5 we compare different estimates of the numerical percolation threshold, obtained as either the leading eigenvalues of the adjacency matrix λ_{\max}^A or of the non-backtracking matrix λ_{\max}^{NB} , eventually corrected by the β' factor. We indeed see that β' can be used to improve, on average, the approximation given by theoretical models.

We believe that the corrective factor is related to finite size effects and non treelike structures, however this hypothesis needs further investigation. Overall, while our approach is just at infant stage and our findings are only preliminary, they may have important concrete applications.

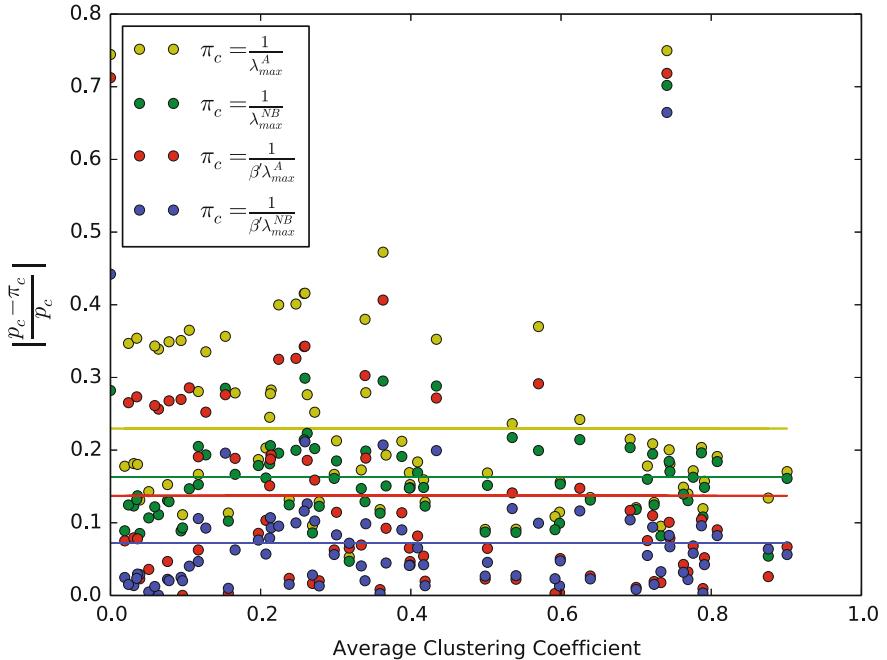


Fig. 5. Relative error associated to different estimates for the percolation threshold. Solid lines indicates average values of dots of the same color.

Acknowledgments. This work was supported by the EU projects CoeGSS (grant no. 676547) and SoBigData (grant no. 654024).

Appendix: Definition and Properties of the Complement Network

Formally, let a_{ij} be the generic element of the adjacency matrix \mathbf{A} associated with a given binary undirected graph G of N vertices, such that $a_{ij} = 1$ if an edge between vertices i and j exists, and $a_{ij} = 0$ otherwise. The adjacency matrix of the complement graph \bar{G} is defined through $\bar{a}_{ij} = 1 - \delta_{ij} - a_{ij}$, where δ_{ij} is the Kronecker delta which excludes self loops from \bar{G} , and $1 - \delta_{ij}$ defines the adjacency matrix of the complete graph. It follows trivially that $M + \bar{M} = \binom{N}{2}$, $\rho + \bar{\rho} = 1$ and $k_i + \bar{k}_i = N - 1 \forall i$, where M , ρ and k_i denote the number of

edges, the edge density, and the degree of (number of edges incident with) generic vertex i , respectively. Thus, given the degree distribution $P(k)$, the distribution of the complement degree is obtained as $\bar{P}(\bar{k}) = P(N - 1 - \bar{k})$, i.e., as the reflection of $P(k)$ on the $\frac{N-1}{2}$ vertical axis. Notably, the degree distribution of both a regular graph and an Erdős-Rényi graph (ER) are invariant under this transformation: the complement of a regular graph is a regular graph, as the complement of an ER is an ER. In particular, the complement of an ER with connection probability f is an ER with connection probability $1 - f$.

Moving to higher-order properties, the number of triangles (closed loop of length 3) of a graph and of its complement is

$$\Sigma_{\Delta} = \frac{\text{Tr}\mathbf{A}^3 + \text{Tr}\bar{\mathbf{A}}^3}{6} = \binom{N}{3} - \frac{1}{2} \sum_i k_i \bar{k}_i. \quad (4)$$

As such, both cases $k_i = 0$ and $\bar{k}_i = 0 \forall i$ (empty and complete graph) lead to $\Sigma_{\Delta} = \binom{N}{3}$ as expected. As for transitivity, a complementarity relation can be written also for the local clustering coefficient $c_i = \frac{\sum_{j \neq i} \sum_{k \neq i,j} a_{ij} a_{jk} a_{ik}}{k_i(k_i-1)}$:

$$c_i k_i (k_i - 1) + \bar{c}_i \bar{k}_i (\bar{k}_i - 1) = k_i^{nn} k_i + \bar{k}_i^{nn} \bar{k}_i - k_i - \bar{k}_i - k_i \bar{k}_i, \quad (5)$$

where $k_i^{nn} = \frac{\sum_{j \neq i} a_{ij} k_j}{k_i}$ is the average nearest-neighbors degree.

References

1. Stauffer, D., Aharony, A.: Introduction to Percolation Theory. Taylor and Francis (1994)
2. Callaway, D.S., Newman, M.E.J., Strogatz, S.H., Watts, D.J.: Network robustness and fragility: Percolation on random graphs. *Phys. Rev. Lett.* **85**, 5468 (2000)
3. Newman, M.E.J.: Spread of epidemic disease on networks. *Phys. Rev. E* **66**, 016128 (2002)
4. Dorogovtsev, S.N., Goltsev, A.V., Mendes, J.F.F.: Critical phenomena in complex networks. *Rev. Modern Phys.* **80**, 1275 (2008)
5. Cohen, R., ben Avraham, D., Havlin, S.: Percolation critical exponents in scale-free networks. *Phys. Rev. E* **66**, 036113 (2002)
6. Serrano, M.A., Boguná, M.: Stability diagram of a few-electron triple dot. *Phys. Rev. Lett.* **97**, 088701 (2006)
7. Newman, M.E.J.: Random graphs with clustering. *Phys. Rev. Lett.* **103**, 058701 (2009)
8. Karrer, B., Newman, M.E.J., Zdeborová, L.: Percolation on sparse networks. *Phys. Rev. Lett.* **113**, 208702 (2014)
9. Hamilton, K.E., Pryadko, L.P.: Tight lower bound for percolation threshold on an infinite graph. *Phys. Rev. Lett.* **113**, 208701 (2014)
10. Hashimoto, K., Namikawa, Y.: Automorphic Forms and Geometry of Arithmetic Varieties. Advanced Studies in Pure Mathematics. Elsevier Science (2014)
11. Radicchi, F., Castellano, C.: Beyond the locally treelike approximation for percolation on real networks. *Phys. Rev. E* **93**, 030302 (2016)

12. Radicchi, F.: Predicting percolation thresholds in networks. *Phys. Rev. E* **91**, 010801 (2015)
13. Timar, G., da Costa, R.A., Dorogovtsev, S.N., Mendes, J.F.F.: Nonbacktracking expansion of finite graphs. *Phys. Rev. E* **95**, 042322 (2017)
14. Newman, M.E.J., Ziff, R.M.: Efficient Monte Carlo algorithm and high-precision results for percolation. *Phys. Rev. Lett.* **85**, 4104 (2000)
15. Radicchi, F., Castellano, C.: Breaking of the site-bond percolation universality in networks. *Nature Commun.* **6**, 10196 (2015)
16. Newman, M.E.J.: The structure and function of complex networks. *SIAM Rev.* **45**, 167 (2003)
17. Clark, L., Entringer, R.: Smallest maximally nonhamiltonian graphs. *Periodica Math. Hung.* **14**, 57 (1983)
18. Gross, J., Yellen, J.: *Graph Theory and Its Applications. Discrete Mathematics and Its Applications*. Taylor and Francis (1998)
19. Nordhaus, E.A., Gaddum, J.W.: A Kaleidoscopic view of graph colorings. *Am. Math. Mon.* **63**, 175 (1956)
20. Kao, M.Y., Occhiogrosso, N., Teng, S.-H.: Simple and efficient graph compression schemes for dense and complement graphs. *J. Comb. Optim.* **2**, 351 (1998)
21. Ito, H., Yokoyama, M.: Linear time algorithms for graph search and connectivity determination on complement graphs. *Inf. Proc. Lett.* **66**, 209 (1998)
22. Duan, Z., Liu, C., Chen, G.: Network synchronizability analysis: the theory of subgraphs and complementary graphs. *Phys. D Nonlinear Phenom.* **237**, 1006 (2008)
23. Bermudo, S., Rodrguez, J.M., Sigarreta, J.M., Tours, E.: Hyperbolicity and complement of graphs. *Appl. Math. Lett.* **24**, 1882 (2011)
24. Haas, R., Wexler, T.B.: Bounds on the signed domination number of a graph. *Discret. Math.* **283**, 87 (2004)
25. Akiyama, J., Harary, F.: A graph and its complement with specified properties. *Int. J. Math. Math. Sci.* **2**, 223 (1979)
26. Xu, S.: Some parameter of graph and its component. *Discret. Math.* **65**, 197 (1987)
27. Petrovic, M., Radosavljevic, Z., Simic, S.: A graph and its complement with specified spectral properties. *Linear Multilinear Algebra* **51**, 405 (2003)
28. Bollobas, B., Borgs, C., Chayes, J., Riordan, O.: Percolation on dense graph sequences. *Ann. Probab.* **38**, 150 (2010)



Optimal Control Rules for Random Boolean Networks

Matthew R. Karlsen^(✉) and Sotiris K. Moschouyannis^(✉)

Faculty of Engineering and Physical Sciences, Department of Computer Science,
University of Surrey, Guildford, Surrey GU2 7XH, UK
[{matthew.r.karlsen,s.moschouyannis}surrey.ac.uk](mailto:{matthew.r.karlsen,s.moschouyannis@surrey.ac.uk})

Abstract. A random Boolean network (RBN) may be controlled through the use of a learning classifier system (LCS) – an eXtended Classifier System (XCS) can evolve a rule set that directs an RBN from any state to a target state. However, the rules evolved may not be optimal, in terms of minimising the total cost of the paths used to direct the network from any state to a specified attractor. Here we uncover the optimal set of control rules via an exhaustive algorithm. The performance of an LCS (XCS) on the RBN control problem is assessed in light of the newly uncovered optimal rule set.

1 Introduction

Controlling complex networks is key in areas as diverse as biological systems and socio-technical systems such as transport networks. Perturbations may cause the network to spontaneously go to a state that is less desirable than others, e.g., perturbations in metabolic networks may indirectly lead to non-viable strains [3]. Recent advances include work on network structure *control nodes* [1, 15, 16], its reconfiguration [8, 17, 18], but also on the network dynamics [3, 9] with application for example to transport [10]. ‘*Controllability*’, is here measured by the extent to which we have the ability to direct the network from any (possibly ‘bad’) state to an *attractor* (possibly a ‘good’ state or states, where the system continues to perform its functions). Herein the network is said to be ‘*controlled*’ when the rule set takes the network from any state to the target attractor. We focus here on developing sets of ‘control rules’ able to achieve control for a given target attractor.

In previous work [9], we have applied rule-based machine learning in the form of an “eXtended Classifier System” (XCS) [20, 21] to the problem of controlling random Boolean networks (RBNs) [11, 12]. RBNs have been used to model biological networks such as gene regulatory networks, e.g., see [7, 13]. We showed

This research was partly funded by the Department for Transport, via Innovate UK and the AIR round 4 programme, under the *Onward Journey Planning Assistant (OJPA)* project and partly funded by EIT Digital under the *Real-Time Flow* project, activity 18387–SGA2018.

that XCS can evolve a rule set that takes an RBN from any state to an attractor. However, no indication of the ‘ideal’ or ‘optimal’ set of rules was supplied.

In this paper, we provide an algorithm that derives an optimal set of control rules for a given network and a given intervention cost, which is represented by a weight greater than one for intervention links (contrasted to a weight of ‘1’ for a ‘natural step’). This enables a comparison between the XCS rules evolved to control the network and the ‘ideal’ or ‘optimal’ set of rules that does the job. The idea for taking a cost-based approach rather than restricting the operations available to XCS can be attributed to Fornasini and Valcher [5] (though they do not work with XCS).

The remainder of this paper is structured as follows. Section 2 briefly describes RBNs whilst Sect. 3 outlines key principles behind LCSs. In Sect. 4 we describe the central algorithm of the paper, which computes the optimal set of control rules for an RBN. Section 5 explains the experiments conducted to compare this algorithm with an XCS-based approach. Results are presented in Sect. 6, and discussed in Sect. 7. Conclusions and future work follow in Sect. 8.

2 Random Boolean Networks

Random Boolean Networks [11, 12] are networks with Boolean values and an update function at each node, where the network structure and/or the update functions are specified at random. Here we focus on the NK Boolean Network [11, 12] with N nodes and K inputs per node. In this model the origin node of each input link is randomly drawn from the set of N nodes such that each of the N nodes is affected by K nodes in the same set (a node may have an input from itself). Each node holds a Boolean variable, initialised randomly with the value 0 or 1. The update functions for each node are randomly determined Boolean functions such that each unique combination of input values supplied by the input links (0, 0; 0, 1; 1, 0; or 1, 1; when $K = 2$) updates the node to a new value (either 0 or 1), overwriting the previous value. An example RBN is shown in Fig. 1 (enclosed within the box).

Random Boolean networks may be asynchronous or synchronous. In an asynchronous network one node is randomly selected to update first, its inputs are read, and a new value is calculated and set at that node. The new value then propagates down any outward links attached to that node, updating further nodes, and so on, until a stable state or state cycle is reached. In a synchronous RBN all nodes are updated simultaneously. Given the current state of the network – the Boolean value at each node – the new Boolean values are calculated without being set. Once all the calculations are complete, a new Boolean value is set at each node. Herein we consider networks that update synchronously.

Random Boolean networks can be controlled via timed bit flips at available targeted nodes (i.e. if the current Boolean value at the node is a 0 it can be flipped to a 1 or vice versa). This can be represented as an integer (starting at 1) indicating which node to bit flip. The state of the network can be shown as a bit string consisting of the state of each node in the network in a fixed order,

see Fig. 1. Thus if you have, for instance, 5 nodes then the state may be 00101 and the action could be 4 (shown as 00101 : 4). In this instance flipping the 4th bit from a 0 to a 1 would result in the state 00111.

Substantial previous work on the control of RBNs is covered in a review by Fornasini and Valcher [6]. This work differs in relation to previous works via (1) taking a programmatic approach rather than an algebraic approach and (2) providing the control instructions as a condensed set of ‘control rules’.

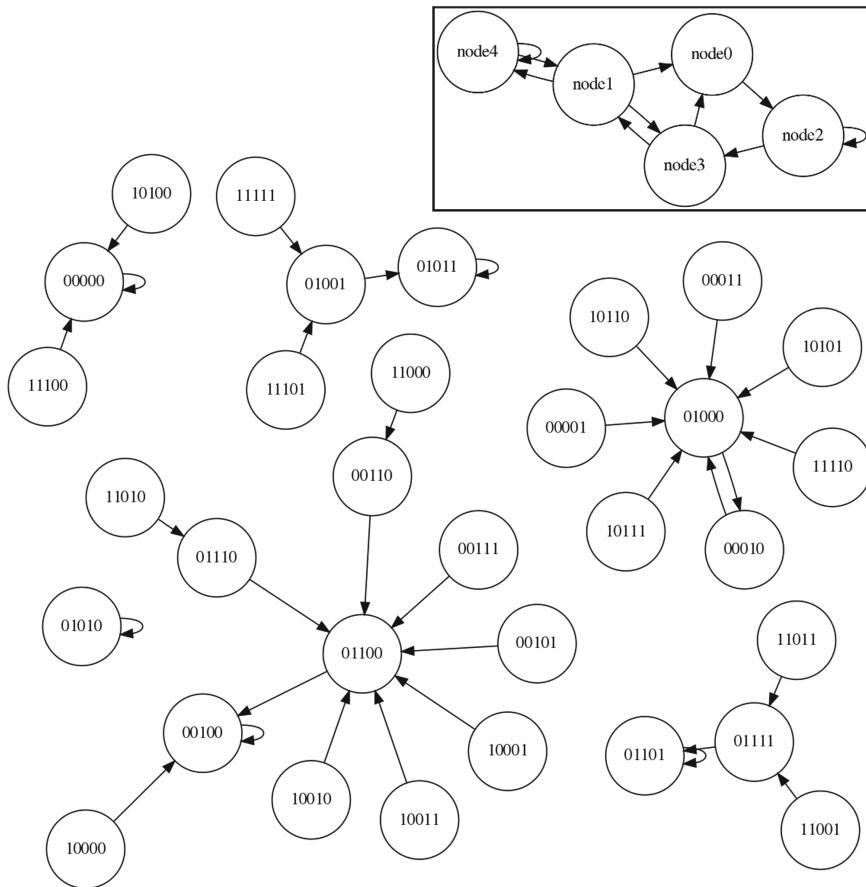


Fig. 1. An RBN with $N = 5$, $K = 2$ (enclosed in box). The resulting state space is also shown.

3 Learning Classifier Systems

A learning classifier system (LCS) [19] is a rule-based machine learning technique comprising a population of rules, a reinforcement (or supervised) learning mechanism and a genetic algorithm. Further additional components include a number of filters, a ‘covering mechanism’ (to generate new rules when needed), a prediction array (to assess the quality of proposed actions) and an action selector (to choose which of the potential actions actually gets implemented in the environment).

At the heart of an LCS is a number of IF <condition> THEN <action> rules in a *population*. The rules in this instance are represented as follows: 10100 : 3 (a bit string condition followed by a colon separator, followed by an integer representing an action). The condition consists of 0s, 1s and # symbols (where the # wildcard indicates ‘don’t care’). The environment state (represented as a bit string) matches a rule’s condition when at each index the symbol either matches or the condition is #. For example, 11101 will match 111#1 : 5. ##### : 1 will match *any* environment, taking action 1.

The overall system for the XCS [20, 21] variant can be found in [9] (see Fig. 5 in [9]). The implementation used is that described in the algorithmic specification provided by Butz and Wilson [2], with a post-run rule set compression based on [22].

The overall steps of the algorithm are as follows:

1. The RBN state is read in as a bit string (e.g. 10100)
2. The subset of rules that match the RBN state are selected from the rule population
3. The size of this ‘match set’ is considered. While match set too small:
 - (a) Generate a new rule
 - (b) Add the rule to the population
 - (c) Regenerate the match set
4. Create a prediction array (contains fitness-weighted payoffs for each match set action)
5. Select an action (random with a certain % probability, or else best action picked from action set)
6. Effect action in the environment (i.e. alter one state bit within the RBN)
7. Derive the action set from the match set (the subset that suggest the effected action)
8. Apply the genetic algorithm to the action set, adding new rules to the population
9. Repeat the above steps.

4 Optimal Control Rules

An overview of the algorithm for producing optimal control rules is as follows:

Construct initial rules

1. Construct the original state space graph of the RBN (i.e., with no interventions)
2. Assign each original link a weight of 1
3. For each state-space node, create one ‘intervention’ link to each nearest neighbour with an intervention-level weight (2.0, 3.0, etc).
4. For each state-space node:
 - (a) run the Dijkstra shortest path algorithm to each attractor node in the specified attractor
 - (b) store the shortest path for the origin node
 - (c) calculate total cost for the path (link weight sum) and store
 - (d) calculate number of interventions (count of link with weight > 1) and store
 - (e) for each state node in the path:
 - i. create a new classifier with condition == current node and action as required (to point to the next step on the shortest path; note that the classifier contains no # symbols)
 - ii. if no classifier exists with the new classifier condition, add the new classifier to population
5. Calculate average statistics (average cost and average number of interventions)

Compress rules (*optional*)

6. Run rule merger:
 - (a) Group classifiers by action
 - (b) For each action group:
 - i. For each classifier in action group, for every other classifier in action group, check subsumption
 - ii. If classifier subsumes other classifier, delete classifier to be subsumed and increment numerosity of subsumer
 - (c) For each action group; while all combinations not explored without a merge:
 - i. iterate through all pair combinations of classifiers
 - ii. if ‘is adjacent or overlaps’ then merge the two rules.

We shall now explain some steps in greater detail. Construction of the original state space graph of the RBN is performed by initialising the network at each state of the network and then evolving the network to an attractor. For each path traced out in this way, the subset of the state space explored is recorded. When complete, this process yields the full state space of the network. Each link in the original state space network is given a weight of 1.

Once the original state space has been constructed all possible intervention links are added to the graph. These ‘nearest neighbour’ links are created as follows. For each state space node the neighbouring nodes are identified as those

that differ from the selected node by a single bit flip. Then, a directed weighted link is created from the originally selected node to each neighbouring node. The precise weight used depends upon the cost of an intervention.

Next, the Dijkstra shortest path algorithm is applied from every node to every node in the target attractor, using the GraphStream library [4]. The algorithm was selected because it is capable of considering weighted edges, unlike the A* algorithm (also in GraphStream). Once the Dijkstra shortest path has been worked out for each of the nodes in the network state space, for each node on each path we create a new classifier with a condition equal to the current node (with no wildcards) and the action required to alter the state of the network to the next state on the shortest path. For instance, if the current state is 11011 and we required a shift to 10011 and the only link connecting the two nodes was a link with weight > 1 then we work out which bit must be flipped (bit 2 in this instance) and then construct the rule 11011 : 2. In contrast, if the current state was 00110 and the next desired state was 00111 and there was an original (weight 1) link connecting the two states then the rule 00110 : 0 would be created where 0 is the ‘no operation’ action, indicating that no action needs to be taken to alter the state. Upon completion of the classifier, we add the new classifier to the population *only if* no classifier already exists with the same condition.

Note that it is possible for the shortest path algorithm to find more than one shortest path from a node to the attractor. For this reason, we only add a classifier if its condition is not already present in the population.

The rule merger (used for rule set compression, as outlined in the itemised steps above) first groups classifiers by action (this is a quick way of reducing combinations later in the process). Once grouped by action, we check subsumption for each possible pair combination of classifiers (see [9]). The main merging code is then initialised, using a while loop that exits when no further modifications are possible. This loop iteratively checks whether the classifier pairs have a matching action and matching condition components, *less one, which is adjacent or overlaps* (i.e. one rule has 0 and the other 1; or one or both have # at the specified location) – if this condition is met the rules are merged.

Post compression, further steps are performed to produce a state-space diagram, displaying both the ‘natural’ links (in green) and the interventions (in red). Note that the cost of interventions affects the resulting network.

5 Experiments

Here we use the aforementioned optimal rule discovery algorithm to create the optimal rule sets for 10 different networks. We then run XCS on each of these networks 25 times and produce aggregate performance results (necessary due to XCSI’s stochastic components). These results are then critically compared to the solutions produced via the optimal rule discovery algorithm. We evaluate 3 levels of intervention cost: 2.0, 2.5 and 3.0.

There are some changes to the way XCS is used in [9]. Rather than enforcing a mandatory ‘natural step’ after each intervention we introduce the weighting of

links such that intervention links are more ‘expensive’ than ‘natural step’ links. This necessitates changes to the reward function of the XCS implementation. Reward is given as 0 for an action that does not reach an attractor and 1000 as the reward for an action that reaches the attractor (even if said action is a ‘no-op’). Two interventions are therefore permitted in succession. However, because both incur the increased cost this is deemed acceptable. The XCS parameters used are shown in Table 1 (the brief descriptions are based on those in the original table in [9]). Due to the simplification of the more complex reward structure used in the previous paper, we find that the ‘default’ XCS parameters (as described by Butz and Wilson [2]) function suitably. We use these parameters for these initial results, with the advantage that the parameter combination used is not randomly generated by a parameter explorer and thus is more human-comprehensible.

A condensation technique [14] is used towards the end of the XCS run to reduce the number of rules present in the population. This is applied before the optional compression step as described in [9]. After 10,000 steps the XCS algorithm evaluates the solution and proceeds to the condensation phase. If a solution is not present, it continues to run and re-evaluates every 1000 steps. In the condensation phase θ_{mna} is set to 1, and the GA mutation and cross-over are disabled. The XCS algorithm then runs for 20,000 additional steps and evaluates the solution. If the solution is viable the program writes the output and halts. If the solution is not viable the condensation algorithm runs for a further 10,000 steps. It is technically possible that the algorithm will never halt and will require a restart. This is comparatively rare: on the 3004 runs conducted this occurred only 4 times.

6 Results

Table 2 presents a highly condensed summary of the results. Each unique combination of N, K and IC (intervention cost) represents 10 distinct networks of 25 runs using XCS and a single computation of optimal results using the algorithm described earlier in the paper for each distinct network. Each network has an optimal control cost associated with it. The ‘Opt. Min’ here shows the minimum optimal control costs from the 10 different control costs for the networks. The ‘Opt. Avg.’ shows the average of these 10 control costs. Finally, the ‘Opt. Max’ shows the maximum optimal control cost from the 10 different control costs for the networks. The XCS columns are equivalent to the optimal columns except that they apply to the 25 XCS runs per network rather than the application of the optimal rule algorithm (above).

Tables 3, 4 and 5 present the detailed results for the 10 N = 5, K = 2 networks (with intervention costs 2.0 and 3.0 respectively). The ‘Opt Cost’ is the average cost of navigating from any state to one of the target attractor states. To produce this measure the cost of navigation from every node in the network is summed (at a cost of 1.0 per ‘natural’ link and the specified cost per ‘intervention’ link) and then divided by the total number of states. ‘OC Rule Count’ is the number of rules required to achieve the Optimal Cost outcome. ‘XCS Avg. Cost’ is the

Table 1. Parameter settings and brief descriptions

Parameter	Value	Brief description
R	1400	Rule population size
γ	0.71	Discount rate
θ_{mna}	$N + 1$	Min. number of actions in match set
$P_{\#}$	0.33	Probability of hash
p_I	0.0	Initial payoff
ε_I	0.0	Initial error
F_I	0.0	Initial fitness
ε_0	10	Error threshold
θ_{ga}	5.0	Genetic algorithm frequency
θ_{del}	20.0	Deletion threshold
β	0.2	Affects update of p , ε , and action set size for classifiers
α	0.1	Affects fitness updates
ν	5.0	Affects fitness updates
χ	0.8	Likelihood of GA crossover operation
μ	0.05	Likelihood of GA mutation operation
δ	0.1	Mods. effect of fitness on the ‘deletion vote’ of a classifier
θ_{sub}	30.0	Subsumption threshold
p_{explr}	0.5	Likelihood of exploring
$doAsSubsumpt.$	true	Perform subsumption in the action set?
$doGaSubsumpt.$	true	Perform subsumption in the GA?

average value of all the average costs for each of the 25 runs on the network in question. The ‘XCS Rule Count’ is the average rule count required to achieve the XCS average cost. The ‘XCS time’ is the average time (in seconds) required for the 25 runs on a given network.

For each of the networks shown in Tables 3, 4 and 5, there is an associated optimal set of control rules and 25 distinct sets of control rules evolved by XCS. Whilst displaying them all is not feasible due to space constraints, an example set of optimal rules is shown in Table 6, along with two example sets of XCS-evolved rules. The square brackets next to each rule display the final predicted payoff and fitness of each rule, separated by a ‘/’. Note that the predicted payoff and fitness are related specifically to XCS and thus are not shown for the rules produced using the non-XCS algorithm.

Table 2. Control costs – optimal versus XCS

N	K	IC	Opt. Min.	Opt. Avg.	Opt. Max.	XCS Min.	XCS Avg.	XCS Max.
5	2	2.0	1.72	2.61	3.59	3.69	5.11	6.42
5	2	2.5	1.63	2.59	3.59	3.73	5.88	7.35
5	2	3.0	1.94	3.13	4.56	4.53	6.43	8.91
5	3	2.0	1.75	2.68	3.34	4.63	6.27	7.69
5	3	2.5	1.78	2.63	3.34	5.10	7.08	9.35
5	3	3.0	2.38	3.45	4.47	6.15	7.55	9.45
7	2	2.0	2.46	3.30	4.44	5.21	6.48	9.24
7	2	2.5	2.41	3.21	4.44	5.72	7.25	10.06
7	2	3.0	2.88	3.99	5.94	6.30	7.93	10.50
7	3	2.0	2.61	3.19	3.71	4.94	7.24	13.41
7	3	2.5	2.55	3.08	3.52	6.03	7.98	13.75
7	3	3.0	3.12	3.87	4.77	6.05	8.87	14.45

Table 3. Results for IC = 2.0 (N = 5, K = 2)

Nw num	Opt. cost	OC rule count	XCS avg cost	XCS rule count	XCS time
1	3.41	15	5.99	37.52	73.16
2	1.72	13	4.18	22.6	55.68
3	2.00	13	3.69	19.04	47.28
4	3.25	13	5.92	28.24	70
5	1.91	19	5.67	16.48	88.12
6	2.25	13	5.49	30.72	77.36
7	2.16	7	3.86	14.52	49.32
8	2.69	14	4.65	31.16	63.2
9	3.59	13	6.42	35.64	90.04
10	3.13	13	5.29	41.04	118.72

Table 4. Results for IC = 2.5 (N = 5, K = 2)

Nw num	Opt. cost	OC rule count	XCS avg cost	XCS rule count	XCS time
1	3.41	18	7.35	43.32	72.6
2	1.63	12	4.35	21.8	53.44
3	2.00	5	3.73	17.36	47.16
4	3.25	11	6.82	22.04	70
5	1.91	19	6.20	15.88	85.72
6	2.19	12	6.43	27.72	97.04
7	2.16	7	4.72	18.64	48.64
8	2.69	13	5.52	32.32	63.48
9	3.59	9	7.30	40.32	87.96
10	3.13	10	6.40	35.56	112.6

Table 5. Results for IC = 3.0 (N = 5, K = 2)

Nw num	Opt. cost	OC rule count	XCS avg cost	XCS rule count	XCS time
1	4.56	19	8.09	40.08	71.84
2	1.94	9	4.94	21.28	54.72
3	2.03	5	4.53	19.88	46.92
4	3.81	11	7.67	31.2	69.28
5	2.53	19	6.09	20.4	83.08
6	2.75	14	7.12	30.16	77.92
7	2.66	7	4.78	21.52	50.36
8	2.84	11	5.79	36.36	63.56
9	4.47	13	8.91	35.92	87.08
10	3.75	12	6.36	43.16	115.36

7 Discussion

As can be seen from Table 2 the average cost for the optimal set of rules ranges from 2.61 for an N = 5, K = 2 network with an intervention cost of 2 to 3.99 for an N = 7, K = 2 network with an intervention cost of 3. In contrast, the XCS average cost is 5.11 for an N = 5, K = 2, IC = 2.0 network, with a cost of 8.87 for an N = 7, K = 3, IC = 3.0 network. The min. optimums can be substantially below the average optimums (i.e. one or more particular network structure of the 10 randomly generated ones are much easier to control than average).

The above observation can be substantiated with Tables 3, 4, and 5 – substantial diversity of results can be seen across these tables. One of the major ways in which the complexity of the control problem varies is in the number

Table 6. Optimal versus two example rule sets for an N = 5, K = 2 network (IC = 2.0)

Optimal rule set	XCS example rule set 1	XCS example rule set 2
01000 : 2	#0### : 0 [826.14/0.9978]	0### : 0 [610.71/1]
00100 : 3	01000 : 2 [1330.26/1]	01000 : 2 [1277.48/1]
01010 : 4	##1#0 : 3 [1090.68/1]	00### : 2 [792.98/0.7836]
11101 : 5	0#000 : 1 [779.48/1]	0#010 : 4 [963.7/1]
11010 : 3	#0### : 2 [792.56/0.7614]	0#00# : 1 [645.64/0.9968]
00#11 : 0	0#000 : 4 [785.99/1]	###11 : 5 [669.3/1]
#1111 : 0	0#0#0 : 5 [677.82/1]	1#10# : 0 [1241.56/1]
100#1 : 0	01100 : 1 [1043.01/1]	0#000 : 4 [652.45/1]
#11#0 : 0	0#000 : 3 [735.1/1]	0#001 : 5 [933.07/1]
101## : 0	10000 : 1 [1111.64/1]	0##00 : 5 [621.72/1]
100#0 : 1	#0#010 : 3 [688.34/1]	0#01# : 3 [542.8/1]
#1011 : 2	1#000 : 3 [999.7/1]	0#000 : 3 [664.91/1]
1100# : 3	##100 : 4 [757.19/1]	1#000 : 3 [865.67/1]
00#10 : 4	0#010 : 4 [1329.22/0.6272]	10##0 : 1 [1096.53/0.9996]
0##01 : 5	#1#10 : 2 [811.6/1]	11##0 : 1 [732.65/0.9979]
	0#101 : 5 [747.06/1]	##1#0 : 3 [802.9/0.315]
	011#0 : 0 [751.29/1]	##010 : 5 [499.36/0.9975]
	11100 : 0 [1386.61/1]	###01 : 3 [577.9/1]
	##110 : 4 [788.72/1]	####1 : 1 [612.58/0.9038]
	010#0 : 0 [792.48/1]	1#0## : 0 [648.04/1]
	1#0## : 4 [551.88/0.8454]	####1 : 4 [538.15/0.4199]
	####1 : 3 [537.19/1]	1#110 : 0 [948.36/1]
	####1 : 1 [535.8/0.8495]	0#1## : 4 [545.14/0.9135]
	1##10 : 1 [758.33/1]	00010 : 1 [690.26/1]
	01##1 : 0 [534.25/1]	0#110 : 3 [724.95/0.9887]
	0##1# : 1 [552.97/0.7678]	####10 : 2 [685.01/0.978]
	0#100 : 5 [570.96/1]	0101# : 1 [547.31/0.9862]
	0#001 : 5 [996.12/0.1555]	1#000 : 2 [639.82/1]
	##11# : 5 [673.12/0.9958]	1#01# : 3 [656.55/1]
	0##011 : 5 [740.69/1]	01100 : 3 [987.87/0.9938]
	11##0 : 2 [722.05/0.8468]	0#101 : 5 [658.58/1]
	1#100 : 5 [618.54/1]	11111 : 0 [694.67/1]
	0#100 : 2 [753.52/0.9925]	1##10 : 4 [676.44/1]
	01010 : 4 [1032.16/0.9781]	####1 : 2 [567.87/0.743]
	1#100 : 1 [751.96/0.9846]	##1## : 2 [605.81/0.4683]
	#1#01 : 4 [510.13/0.8903]	1##01 : 5 [677.38/1]
	0##011 : 4 [746.69/0.9999]	0#110 : 5 [515.22/1]
	0010# : 1 [705.39/0.9299]	0#100 : 1 [787.68/1]
	110## : 0 [598.88/1]	##111 : 3 [536.47/1]
	01001 : 5 [984.98/0.9288]	1#00# : 4 [539.28/0.5748]
	1##1#1 : 5 [774.49/0.3823]	1#10# : 4 [646.66/0.8809]
	1#001 : 5 [719.66/1]	##110 : 1 [656.67/0.9729]
	##1#1 : 4 [561.8/0.8403]	10111 : 0 [942.59/1]
	111#1 : 0 [736.48/1]	

of attractors. The networks with a higher number of attractors tend to have a higher control cost.

Table 6 shows us the optimal rule set and two of the rule sets that were actually evolved for the network. We can see that the entire XCS procedure has produced rule sets that are about 2.5 times the size of the optimal rule set. We can also see that the XCS rule sets do not contain the optimal rule set as a subset of their rules. In terms of exact matches, the rule 01000 : 2 is found in all three sets whilst 01010 : 4 is found in one of the XCS rule sets. That said, whilst the number of exact matches is few, we can see a number of rules in the two XCS rule set examples are close to the rules in the optimal rule set (for instance there may be one too many # symbols – the classifiers may be too general).

Overall comparative results are presented in Figs. 2a and 2b. The x-axis displays the average cost of the XCS intervention strategies for each network as a percentage of the optimal cost, whilst the y-axis shows the average rule set size for the XCS intervention strategies as a percentage of the number of rules required to achieve the optimal strategy.

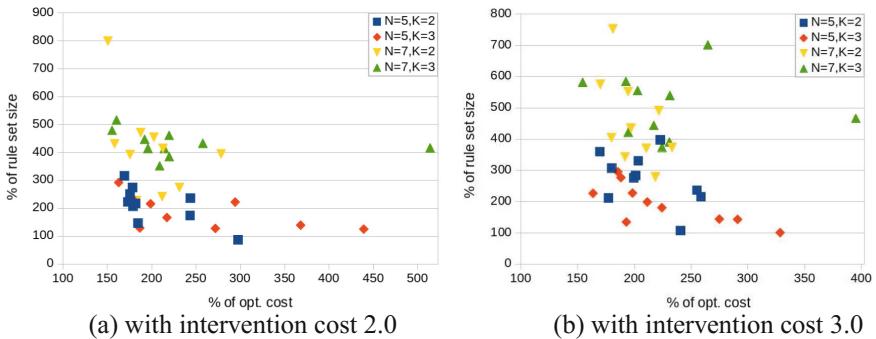


Fig. 2. XCS relative performance

In Fig. 2a we can see that for the intervention cost of 2.0 the XCS solutions range between approximately 150 and 300% of optimal cost (excluding 3 outliers) whilst rule set size is between approximately 120 and 510% (excluding one outlier).

With cost 3.0 performance is more varied across the various network configurations, as shown in Fig. 2b. The cost of the XCS rule sets falls between approximately 150 and 300% of optimal (excluding two outliers) whilst the rule set size falls between 100 and 600% of optimal (excluding two outliers).

Finally, it should be noted that the different costs may bring about substantially different control graphs – when intervention cost is 3.0 the number of intervention links is notably lower than when intervention cost is 2.0. Due to space restrictions we do not include the control graphs here as in [9].

8 Conclusions and Future Work

We presented an algorithm for uncovering the optimal set of control rules for random Boolean networks. We compared the performance of this optimal rule calculator algorithm and the XCS variant of learning classifier systems when applied to these networks. We find that, whilst XCS presents a viable means of evolving control rules for Boolean networks, the currently produced rules are not optimal in terms of cost or rule set size. Conversely, the optimal rule calculator algorithm produces very strong performance in controlling relatively small Boolean networks. Further work is required to narrow the gap between the XCS solution and the optimal solution.

There are a number of directions for future investigation.

Firstly, the technique is currently only applicable to small networks. However, the ‘compressed’ nature of the final rule set suggests that we may be able to develop a more general technique applicable to larger networks.

Secondly, XCS could be modified to increase performance, via parameter adjustments or structural modification of XCS itself (incrementally or via replacement of XCS with another classifier system variant).

Thirdly, improvement of the condensation or compression algorithm could also be possible – variants exist (see [14]).

Finally, the different network structures could be investigated further. This could be in the form of larger NK Boolean networks or networks with higher values of K, or network structures produced with different generators entirely, such as Erdős–Rényi model-based graphs or the Barabási–Albert model.

References

1. Bianconi, G., Pin, P., Marsili, M.: Assessing the relevance of node features for network structure. *Proc. Natl. Acad. Sci.* **106**(28), 11433–11438 (2009). <https://doi.org/10.1073/pnas.0811511106>. <http://www.pnas.org/content/106/28/11433>
2. Butz, M.V., Wilson, S.W.: An algorithmic description of XCS. In: International Workshop on Learning Classifier Systems, pp. 253–272. Springer (2000)
3. Cornelius, S.P., Kath, W.L., Motter, A.E.: Realistic control of network dynamics. *Nat. Commun.* **4**, 1942 (2013)
4. Dutot, A., Guinand, F., Olivier, D., Pigné, Y.: Graphstream: A tool for bridging the gap between complex systems and dynamic graphs. In: Emergent Properties in Natural and Artificial Complex Systems. 4th European Conference on Complex Systems (ECCS'2007) (2007)
5. Fornasini, E., Valcher, M.E.: Optimal control of Boolean control networks. *IEEE Trans. Autom. Control* **59**(5), 1258–1270 (2014)
6. Fornasini, E., Valcher, M.E.: Recent developments in Boolean networks control. *J. Control Decis.* **3**(1), 1–18 (2016)
7. Gates, A.J., Rocha, L.M.: Control of complex networks requires both structure and dynamics. *Sci. Rep.* **6**, 24456 (2016)
8. Haghghi, R., Namazi, H.: Algorithm for identifying minimum driver nodes based on structural controllability. *Math. Probl. Eng.* **2015** (2015)

9. Karlsen, M.R., Moschoyiannis, S.: Evolution of control with learning classifier systems. *Appl. Netw. Sci.* **3**(1), 30 (2018). <https://doi.org/10.1007/s41109-018-0088-x>
10. Karlsen, M.R., Moschoyiannis, S.: Learning condition–action rules for personalised journey recommendations. In: RuleML+RR: Rules and Reasoning. LNCS, vol. 11092, pp. 293–301 (2018)
11. Kauffman, S.: The Origins of Order. Oxford University Press, New York, NY (1993)
12. Kauffman, S.A.: Metabolic stability and epigenesis in randomly constructed genetic nets. *J. Theor. Biol.* **22**(3), 437–467 (1969)
13. Kim, J., Park, S.M., Cho, K.H.: Discovery of a kernel for controlling biomolecular regulatory networks. *Sci. Rep.* **3**, 2223 (2013). <https://doi.org/10.1038/srep02223>. <https://www.nature.com/articles/srep02223>
14. Kovacs, T.: XCS classifier system reliably evolves accurate, complete, and minimal representations for Boolean functions. In: Soft Computing in Engineering Design and Manufacturing, pp. 59–68. Springer (1998)
15. Liu, Y.Y., Slotine, J.J., Barabási, A.L.: Controllability of complex networks. *Nature* **473**(7346), 167 (2011)
16. Moschoyiannis, S., Elia, N., Penn, A., Lloyd, D.J.B., Knight, C.: A web-based tool for identifying strategic intervention points in complex systems. In: Proceedings of the Games for the Synthesis of Complex Systems (CASSTING'16 @ ETAPS 2016), EPTCS, vol. 220, pp. 39–52 (2016)
17. Savvopoulos, S., Moschoyiannis, S.: Impact of removing nodes on the controllability of complex networks. In: 6th Conference on Complex Networks and Applications, pp. 361–363 (2017)
18. Savvopoulos, S., Penn, A., Moschoyiannis, S.: On the interplay between topology and controllability of complex networks. In: Conference on Complex Systems (CCS'17) (2017)
19. Urbanowicz, R.J., Moore, J.H.: Learning classifier systems: a complete introduction, review, and roadmap. *J. Artif. Evol. Appl.* **2009**(1), 1–25 (2009)
20. Wilson, S.W.: Classifier fitness based on accuracy. *Evol. Comput.* **3**(2), 149–175 (1995)
21. Wilson, S.W.: Generalization in the XCS classifier system. In: Koza, J., Banzhaf, W., Chellapilla, K., Deb, K., Dorigo, M., Fogel, D., Garzon, M., Goldberg, D., Iba, H., Riolo R. (eds.) Genetic Programming 1998: Proceedings of the Third Annual Conference. Morgan Kaufmann, San Francisco, CA (1998)
22. Wilson, S.W.: Compact rulesets from XCSI. In: International Workshop on Learning Classifier Systems, pp. 197–208. Springer (2001)



Robustness Through Regime Flips in Collapsing Ecological Networks

Suresh Babu¹ and Gitanjali Yadav^{2,3()}

¹ School of Human Ecology, Ambedkar University of Delhi, Kashmere Gate, Delhi 110006, India

² Department of Plant Sciences, University of Cambridge, Downing Site, Cambridge CB23EA, UK
gy246@cam.ac.uk

³ Genomics & Systems Biology Laboratory, National Institute of Plant Genome Research (NIPGR), Aruna Asaf Ali Marg, New Delhi 10067, India

Abstract. There has been considerable progress in our perception of organized complexity in recent years. Recurrent debates on the dynamics and stability of complex systems have provided several insights, but it is very difficult to find identifiable patterns in the relationship between complex network structure and dynamics. Traditionally an arena for theoreticians, much of this research has been invigorated by demonstration of alternate stable states in real world ecosystems such as lakes, coral reefs, forests and grasslands. In this work, we use topological connectivity attributes of eighty six ecological networks and link these with random and targeted perturbations, to obtain general patterns of behaviour of complex real world systems. We have analyzed the response of each ecological network to individual, grouped and cascading extinctions, and the results suggest that most networks are robust to loss of specialists until specific thresholds are reached in terms of network geodesics. If the extinctions persist beyond these thresholds, a state change or ‘flip’ occurs and the structural properties are altered drastically, although the network does not collapse. As opposed to simpler or smaller networks, we find larger networks to contain multiple states that may in turn, ensure long-term persistence, suggesting that complexity can endow resilience to ecosystems. The concept of critical transitions in ecological networks and the implications of these findings for complex systems characterized by networks are likely to be profound with immediate significance for ecosystem conservation, invasion biology and restoration ecology.

1 Introduction

Interest in ‘robust, yet fragile’ nature of complex systems transcends disciplinary domains of biology, engineering, sociology and ecology, with much to be gained through investigations into the behavior of complex dynamical systems like ecosystems that are robust by virtue of their continued existence in evolutionary time [1, 2]. Structural attributes shared by these systems can provide clues about their stability and robustness, and here we focus on large scale free real world systems such as ecological webs, which are known to display an unexpected degree of tolerance or structural

robustness to loss of specialist species [3, 4]. Studies on mutualistic networks have highlighted that modularity; one of the emergent properties of networks, endows robustness to systems as diverse as fire prone savannahs, spread of infectious diseases or financial networks like the Fedwire [5]. Compartmentalization has been shown to render the much needed robustness to these systems, suggesting that dynamics of large complex networks formed by interacting species impact the way biodiversity influences ecosystem functioning [2, 6]. Understanding the behavior of ecological networks, as envisaged in this work, is also central to understanding the response of biodiversity and ecosystems to perturbations.

Although there is adequate evidence to imply that structural and topological attributes of networks influence dynamics and function [7–9], the attributes of nodes and overall topological properties of networks that endow stability against perturbations are not sufficiently understood. The ‘targeted extinction’ approach for exploring the effects of node loss and associated co-extinctions has been well established over the last decade [3, 10]. We have automated this conventional approach through development of an online interactive web server called *NEXCADE* [11] that performs simulations of random or targeted primary extinctions on a network based on a user selected node attribute such as the number of links or ‘degree’. The response of the network in terms of resulting secondary extinctions or other topological parameters can then be visualized and sequentially investigated to infer the significance of the node attribute being studied [4, 12]. Extensive work on the robustness of ecological networks and attributes that enable species coexistence and diversity have revealed that these networks are highly robust to loss of specialists but are unable to withstand the targeted removal of generalists. This study was undertaken with the aim to understand how species persist in a collapsing mutualistic network following targeted extinctions and associated secondary coextinctions. The initial analysis was carried out using NEXCADE on primary frugivory data collected from Great Nicobar Island, India (GNIC) followed by exploration of response and behavior of the network, in terms of species richness, secondary extinctions, nestedness, fragmentation and diameter. The GNIC study led to the detection of alternate stable states that help sustain the integrity of the collapsing network. These states are identifiable in terms of two attributes of the Network Diameter. As a well-studied network attribute, the diameter is often interpreted to reveal the extent of internal communication within a network, both in terms of its exact size or pathlength (LDia), as well as the total number of diameters in a given network (NDia). In this work, we describe the GNIC frugivory network, our initial observations of alternate stable states in GNIC, followed by a large scale generalization and validation of our findings across eighty-five ecological networks of varying sizes. We report these data here, revealing that alternate states or flips may pervasively exist across all real world ecological networks, bestowing stability and robustness to ecosystems.

2 Methods

The GNIC Dataset: Primary data in the form of direct observations of foraging by vertebrates on fruits was collected from the tropical rainforests of Great Nicobar Island (spread from $6^{\circ} 45'$ to $7^{\circ} 15'N$ and $93^{\circ} 38'$ to $93^{\circ} 55'E$, spanning a total area of about 1045 km^2), the southernmost Island in the Andaman and Nicobar archipelago, India, spanning a period of seven years with field work being conducted on fifty nine transects, each 500 m long, in various regions of the island from December 1999 to November 2006. This study was undertaken as part of a larger initiative by the Ministry of Environment and Forests, Government of India, under the Man and Biosphere (MAB) Programme on Great Nicobar (GNIC) Biosphere Reserve, India. Direct observations of instances of foraging by vertebrates on fruits were recorded as an interaction matrix consisting of 181 plant species and 38 frugivores (33 birds and 5 mammals). Plant and frugivore species were identified and the interaction data obtained was compiled for the entire island. The GNIC data is a binary interaction matrix where nodes are species and edges represent an frugivore relationship between two species. Preliminary analysis and visualization of network architecture was done using Cytoscape [13] version 2.6.2.

Co-extinction Analysis: Primary species loss was simulated on GNIC by carrying out cascades of directed species removals/extinctions, based upon degree (the number of links). These sequential extinctions were performed in two opposing directions, namely specialist-first (i.e. least-linked to most-linked species) and generalist-first (most-linked to least linked species) cascades. Topological parameters of the resulting reduced networks were compared with random extinction cascades, following Memmott et al. [3]. Random removals were analysed after averaging from 300 replicates. In all extinction cascades, upon removal of a given node, those species that are left without any interaction are assumed to undergo co-extinction. The network remaining after every primary extinction and subsequent co-extinction was assessed for various network attributes commonly used to summarise patterns in ecological webs, such as degree, species richness, secondary extinction, fragmentation, lost interactions, degree-distribution-gamma values, axes, length and number of diameters etc.

Network Attributes: All network indices were calculated using in-house fortran scripts and R CRAN packages IGRAPH [14] version 0.5, SNA [15] version 1.5, and BIPARTITE [16] version 0.91. Detailed description of each of the indices can be found within the respective package manuals. We examined the exponential, power law and truncated power law models to cumulative distributions for each network. Nestedness was calculated using the recently proposed nestedness metric NodF [17] using the ANINHADO program [18]. To assess the significance of nestedness values, the observed NodF was compared with benchmarks provided by three different null models. For each network, a population of $n = 300$ random networks was generated for each null model. As a statistic indicating significance, we estimated the probability, p , that a randomization was equally or more nested than the real matrix. Only the significant NodF values were used for further analysis. Comparison of nestedness across reduced networks was done without normalizing these values for variation in species richness or number of interactions, since each reduced network is essentially a subset of the original unperturbed network.

Robustness and Regime Flips: For any given network, the retention of connectivity between any two pairs of nodes is considered as the most important feature in terms of communication or information flow [2, 9]. Accordingly, in all our comparative assessments, networks that preserve a single connected character are treated as more stable, as compared to networks that undergo fragmentation, since a connected web enables flow of information between any two nodes. This ability to communicate between network nodes is further measured in terms of the diameter of the network. The shortest paths (also called geodesics) were calculated by using breadth-first search in the graph. The diameter (L_{Dia}) of a graph is defined as the length of the longest geodesic. The number of diameters (N_{Dia}) was calculated as the sum of all diameters between every pair of nodes separated by a distance equivalent to the diameter (L_{Dia}). The N_{Dia} value, representing the total number of longest geodesics shortest in the entire network is used as a proxy for assessing network robustness and stability. The higher this number, the greater the interconnectivity or communication between any two nodes of the network. A regime flip is considered to have taken place when the network abruptly increases its N_{Dia} by reducing its L_{Dia} value.

Open Access Source Code: A Unix program was designed to automate the entire analysis. This code takes a given binary network as input, simulates different co-extinction sequences and evaluates the sub-network remaining after every subsequent species removal, for its stability and robustness, and then extracts the attributes required for detection of regime flips or alternate stable states. For each reduced network, it creates a list of extinct and co-extinct species and calculates seven network level indices, namely species richness, secondary extinction, lost and remaining interactions, number of fragments, L_{Dia} and N_{Dia} and compares these indices across and between the different extinction sequences, and finally plots the results into vector format files. The entire source code has been developed into an online interactive open source web server, namely NEXCADE, available freely at www.nipgr.res.in/nexcade.html [11].

Validation Through Meta-Analysis: In addition to GNIC, data records were obtained from a set of 85 ecological networks using previously published reports as well as the Interaction Web Database repository at the National Centre for Ecological Analysis and Synthesis (NCEAS) website (<http://www.nceas.ucsb.edu/interactionweb>). These 85 webs include one Anemone Fish network, four plant-herbivore, four ant-plant, seven host-parasite, one Predator-prey, 25 Seed dispersal or Frugivory networks and 43 Pollination networks. Each network was analysed using the NEXCADE code described above and subsequently examined for the occurrence of regime flips, as they appeared on plots of N_{Dia} and L_{Dia} with primary extinctions. Statistical analyses on the results across networks were carried out in R (V 2.11.0).

3 Results

Primary Data: The GNIC is a bipartite network having 812 interactions between 38 frugivores and 181 tree species, as shown in Fig. 1. GNIC has highly asymmetric interactions, a characteristic path length of three, and a diameter of six ($L_{Dia} = 6$). Several diameters are highlighted in the Figure, and the unperturbed network contains 10065 such diameters i.e. independent shortest paths of length six, between a given pair

of nodes ($NDia = 11065$). As expected of ecological networks, species interact with nested subsets of partners as shown for one mammal visiting thirteen different trees (Fig. 1). The nestedness of GNIC is high (NodF value 21.02) and its degree distribution showed best fit to a truncated power law distribution. In comparison with other frugivory webs reported to date, GNIC has higher links per species (L/S), greater density, asymmetry and specialization, and a comparatively lower Connectance (L/S^2).

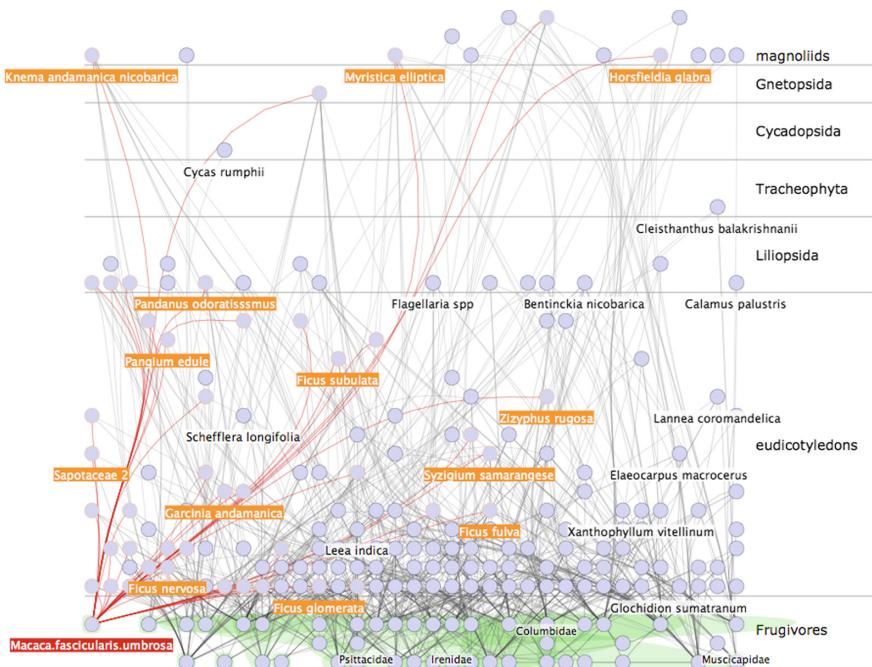


Fig. 1. Illustration of the bipartite GNIC frugivory network from Great Nicobar Island, India. Nodes represent species (green-plants; pink-birds, blue-mammals); Edges are interactions. Plants have been arranged horizontally into major taxonomic classes.

Sequential Co-extinctions and Network Robustness: Degree based co-extinction simulations were carried out for GNIC, in two opposing degree-based sequences, as described in Methods, in order to investigate the contrasting ability of mutualistic networks to withstand attacks on specialists as against generalists, the former a more realistic extinction threat. Distinct responses were observed: Generalist first extinction cascade of GNIC caused the species richness to plummet due to steep rise in secondary co-extinctions, whereas the specialist first cascade shows a linear decrease in species richness as it does not involve the loss of associated species, as shown in Fig. 2a–b. A similar effect was observed in case of Nestedness, one of the most significant and widely observed non-random pattern in networks of ecological interactions, that is known to greatly affect the robustness of mutualistic networks [19]. As specialists are removed, nestedness of the resulting networks tends to increase, while the removal of generalists triggers a rapid loss of nestedness in the corresponding reduced webs

(Fig. 2c), supporting the notion that nestedness provides alternate routes for system responses after perturbations such as link removals, and that extinctions of specialists improves the robustness of the reduced networks. Interestingly, in the specialists-first cascade, the graph does not fragment unless at the very end (Fig. 2d), whereas, the reverse sequence (generalist first extinction) results in catastrophic network fragmentation into many disconnected sub-webs and complete collapse within the first 23% primary species removals. Our hypothesis was that in the specialist-first scenario, network attributes ‘re-wire’ to make the reduced network more compact, thereby maintaining optimal communication between the remaining nodes, and we explored this further as described below.

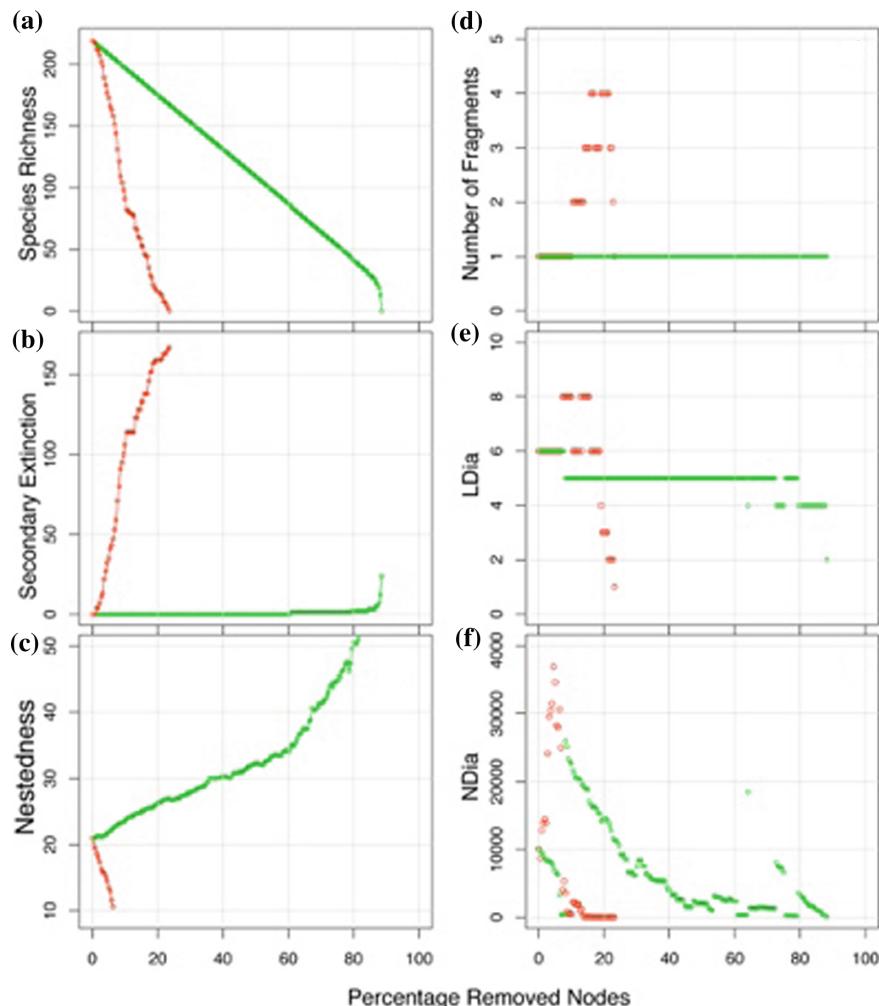


Fig. 2. GNIC response to targeted extinction in terms of six attributes resulting from generalists-first (red) and specialists-first (green) extinction sequences are plotted against the fraction of nodes removed. These six attributes are (a) species richness (b) secondary extinctions (c) nestedness (d) fragmentation (e) LDia and (f) NDia.

Emergent Properties of Collapsing networks: In order to understand the observed robustness of GNIC to loss of specialists, a detailed examination of both cascades was performed, in terms of bipartite network attributes such as degree-distribution exponent gamma, density, asymmetry, connectance, generality, specialisation, C-score, V-ratio and various aspects of geodesics, including number, length and unique sets of shortest paths. Only two of these properties, showed significant patterns and these two represent mutually independent attributes describing internal network communication (LDia and NDia), as shown in Fig. 2e and f. Most prominently, these two attributes show a coordinated response when specialists are removed, revealing a characteristic pattern that is absent in case of generalists-first extinctions. Defined as the longest geodesic of the network, the diameter has been relatively less studied in mutualistic webs, despite being a well established measure of topological robustness of several complex communication systems, ranging from cells to social, civilian networks and the Internet [10]. For a given network, a low diameter is considered advantageous as it can contribute to greater interconnectedness, shorter communication paths and lower load on links, or edges. As described in methods, we examined two aspects of the diameter: (a) its length or 'LDia', and (b) the number of diameters or 'NDia', and detected a topological re-adjustment between LDia and NDia in the specialist-first extinction scenario, presumably an internal compensation that endows the perturbed network with the ability to avoid fragmentation. Every instance of a very low NDia value coincides exactly with a corresponding single unit change in the LDia value, leading to a reversal of NDia reduction. Recovery of sufficient number of NDia in the collapsing network presumably enable it to maintain communication between remaining nodes, which stay connected despite the sustained perturbations. Such a coupling was not observed in the generalist-first extinction sequence, where the perturbed network undergoes multiple fragmentations, driven by an excessive decline of NDia.

Alternate Stable States circumvent Network Fragmentation: As can be seen in Fig. 2e–f, in the generalists-first extinction sequence, LDia and NDia both undergo a steep decrease and rapid collapse, a trend that corresponds with NDia curves; loss of generalists leads to a brief but drastic increase in NDia (to over 36900 at just under 5% primary extinctions), after which it steeply drops (to 455 by 10% deletions). This low number of diameters (NDia) corresponds to a failure of internal communication and subsequently the network undergoes fragmentation. Juxtaposition of plots 2d and e shows that the first instance of fragmentation in the collapsing network occurs at about 10% deletions, coinciding exactly with the lowest value of NDia. Further node deletions rapidly result in more fragments and the network collapses by 23% removals. In contrast, when specialists are removed first, LDia remains constant and NDia decreases steadily. By about 8% deletions, NDia reaches its lowest value of 436. However, the plot in Fig. 2d shows that despite minimal internal communication, the single unit connected character of the collapsing network is preserved. Interestingly, the lowest value of NDia corresponds to a single unit reduction in the LDia, which in turn, results in a steep recovery of NDia values (from 436 to 25966). Subsequently this pattern repeats itself iteratively, i.e., NDia decreases at pace with loss of specialists till about 80% extinctions. At its lowest value, it drastically rises again - corresponding to a further unit change in LDia. Evidently, the coordinated response between NDia and LDia and the associated renewal of internal communication, makes it possible for the

reduced network to make a stable transition to a new state and remain unfragmented, all through the specialist-first extinction sequence. Such a compensatory ‘flip’ response between two mutually exclusive network attributes, specific to the specialist-first scenario, and absent in the generalist-first scenario, has not been reported before. Complete details of this analysis for all 86 webs, along with the number of fragments, co-extinctions, LDia and NDia measured after every consecutive species deletion are in Nexcade.

Meta-Analysis of Ecological Networks: A comparative analysis of 85 additional ecological networks showed these patterns in LDia, NDia and fragmentations to be consistent and pervasive across all networks during the specialists-first breakdown scenario, and not limited to mutualistic webs only. As with GNIC the coordinated variation between NDia and LDia values endowed robustness to the perturbed networks and they persisted as single connected units during the attacks. In several cases, the different states were more pronounced than observed for GNIC. At least two and upto six flips were observed across the networks. In networks with low interaction

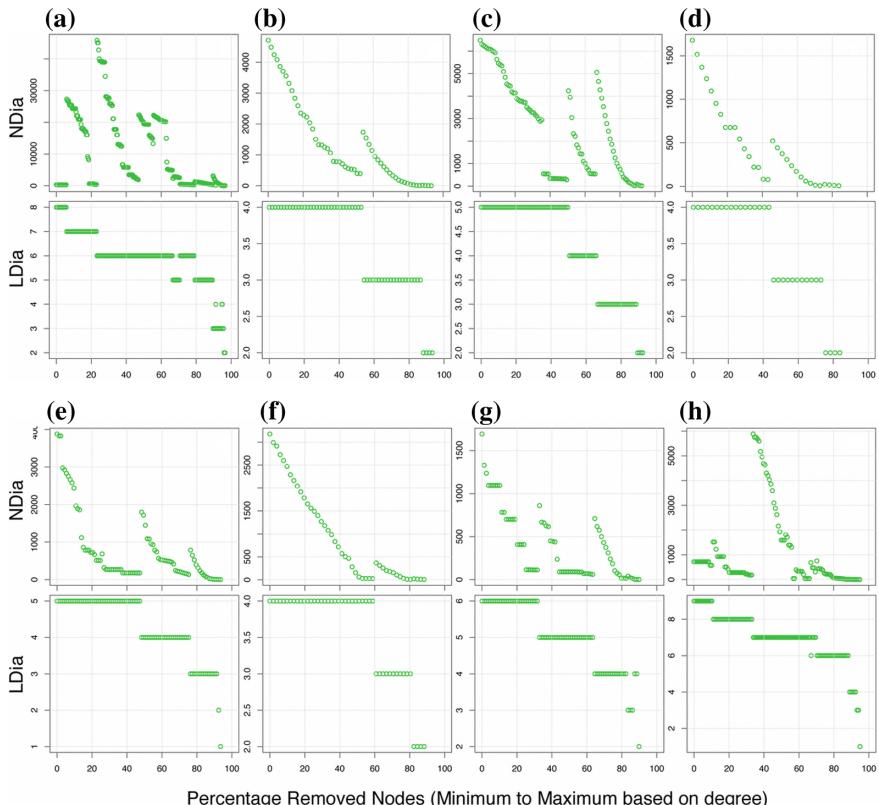


Fig. 3. Variation between NDia and LDia in eight representative ecological networks following extinction cascades, resulting in flips and alternate stable states. For each network, the panels contain the corresponding NDia and LDia plots arranged vertically below each other

density (<1.35), the transition between states was not very clear. In cases where the initial network was disconnected, the specialist-first extinction sequence began with the removal of the smaller unit/s, and eventual persistence of the single largest unit. Figures 3a and 3b depict the synchronised behaviour observed in two of the 25 frugivory webs studied (codes SILV and JOR1). Figure 3c shows the flips observed in MEMO, a pollination network with 299 interactions among 104 species. Figures 3d and e depict similar plots for an anemone-fish network (ANEM), and an ant-plant network (BLUT) respectively. Figures 3f-h depict the LDia-NDia plots for a host-parasite, plant-herbivore and a predator-prey network (LAKE, JEOM and MART) respectively.

Evidence for generalised regime flips: For all networks, ‘resistance to flip’ was estimated in terms of percentage of primary species extinctions after which the first flip was observed. Therefore ‘resistance to flip’ is higher if a large number of species deletions are required before the flip is observed, and lower if fewer node deletions cause the state change. Figure 4a shows the relationship between the ‘resistance to flip’ and initial LDia across the 86 ecological networks. Compact networks with small diameters require over 70% primary extinctions for a flip in their state (Fig. 4a). Networks with initial diameters of 6 or more require a much smaller proportion of primary extinctions to switch to lower diameters. A linear regression of ‘resistance to flip’ on initial LDia indicates a significant negative relationship (Adjusted $R^2 = 0.3598$, $\beta = -7.455$, $df = 84$, p-value: $6.288e-10$). However a Loess plot of the same indicates a curvilinear rather than a linear relationship between the variables, when examined at different spans (Fig. 4b). The data was found to meet the assumptions of homoscedasticity and normality of errors. It was also observed that a positive, nearly linear relationship exists between the likely number of flips and the initial diameter of the network (Linear model; Adjusted $R^2 = 0.5981$, $\beta = 0.559$, $df = 84$, p-value: $<2.2e-16$). Results of the entire analysis of 86 ecological networks are available from the NEXCADE website.

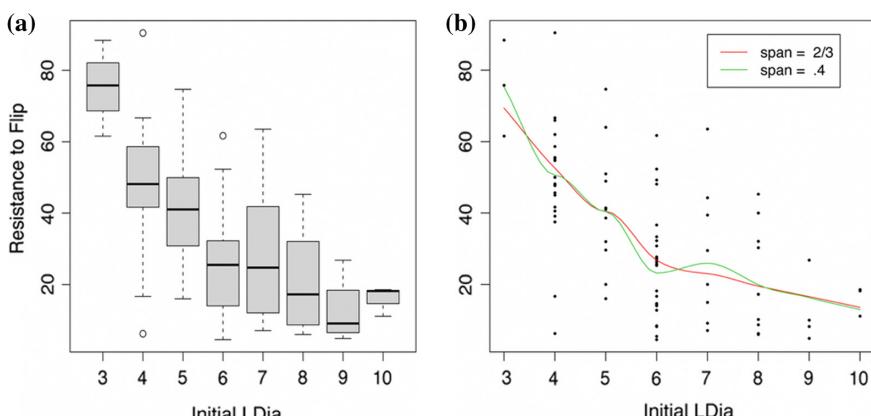


Fig. 4. Influence of initial LDia of a network on its ‘resistance to flip’ or the primary extinctions required for a flip. The box plot provides a summary of the observations for 86 networks analysed in this study (a), while (b) Loess of the same.

4 Discussion

Co-extinctions are now recognized as a major driver of global biodiversity loss, along with habitat destruction, species invasion and overkill [20]. Since more than half of all known species and a large proportion of unnamed ones are involved in host specific relationships in atleast some stage of their life, specialists face a greater risk due to secondary extinctions [21, 22]. There is added relevance of re-examining the threats of extinction knowing that interacting species may exist in alternate states. Besides broad implications on our understanding of bipartite networks in general, our findings have significance in conservation biology, invasion biology, and restoration ecology. Based on the present positioning of an interaction network along an extinction cascade, it may be possible to predict the proximity of the system to a catastrophic change and model real time stability indicators of networks. In addition to the dynamics associated with ‘critical slowing down’ [23], this may be an alternate approach to predict the likelihood and proximity of a system to regime flips. Conservation programmes could benefit from directly identifying the most threatened systems, requiring immediate attention or prioritization.

The iterating pattern of gradual decrease in NDi_a till a threshold of extinctions is reached, followed by a sudden transition to a new high value at a lower LDia, resembles the behaviour of ecosystems that can exist in multiple states characterized by unique sets of conditions [24]. The theory of alternate stable states suggests that the discrete states are separated by thresholds and the system remains in one state unless perturbation is large enough to tip it over to the next state [16]. In case of GNIC, the

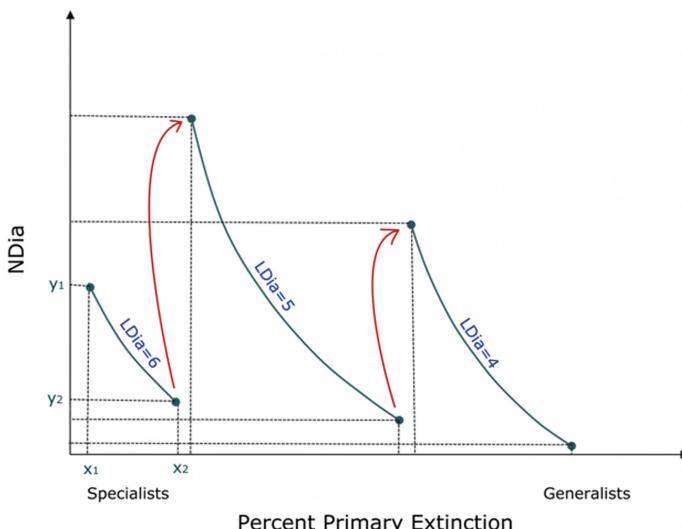


Fig. 5. A schematic framework for GNIC explaining alternate states in ecological networks. The X-axis corresponds to percentage primary extinctions with specialists being removed first along a specialist-generalist continuum. The Y-axis corresponds to NDi_a. Each state is depicted as a grey arc.

network retains its integrity by flipping between alternate levels of communication and complexity expressed in terms of LDia. However for the reduced network, the increased NDia now endows the system with high resilience, as the threshold required for the next flip or shift in LDia requires over 60% primary extinctions. The state with the widest stability basin, characterized by the maximum range of NDia at a given LDia, provides much of the robustness of the network. Figure 5 depicts this in a schematic representation.

Hysteresis or path dependency, characteristic of alternate equilibria, becomes evident once a flip in LDia has occurred. If the lost node were to be returned to the network at this stage, it may not bring the system back to the previous state. Rather, it would lead to an increase in the NDia within the current state, i.e. at the new value of LDia. This is likely because of the increasingly nested pattern of the reduced networks (Fig. 2c) a new species is likely to preferentially attach to the ‘hub’ nodes or generalists [6]. Attachment to a hub node does not lead to an increase in LDia; it can only result in additional alternate paths or NDia. As a result, the system will not return to its previous state just by a simple reversal of extinction, or re-introduction of lost species. This observation may have wide implications in the area of restoration ecology and invasion biology, as we discuss later.

The following generalizations emerge from our observations on GNIC: (a) total number of alternate paths or diameters (NDia) decrease with loss of specialists, (b) for a given network, reduction in the diameter (LDia) increases the NDia, (c) the LDia reduction occurs only at, or beyond, a critical loss of specialists, (d) the network precludes fragmentation, with the loss of specialists across the entire cascade and (e) for a given network, there may be several alternate stable states that can spring surprises against slow moving perturbations which can be masked by internal adjustments of the network. The generality of these observed patterns in LDia-NDia was established by a comparative analysis across 85 additional ecological networks including mutualisms as well as antagonistic webs. Our results show that the collapsing network sustains its connected or un-fragmented nature during the loss of specialists by internal structural readjustments in terms of LDia and NDia, which is not evident during the loss of generalists, thereby leading to immediate collapse. We also find that initial network size corresponds to the number of flips observed. Larger networks are likely to have more number of alternative stable states to cope with uncertainties in evolutionary time. For example, a small network like the anemone-fish network has only 36 species and an unperturbed LDia of 4, resulting in only one alternate stable state which may restrict its ability to withstand perturbations (Fig. 3d). Larger networks like the Brazilian Amazon (code SILV) and GNIC have several possible alternate stable states and are more likely to persist under long periods of adversity. Smaller perturbations tend to flip larger, more complex networks to alternate states (Fig. 4a and b) and since they have several such possible states, the network architecture endows resilience to such networks. The smaller, less complex networks do not show any state changes under small perturbations indicating resistance. However, since smaller networks also have very few possible alternate states, they are low on resilience. The width of stability basins and the number of possible stable states that accompanies the loss of specialists progressively shrinks, as the network size reduces, thus affecting its overall resilience. Therefore there may be an evolutionary advantage

in making ever larger webs of interactions that facilitate long-term persistence of species rich communities, a finding that complements a recent study [9] as to how mutualistic communities can enhance co-existence of species.

Implications of alternate states in ecological networks: Our results provide empirical evidence for the direct link that exists between topological heterogeneity and system dynamics. We show by means of detailed analysis of eighty-six ecological networks of varying nature that the networks can exist in alternate diameters and levels of communication. The outwards stability and unfragmented nature of these networks against perturbations often mask the internal re-wiring that progressively reduces their resilience resulting in sudden flips or transitions to lower levels of communication. This study shows that the continuous loss of specialists leads to significant loss of resilience for the networks, which is irreversible - something impossible to demonstrate experimentally. On one hand these findings hint at an evolutionary advantage in building ever-larger interaction networks (moving to higher levels of robustness), and on the other hand also highlights the inability of heavily damaged networks to respond to restoration in tangible amounts of time. The increased likelihood of an invasive species attaching to generalists in an impoverished native network partly explains its success in invaded ecosystems. The robustness of scale free networks could disguise enhanced percolation of disturbances across the network. This study establishes a prevailing pattern across known complex ecological networks and open ups possibilities for empirically driven dynamical modelling of these networks. We expect our findings to be the starting point for an array of investigations into the importance of alternate states in ecological networks in particular and other kinds of networks in general.

References

1. Carlson, J.M., Doyle, J.: Complexity and robustness. Proc. Natl. Acad. Sci. USA **99**(Suppl 1), 2538–2545 (2002)
2. May, R.M., Levin, S.A., Sugihara, G.: Ecology for bankers. Nature **451**, 893–895 (2008)
3. Memmott, J., Waser, N.M., Price, M.V.: Tolerance of pollinator networks to species extinctions. Proc. R. Soc. Lond. B **271**, 2605–2611 (2004)
4. Srinivasan, U.T., Dunne, J.A., Harte, J., Martinez, N.D.: Response of complex food webs to realistic extinction sequences. Ecology **88**, 671–682 (2007)
5. Bascompte, J., Jordano, P., Olesen, J.M.: Asymmetric coevolutionary networks facilitate biodiversity maintenance. Science **312**, 431–433 (2006)
6. Pascual, M., Dunne, J.A.: Ecological Networks: Linking Structure to Dynamics in Food Webs. Macmillan Publishers Limited, All rights reserved (2006)
7. Montoya, J.M., Pimm, S.L., Sole, R.V.: Ecological networks and their fragility. Nature **442**, 259–264 (2006)
8. Sugihara, G., Ye, H.: Complex systems: cooperative network dynamics. Nature **458**, 979–980 (2009)
9. Bastolla, U., Fortuna, M.A., Pascual-Garcia, A., Ferrera, A., Luque, B., et al.: The architecture of mutualistic networks minimizes competition and increases biodiversity. Nature **458**, 1018–1020 (2009)
10. Albert, R., Jeong, H., Barabasi, A.L.: Error and attack tolerance of complex networks. Nature **406**, 378–382 (2000)

11. Yadav, G., Babu, S.: Nexcascade: perturbation analysis for complex networks. *PLoS ONE* **7**(8), e41827 (2012)
12. Dunne, J.A., Williams, R.J.: Cascading extinctions and community collapse in model food webs. *Philos. Trans. R. Soc. Lond. B Biol. Sci.* **364**, 1711–1723 (2009)
13. Shannon, P., Markiel, A., Ozier, O., Baliga, N.S., Wang, J.T., et al.: Cytoscape: a software environment for integrated models of biomolecular interaction networks. *Genome Res.* **13**, 2498–2504 (2003)
14. Csardi, G., Nepusz, T.: The igraph software package for complex network research. *Complex Syst. Inter J.* **16**95 (2006)
15. Butts, C.T.: Social network analysis with sna. *J. Stat. Softw.* **24** (2008)
16. Dormann, C.F., Fründ, J., Bluthgen, N., Gruber, B.: Indices, graphs and null models: analyzing bipartite ecological networks. *Open Ecol. J.* **2**, 7–24 (2009)
17. Almeida-Neto, M., Guimarães, P., Jr. PRG, Loyola, R.D., Ulrich, W.: A consistent metric for nestedness analysis in ecological systems: reconciling concept and measurement. *Oikos* **117**, 1227–1239 (2008)
18. Guimarães, J.P.R., Guimarães, P.: Improving the analyses of nestedness for large sets of matrices. *Environ. Model Softw.* **21**, 1512–1513 (2006)
19. Jordano, P.J., Bascompte, J., Olesen, J.M., Waser, N.M.: The ecological consequences of complex topology and nested structure in pollination webs. In: Waser, N.M., Ollerton, J. (eds.) *Plant-Pollinator Interactions: From Specialization to Generalization*, pp. 173–199. University Of Chicago Press, Chicago, IL (2006)
20. Diamond, J.M.: The present, past and future of human-caused extinctions. *Philos. Trans. R. Soc. Lond. B Biol. Sci.* **325**, 469–476; discussion 476–467 (1989)
21. Koh, L.P., Dunn, R.R., Sodhi, N.S., Colwell, R.K., Proctor, H.C., et al.: Species coextinctions and the biodiversity crisis. *Science* **305**, 1632–1634 (2004)
22. Dunn, R.R., Harris, N.C., Colwell, R.K., Koh, L.P., Sodhi, N.S.: The sixth mass coextinction: are most endangered species parasites and mutualists? *Proc. Biol. Sci.* **276**, 3037–3045 (2009)
23. Scheffer, M., Bascompte, J., Brock, W.A., Brovkin, V., Carpenter, S.R., et al.: Early-warning signals for critical transitions. *Nature* **461**, 53–59 (2009)
24. Sutherland, J.P.: Multiple stable points in natural communities. *Am. Nat.* **108**, 859–873 (1974)



Enhancing Synchronization Stability in Complex Networks with Probabilistic Natural Frequencies

K. Y. Henry Tsang¹(✉), Bo Li², and K. Y. Michael Wong¹

¹ Department of Physics, The Hong Kong University of Science and Technology,
Hong Kong, Hong Kong
kytsangab@ust.hk

² Department of Science and Environmental Studies, The Education University
of Hong Kong, Hong Kong, Hong Kong

Abstract. Synchronization is crucial for different natural or artificial systems. In power grids, synchronization in the system is essential for stable electricity transmission. However, fluctuations in power supply and demand can destabilize synchronization, especially with the increasing deployment of renewable sources. In real-time applications, one can only access their probabilistic information in the near future. Hence the synchronization stability is no longer a well-defined value, and we need to minimize the tail of its distribution. Remarkably, we found that by optimizing the mean value of the synchronization stability, the variance is also reduced. Hence the load shedding scheme optimizing the mean stability is sufficient in the presence of probabilistic uncertainties of the natural frequencies. In addition, we introduce a vulnerability measure of individual nodes to demonstrate how the topology of the network affects the synchronization stability.

Keywords: State-dependent algebraic connectivity
Synchronization stability · Optimal resources adjustment

1 Introduction

Synchronization phenomena occur broadly in complex systems, such as neural systems, heart cells, wireless sensor networks and power grids [1–3]. To have a proper functioning of the synchronization systems, it is essential to maintain a robust synchronous state to guard against various kinds of perturbations and fluctuations. For example, power grids operate in the synchronized state such that electrical power can be transmitted to consumers [4]. Hence, they have to be designed to be resilient to disturbances [5] and real time active controls are used for maintaining a stable synchronization for normal operations [6]. Despite the advanced technology and paradigm of power grid controls, widespread blackout of different scales still occurred frequently [7]. Large blackouts are usually caused

by the cascading failures [8] which may be due to the loss of synchrony locally. Therefore, there is an upsurge of studies on the stability and robustness of the synchronization state, such as the findings about the destabilizing effects of inappropriate new links [9] and dead ends [10].

In recent years, with the introduction of renewable energy, there are new challenges about the stability in power grid systems [11]. Renewable energy generation usually has strong fluctuations. In real-time applications, one is often given the forecast of the power supply and demand in the near future, and needs to make an adjustment to ensure network stability. Hence the objective is to enhance the synchronization stability given a *probabilistic* distribution of resources instead of a fixed one [12]. A common approach is to recognize that the resource of each node has both controllable (deterministic) and uncontrollable (probabilistic) components, and the controllable components can be adjusted to maintain the synchronization stability, defined as the Lyapunov exponent when the network state is perturbed. This is often known as load shedding in power engineering [13].

Hence, in this paper, we study the minimization of the tail of the distribution of synchronization stability incorporating the probabilistic distribution of resources, whose means and variances are only known from forecast data. The objective is to minimize the tail of the distribution of state algebraic connectivity at zero value. Remarkably, using the optimization scheme in [14], we will show that the variance of the state algebraic connectivity decrease significantly during the process of maximizing the mean in first-order perturbations. Thus, the objective reduces to the maximization of the mean state algebraic connectivity. Similarly to [14], this objective is approached by an optimal adjustment scheme of resource (power supply and demand in the context of power grid) in the form of a constrained optimization problem, and introduce an efficient way of calculating the gradient and hessian of the objective function using the perturbation theory and the cut-set space approximation [15]. Using the optimal adjustment scheme, we investigate how the variance of the state algebraic connectivity drops during the optimization and we introduce a vulnerability measure of network nodes to discuss how the topology of the network affects the synchronization stability.

2 Linear Stability of Kuramoto Model

Synchronization in networks of coupled oscillators are commonly studied using the Kuramoto model [16]. Network stability in the presence of short-term fluctuations is characterized by the rate of relaxation towards the steady state. When the phase differences are small, the relaxation rate of the Kuramoto model is determined by the algebraic connectivity, that is, the second smallest eigenvalue of the graph Laplacian matrix. Since graph algebraic connectivity depends on the network topology, there were studies on how a network topology affects its synchronization stability [16, 17] and how to enhance the network synchronizability through maximizing the spectral gap of the graph [18]. It was also found that algebraic connectivity can be used in measuring the network connectivity

[19]. In our previous work, we found that due to the heterogeneous distribution of the natural frequencies, the stability is no longer only determined by the graph algebraic connectivity and its network structure [14]. Instead, we should use the state algebraic connectivity that incorporates the dependence on the system steady state [20]. The strategy enabled us to enhance the synchronization stability given the configuration of resources in the nodes.

We begin with the first-order Kuramoto model on a single connected network which is given by

$$\dot{\theta}_i = \omega_i + \sum_{j \in \partial i} K_{ij} \sin(\theta_j - \theta_i), \quad (1)$$

where θ_i is the phase angle, ω_i is the natural frequency of node i and $K_{ij} = K_{ji} > 0$ is the coupling strength between node i and j . ∂i is denoted as the set of neighbouring nodes of node i . In general, we assume $\sum_i \omega_i = 0$. Such assumption is equivalent to the balance of net power load in the power grid or balance of resources in the transportation network. The steady state of Eq. (1) is given by

$$0 = \omega_i + \sum_{j \in \partial i} K_{ij} \sin(\theta_j^* - \theta_i^*), \quad (2)$$

where θ^* is the steady state and is usually solved numerically. To study the linear stability of the system, consider a small perturbation on the phase angles that give a small deviations from the steady state $\delta\theta_i = \theta_i - \theta_i^*$, and we obtain

$$\delta\dot{\theta}_i \approx \sum_{j \in \partial i} K_{ij} \cos(\theta_j^* - \theta_i^*) (\delta\theta_j - \delta\theta_i) = - \sum_j L(\theta^*)_{ij} \delta\theta_j, \quad (3)$$

where $L(\theta^*)_{ij} := \delta_{ij} \sum_l K_{il} \cos(\theta_l^* - \theta_i^*) - K_{ij} (\theta_j^* - \theta_i^*)$ is the Laplacian matrix. Unlike the usual graph Laplacian matrix $L[K] = \delta_{ij} \sum_l K_{il} - K_{ij}$ that depends only on edge weights and network topology, $L(\theta^*)$ further depends on the steady state θ^* (due to the dependence on θ^* in the edge weights $W_{ij}(\theta^*) = K_{ij} \cos(\theta_j^* - \theta_i^*)$) and we denote it as the state-dependent Laplacian matrix. Suppose $|\theta_j^* - \theta_i^*| < \pi/2$ for all edges such that all $W_{ij} > 0$, then $L(\theta^*)$ is positive semi-definite. The lowest eigenvalue is 0 corresponding to a global phase shift and all other eigenvalues are positive which make the dynamical system locally exponentially stable. Moreover, the slowest mode corresponds to the second smallest eigenvalue of $L(\theta^*)$. We call this eigenvalue, $\lambda_2(L(\theta^*))$, the state algebraic connectivity, to distinguish it from the usual graph algebraic connectivity $\lambda_2(L(K))$. $\lambda_2(L(\theta^*))$ indicates the inverse time scale of the worst case decay of $\delta\theta$ in Eq. (3). Thus, increasing $\lambda_2(L(\theta^*))$ can indeed improve the stability of the network [21]. The key objective in this paper will then be focused on how to maximize $\lambda_2(L(\theta^*))$ with ω_i being the control variable (the steady state θ^* depends implicitly on ω through Eq. (2)).

So far, we have only considered the first-order Kuramoto model in the analysis, yet, it has been shown that $\lambda_2(L(\theta^*))$ can also indicate the synchronization stability in the second-order Kuramoto model [14]. Therefore, the methods of optimizing $\lambda_2(L(\theta^*))$ developed in this paper can also be applicable in improving

the stability of the second-order Kuramoto model. For simplicity, we will focus on the first-order Kuramoto model in the rest of this paper. Additionally, the Kuramoto model is widely used for modelling the power grid [22] in which one can treat the natural frequency ω_i as net power input P_i in the power grid (or resources in transportation network) and $K_{ij} \sin(\theta_j^* - \theta_i^*)$ as the power (resources) transmitted from node j to node i . Nodes with $P_i > 0$ is a net generator while $P_i < 0$ is a net consumer. Equation (2) can be viewed as the condition of flow conservation on each node or the AC power flow equation. Thus, our approach on controlling the natural frequency ω to optimize $\lambda_2(L(\theta^*))$ can have potential application in power scheduling to increase the synchronization stability in power grid networks.

3 Optimization Method

As a first step, we maximize $\lambda_2(L(\theta^*))$ to improve the robustness of the first and second-order Kuramoto model with the natural frequencies as control variables. However, there is no analytical expression for $\lambda_2(L(\theta^*))$ and it is computational costly to directly optimize $\lambda_2(L(\theta^*))$. Thus, we adopt an efficiency optimization method developed in [14] to optimize the state algebraic connectivity. Only the main steps are shown here. Since there is no analytic form of $\lambda_2(L(\theta^*))$, we derive the variation of $\lambda_2(L(\theta^*))$ using the first-order perturbation theory. For simplicity, we assume that the state algebraic connectivity is non-degenerate, which usually holds when the corresponding graph algebraic connectivity is non-degenerate. With this assumption, we can obtain the gradient of the state algebraic connectivity with respect to ω_i as

$$\frac{\partial \lambda_2(L(\theta^*))}{\partial \omega_k} = \sum_{(ij)} \frac{\delta W_{ij}(\theta^*)}{\delta \omega_k} [\nu_2(\theta^*)_i - \nu_2(\theta^*)_j]^2, \quad (4)$$

where $\nu_2(\theta^*)$, known as the Fiedler vector, is the normalized eigenvector of $L(\theta^*)$ corresponding to $\lambda_2(L(\theta^*))$ and $W_{ij}(\theta^*) = K_{ij} \cos(\theta_j^* - \theta_i^*)$. However, directly using the gradient with the above form is computational expensive for optimization. Whenever we update the value of ω , there is a shift in the steady state θ^* and since $W_{ij}(\theta^*)$ depends on the state θ^* , for every iteration during the optimization, one must solve Eq. (2) to obtain the steady state again. This process is extremely time consuming and therefore, we simplify this process using the cut-set space approximation of network flow.

The computational complexity comes from the necessity of solving the steady state of non-linear Eq. (2) in every iteration during the optimization. However, it has been found that using the cut-set space approximation of power flows, one can simplify the problem and it is very accurate in many regimes [15]. Using the cut-set space approximation, we can omit the need for the stepwise solution of the nonlinear flow equation by approximating $\sin(\theta_j^* - \theta_i^*) \approx \phi_j - \phi_i$, where ϕ_i is a potential function given by $\phi = G[K]\omega$. $G[K]$ is the discrete Green's function which is the generalized inverse of the graph Laplacian matrix. Since the discrete

Green's function $G[K]$ is independent of state (as the graph Laplacian matrix is independent of state), one can simply calculate and record $G[K]$ at the beginning of the optimization process. Applying the cut-set space approximation to the problem, one can obtain the edge weight $W_{ij}(\theta^*) \approx K_{ij}\sqrt{1 - (\phi_j - \phi_i)^2}$. This approximation can save lots of computational time, as we can obtain W_{ij} by solving ϕ which requires only simple matrix operations. Note that we always assume $|\phi_j - \phi_i| < 1$ such that the approximated $W_{ij}(\theta^*)$ are always real valued. From the above approximation methods, we can obtain an explicit expression of the gradient of $\lambda_2(L(\theta^*))$ with respect of ω . With a similar procedure, we can also obtain the Hessian and perform optimization using either gradient ascent or quasi-Newton method.

4 Optimization for Probabilistic Natural Frequencies

It is anticipated the future power grids will increase the usage of renewable energy. Yet, renewable energy such as wind and photovoltaics are extremely volatile which bring a large source of fluctuations to the power grid. With the increasing deployment of renewable energy, it may easily cause a local desynchronization in the grid and trigger a cascading failure. Usually, in dynamic control, only short-term predictions of such fluctuations of power supply and demand in the form of probabilistic distributions are available. Therefore, we are interested in improving the synchronization stability against fluctuations in the power grid by adjusting the power in the network, given the probabilistic information (such as the means and covariance) of the power generation and consumption. Since one can view the natural frequency ω_i as net power input P_i in the power grid, we formulate the problem by considering the case that the natural frequency consists of a controllable component π_i and a probabilistic component ω_i . In general, the controllable component can be treated as the power generated from natural gas or nuclear or the load to be shed while the probabilistic component can be treated as fluctuations in the renewable power generated from wind and photovoltaics or other perturbations such as power consumed by volatile consumers. Our objective is to adjust the controllable power so that the synchronization stability of the grid is maximized. In power engineering this is known as load shedding. To approach this problem, we write Eq. (2) as

$$0 = \pi_i + \omega_i + \sum_{j \in \partial i} K_{ij} \sin(\tilde{\theta}_j^* - \tilde{\theta}_i^*) . \quad (5)$$

π_i is the control variable for optimizing the state algebraic connectivity and $\tilde{\theta}^*$ is the steady state for a given instance drawn from the distribution. For the probabilistic component ω_i , we assume that $\omega_i = \omega_i^0 + \delta\omega_i$ where $\delta\omega_i$ follows a probabilistic distribution with mean 0. Instead of optimizing $\lambda_2(L(\theta^*))$ for a given instance of ω drawn from the distribution, the objective in the following is to optimize the average algebraic connectivity $\langle \lambda_2 \rangle$ (average over the power distribution of the probabilistic component) with π_i being the control variables.

To optimize $\langle \lambda_2 \rangle$, we adopt a similar procedure as described in the previous section. In practice, it is easier to derive the variation of $\lambda_2(L(\theta^*))$ for a given instance and then average over the power distribution afterwards. To consider the probabilistic distribution of ω_i , we expand $W_{ij}(\theta^*)$ up to first order in the cut-set space approximation as

$$W_{ij}(\theta^*) \approx K_{ij} \left\{ \sqrt{1 - \sum_{kl} (\omega_k^0 + \pi_k) A_{kl}^{(ij)} (\omega_l^0 + \pi_l)} \right. \\ \left. - \frac{\sum_{kl} (\omega_k^0 + \pi_k) A_{kl}^{(ij)} \delta\omega_l}{\sqrt{1 - \sum_{kl} (\omega_k^0 + \pi_k) A_{kl}^{(ij)} (\omega_l^0 + \pi_l)}} \right\}, \quad (6)$$

where $A_{kl}^{(ij)} = (G[K]_{jk} - G[K]_{ik})(G[K]_{jl} - G[K]_{il})$. Using Eq. (4) and after averaging over the power fluctuations ($\delta\omega$), we obtain the gradient of $\langle \lambda_2 \rangle$ with respect to π_a as

$$\frac{\partial \langle \lambda_2 \rangle}{\partial \pi_a} \approx - \sum_{(ij)} K_{ij} [\nu_2(\phi)_i - \nu_2(\phi)_j]^2 \frac{\sum_l A_{al}^{(ij)} (\omega_l^0 + \pi_l)}{\sqrt{1 - \sum_{kl} (\omega_k^0 + \pi_k) A_{kl}^{(ij)} (\omega_l^0 + \pi_l)}}. \quad (7)$$

Note that $\nu_2(\phi)$ is the Fiedler vector of the graph Laplacian matrix, which approximates the state dependent Laplacian matrix in the cut-set space approximation.

When the natural frequencies are probabilistic, the variance of the state algebraic connectivity should also be considered. Since we only optimize the average state algebraic connectivity above, it is important to examine the variance of the state algebraic connectivity in the optimal state. As verified by simulations presented in the next section, the variance reduces significantly during optimization, with the residual variance coming from higher order contributions. We can also see this by approximating the second moment of the state algebraic connectivity through expanding λ_2 up to first order of $\delta\omega$, which is given by

$$\langle \delta\lambda_2(\{\omega_l^0 + \pi_l\})^2 \rangle = \sum_{i,j} \frac{\partial \lambda_2}{\partial \pi_i} \Big|_{\{\omega_l^0 + \pi_l\}} \frac{\partial \lambda_2}{\partial \pi_j} \Big|_{\{\omega_l^0 + \pi_l\}} \langle \delta\omega_i \delta\omega_j \rangle. \quad (8)$$

Note that the derivative $\frac{\partial \lambda_2}{\partial \pi_i} \Big|_{\{\omega_l^0 + \pi_l\}}$ can be calculated using Eq. (7) and during the optimization, the value of the gradient will drop. As a result, we only optimize the average state algebraic connectivity, the variance of the state algebraic connectivity will decrease significantly as well once it is in the optimal state.

5 Stability Enhancement Results

After obtaining the gradient and Hessian of $\langle \lambda_2 \rangle$, we test the optimization algorithm and investigate the properties with simulations. Without imposing any

constraints other than $\sum_i \pi_i = 0$, maximizing $\langle \lambda_2 \rangle$ should lead to a solution with all natural frequency vanishing $\pi_i = -\omega_i^0$. This is because the value of $\lambda_2(L(\phi))$ is maximum when the phase angle difference along each edge vanishes. To avoid trivial solution and more importantly, to have a practical power controlling scheme, it requires a constraint to restrict the value of π_i . Hence, we introduce an additional constraint, $\sum_i |\pi_i| = c \sum_i |\pi_i^0 + \omega_i^0| = c \sum_i |\tilde{\omega}_i^0|$, where π_i^0 is the initial value before optimization and c is a parameter to control the total amount of changes. In fact, one can view c as the mean ratio to be changed for each node. For simplicity, we combine π_i^0 and ω_i^0 as $\tilde{\omega}_i^0$. Restricting the total amount of changes rather than the changes of each node (i.e. $\pi_i < c|\tilde{\omega}_i^0|$) can have a fair comparison between different adjustment schemes. Moreover, restricting the total amount of changes can help us to identify which nodes can easily endanger the synchronization stability. Combining all the constraints, we propose an optimal resource adjustment scheme by solving the following constrained optimization problem,

$$\begin{aligned} & \underset{\{\pi_i\}}{\text{maximize}} \quad \langle \lambda_2 \rangle, \\ & \text{subject to} \quad \sum_i |\pi_i| = c \sum_i |\tilde{\omega}_i^0|, \quad \sum_i \pi_i = 0. \end{aligned} \tag{9}$$

Since the constraints are linear in the control variables π_i , it can be solved using the projection methods for equality constrained problems together with the approximation method developed in the previous section. As a comparison, we also introduce the proportionate reduction scheme $\pi_i^{prop} = -c(\tilde{\omega}_i^0)$ which can proportionally reduce the flow in the network to reduce the stress in the network. We test both schemes in a Erdös-Rényi (ER) network with 50 nodes and the connection probability is 0.07 with 100 sample averaging. For simplicity, we set $K_{ij} = 1$. $\tilde{\omega}_i^0$ follow a Gaussian distribution with mean equal to 0 and variance equal to 0.32 with the fluctuation $\delta\omega_i$ also follows a Gaussian distribution with mean equal to 0 and variance equal to $0.3|\tilde{\omega}_i^0|$ independently. Figure 1 shows the simulation result. The y -axis is the mean state algebraic connectivity of the network while the x -axis is the shed load $1 - c$. The red dotted curve represents the adjustment using the proportionate reduction while the blue curve represents the optimal adjustment scheme. It is shown that in the optimal adjustment scheme, the network has a much higher stability compared to the proportionate reduction. Moreover, using the optimal adjustment scheme, the system has a large value $\langle \lambda_2 \rangle$ with a small value of c .

In the previous section, it was shown that the variance of the state algebraic connectivity is small once the system is in the optimal state. To see this, we plot the variance divided by the square of the mean of λ_2 during the optimization for the network simulated in Fig. 1. Figure 2 shows the simulation result. Compared with the unoptimized state, the variance of the state algebraic connectivity drops significantly. This shows that even when we merely optimize the average state algebraic connectivity, its variance is also reduced. Therefore, in the optimized state, the system has a high synchronization stability against probabilistic uncertainty.

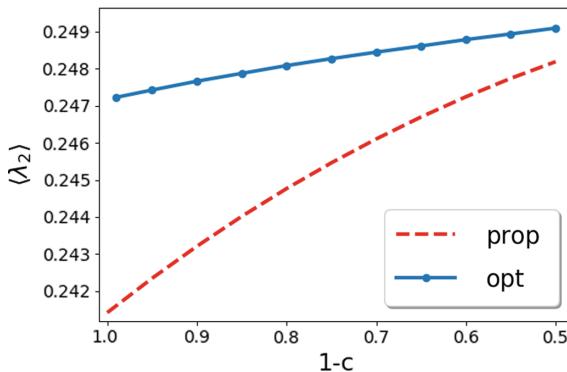


Fig. 1. The mean state algebraic connectivity as a function of the shed load $1 - c$. The red dotted curve represents the adjustment using the proportionate reduction while the blue curve represents the optimal adjustment scheme. This figure shows that using optimal adjustment scheme, the network have a much higher stability compared with the proportionate reduction. In addition, using the optimal adjustment scheme, the system can attain a significant improvement in stability by making a small adjustment in the natural frequencies.

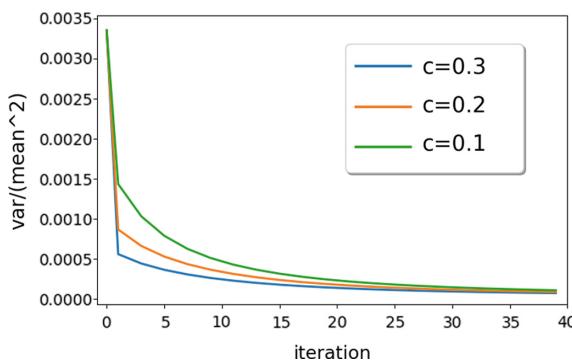


Fig. 2. The y -axis represents the variance divided by the square of the mean of λ_2 (from the sample averaging) while the x -axis represents the iteration steps in the optimization process. From the graph, the fluctuation in the state algebraic connectivity drops significantly compared with the unoptimized state (x -axis at 0). As expected, variance of the state algebraic connectivity is smaller with a larger c .

6 Node Vulnerability Results

The state algebraic connectivity can also facilitate the study of how the network topology affects the synchronization stability against fluctuations. For example, one can observe that nodes located at dead ends (or dead trees) or adjacent to a dead end (or dead tree) are vulnerable against probabilistic uncertainty and endanger the synchronization stability, agreeing with the prediction of the basin stability approach without going through lengthy simulations [10]. Vulnerable nodes are characterized by large changes in load during load shedding. Hence using the optimal adjustment scheme for a given c , we define the *vulnerability* of node i to be the ratio of changes in the natural frequency to reach the optimal state over the initial natural frequency (i.e. $|\pi_i|/|\tilde{\omega}_i^0|$) . It is found that nodes located at dead ends or connected to a dead tree usually have a high vulnerability. As an illustration, see Fig. 3 for a network with the same parameters as in Fig. 1. The reason why the dead ends may worsen the synchronization stability can be understood by the sensitivity of the state algebraic connectivity to be link connecting dead end. In general, the algebraic connectivity can be used in measuring how good a network is connected. Dead ends or dead trees are connected by a single link with the rest of the network and when this link is disconnected, the algebraic connectivity drops to zero. Thus, the state algebraic connectivity (as well as the synchronization stability) is sensitive to the presence of such links. With probabilistic uncertainty of the resources, those links are stressed and can easily endanger the synchronization stability.

For further illustration, the network on the left in Fig. 4(a) is constructed by connecting the IEEE Reliability Test System 96 (RTS 96) power network [23] to a tree and has $\lambda_2^{opt} = 0.0395$ using $c = 0.2$. The most vulnerable node, indicated as the red node, has the vulnerability equal to 0.56 and is located at the tree. Since there is only one link connecting the tree and the regular network, the current flow has to pass through this link rendering this link extremely stressful, especially when there are fluctuations. Moreover, since this edge is the only linkage between the regular network and the tree, it greatly influences the algebraic connectivity. In order to maintain the stability and satisfy the flow conservation in Eq. (2), the optimal adjustment scheme requires a large amount of resource to be shed from that node.

When we add a new link to that node (indicated by the blue line in Fig. 4(b)), the state algebraic connectivity increases to $\lambda_2^{opt} = 0.0413$ as the network is better connected (the regular network and the tree is now connected with two links). Moreover, with the additional link, the current can flow through the additional link and that node has reduced its vulnerability to 0.25. This effect has also been shown in a case study of the Northern European power grid in [10] in which the addition of transmission lines to dead trees can enhance stability. Further investigations about how the rewiring or adding of links affect the state algebraic connectivity is useful for network design.

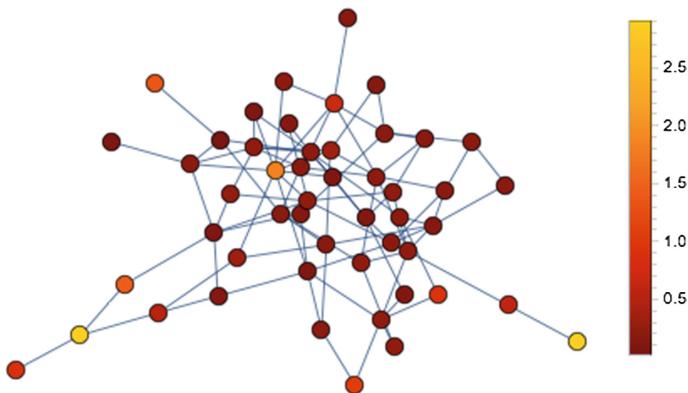


Fig. 3. This figure shows an example of an ER network with 50 nodes. The color of each node represent the vulnerability (shown in the color bar) and we observed that nodes with high vulnerability are usually located at the dead ends or adjacent to a dead ends.

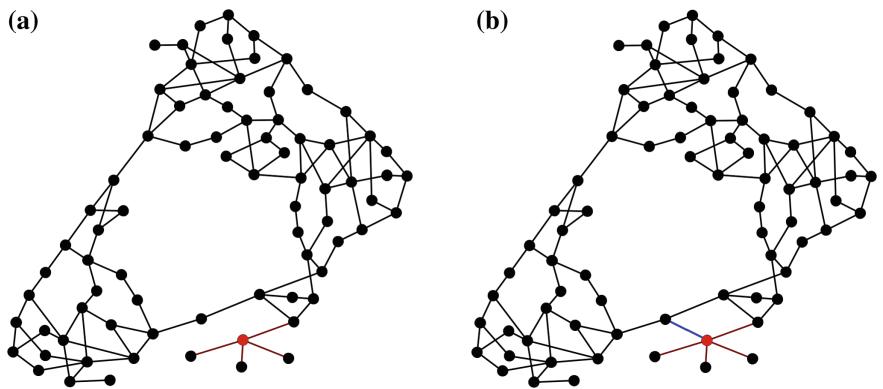


Fig. 4. (a) A network with $\lambda_2^{opt} = 0.0395$ at $c = 0.2$ is constructed by connecting the RTS96 power network to a tree (indicated by brown lines). It shows that the node (red node) in the dead tree is the most vulnerable node whose vulnerability is 0.56. (b) When we add a new link to that node (blue line in the network on the right), the state algebraic connectivity $\lambda_2^{opt} = 0.0413$ increases due to a better connection. Moreover, with the additional link, that node enhances the stability as the vulnerability decrease to 0.25. (color figure online)

7 Conclusion

In this paper, we have studied the enhancement of the synchronization stability against probabilistic uncertainties of the natural frequencies in the Kuramoto model by optimizing the average state algebraic connectivity with natural frequencies as control variables. For a meaningful adjustment in the frequencies, we proposed an optimal adjustment scheme by solving a constrained optimization problem with the average state algebraic connectivity as the objective function. Using the optimal adjustment scheme, the average state algebraic connectivity increases significantly compared with proportionate reduction in the natural frequency. It is also found that in the optimal state, the variance of the state algebraic connectivity distribution drops significantly, indicating that network has a high synchronization stability even when there is uncertainty in the nature frequencies. Since power grids can be modelled by the Kuramoto model, the method presented in this paper can have potential applications in power control to improve the stability in power grids. Furthermore, we introduced a vulnerability measure to identify the vulnerable nodes, which is useful for local approaches to stability enhancement. However, there are still many aspects about the state algebraic connectivity worth studying. For example, we have found that nodes located at dead ends or adjacent to a dead end can worsen the stability and it is useful to study how to increase the state algebraic connectivity by other methods, such as a better rewiring or adding new links to those nodes. Additionally, there are other approaches on studying synchronization stability such as the basin stability approach with higher computational complexity and some similar observations can also be obtained by using state algebraic connectivity. So, it is beneficial to investigate the relations of these approaches.

Acknowledgement. We thank David Saad for fruitful discussions. This work is supported by research grants from the Research Grants Council of Hong Kong (grant numbers 16322616 and 16306817).

References

1. Pikovsky, A., Rosenblum, M., Kurths, J.: *Synchronization: A Universal Concept in Nonlinear Science*. Cambridge University Press, Cambridge (2002)
2. Simeone, O., Spagnolini, U.: Distributed time synchronization in wireless sensor networks with coupled discrete-time oscillators. *EURASIP J. Wirel. Commun. Netw.* **2007**(1), 057054 (2007)
3. Kundur, P., Balu, N.J., Lauby, M.G.: *Power System Stability and Control*, vol. 7. McGraw-Hill, New York (1994)
4. Hill, D.J., Chen, G.: Power systems as dynamic networks. In: 2006 IEEE International Symposium on Circuits and Systems, 2006. ISCAS 2006. Proceedings. IEEE (2006)
5. Schmietendorf, K., Peinke, J., Friedrich, R., Kamps, O.: Self-organized synchronization and voltage stability in networks of synchronous machines. *Eur. Phys. J. Spec. Top.* **223**(12), 2577–2592 (2014)

6. Machowski, J., Bialek, J., Bumby, J.R., Bumby, J.: Power System Dynamics and Stability. Wiley, New York (1997)
7. <http://www.nerc.com/dawg/database.html>. Information on electric systems disturbances in North America
8. Crucitti, P., Latora, V., Marchiori, M.: Model for cascading failures in complex networks. *Phys. Rev. E* **69**(4), 045104 (2004)
9. Witthaut, D., Timme, M.: Braess's paradox in oscillator networks, desynchronization and poweroutage. *New J. Phys.* **14**(8), 083036 (2012)
10. Menck, P.J., Heitzig, J., Kurths, J., Schellnhuber, H.J.: How dead ends undermine power grid stability. *Nat. Commun.* **5**, 3969 (2014)
11. Harrison, E., Saad, D., Michael Wong, K.Y.: Message passing for distributed optimisation of power allocation with renewable resources. In: 2016 2nd International Conference on Intelligent Green Building and Smart Grid (IGBSG), pp. 1–6. IEEE (2016)
12. Bienstock, D., Chertkov, M., Harnett, S.: Chance-constrained optimal power flow: risk-aware network control under uncertainty. *SIAM Rev.* **56**(3), 461–495 (2014)
13. Bienstock, D.: Optimal control of cascading power grid failures. In: 2011 50th IEEE Conference on Decision and Control and European Control Conference (CDC-ECC), pp. 2166–2173. IEEE (2011)
14. Li, B., Michael Wong, K.Y.: Optimizing synchronization stability of the Kuramoto model in complex networks and power grids. *Phys. Rev. E* **95**(1), 012207 (2017)
15. Dörfler, F., Chertkov, M., Bullo, F.: Synchronization in complex oscillator networks and smart grids. *Proc. Natl. Acad. Sci.* **110**(6), 2005–2010 (2013)
16. Arenas, A., Díaz-Guilera, A., Kurths, J., Moreno, Y., Zhou, C.: Synchronization in complex networks. *Phys. Rep.* **469**(3), 93–153 (2008)
17. Dörfler, F., Bullo, F.: Synchronization in complex networks of phase oscillators: a survey. *Automatica* **50**(6), 1539–1564 (2014)
18. Milanese, A., Sun, J., Nishikawa, T.: Approximating spectral impact of structural perturbations in large networks. *Phys. Rev. E* **81**(4), 046112 (2010)
19. Chung, F.R.K., Graham, F.C.: Spectral Graph Theory, vol. 92. American Mathematical Society, Providence (1997)
20. Arapostathis, A., Sastry, S., Varaiya, P.: Analysis of power-flow equation. *Int. J. Electr. Power Energy Syst.* **3**(3), 115–126 (1981)
21. Mallada, E., Tang, A.: Improving damping of power networks: power scheduling and impedance adaptation. In: 2011 50th IEEE Conference on Decision and Control and European Control Conference (CDC-ECC), pp. 7729–7734. IEEE (2011)
22. Grzybowski, J.M.V., Macau, E.E.N., Yoneyama, T.: On synchronization in power-grids modelled as networks of second-order Kuramoto oscillators. *Chaos Interdiscip. J. Nonlinear Sci.* **26**(11), 113113 (2016)
23. Grigg, C., et al.: The IEEE reliability test system-1996. a report prepared by the reliability test system task force of the application of probability methods subcommittee. *IEEE Trans. Power Syst.* **14**(3), 1010–1020 (1999)



Identifying Vulnerable Nodes to Cascading Failures: Centrality to the Rescue

Richard J. La^(✉)

University of Maryland, College Park, MD 20742, USA
hyongla@umd.edu
<http://www.ece.umd.edu/~hyongla>

Abstract. We study the problem of identifying nodes that are more likely to trigger cascading failures in complex systems, which we call *vulnerable nodes*. We show that there is a close relation between the likelihood of a node setting off cascading failures (which we call the cascading failure probability) and its non-backtracking centrality; when every failed node is equally likely to cause the failure of each neighbor, the cascading failure probability and non-backtracking centrality of a node are proportional to each other. Based on this observation, we propose a new approach to finding vulnerable nodes and study its performance using numerical studies.

Keywords: Cascading failures · Nonbacktracking centrality

1 Introduction

Many modern systems that provide critical services consist of interdependent systems. Examples include modern information and communication networks/systems, power systems, manufacturing systems, and transportation systems, among others. In order to deliver their services, the comprising or component systems (which we shall refer to simply as nodes) must work together and oftentimes support each other.

Growing interdependence among nodes also exposes a source of vulnerability: the failure of one node can spread to other nodes via dependence because some nodes may no longer be able to function without the support from failed node(s). From this viewpoint, it is obvious that the overall robustness of the system to failures will depend on the underlying dependence structure among nodes. We adopt a *dependence graph* to capture such interdependence among nodes.¹

¹ These dependence relations are not necessarily the physical links in a network. For example, in a power system, an overload failure in one part of power grid can cause a failure in another part that is not geographically close or without direct physical connection to the former.

Of particular concern is a widespread outbreak of failures among nodes, which can risk a potentially catastrophic system-level failure. We refer to *a cascade of failures* or *cascading failures* (CFs) as an event in which the system experiences widespread failures well beyond the local neighborhood around the initial failure and, in the absence of remedial actions, the spread of failures slows down only when newly failed nodes no longer have other nodes they can cause to fail.

Recently, Yang et al. [13] examined cascading failures of transmission lines in power grids under different conditions. Key findings of their study include following observations: (i) only a small fraction of transmission lines are susceptible to overload failures, and (ii) initial failures close to such susceptible lines are much more likely to trigger widespread CFs.

In light of these findings, a critical challenge is how we can identify nodes whose failures are more likely to trigger CFs, which we call *vulnerable nodes*. Obviously, an answer to this question will help us avoid CFs in complex systems via remedial actions during both design and operational phases. We are especially interested in scenarios where insufficient data are available for finding such vulnerable nodes.

One might suspect that finding vulnerable nodes requires estimating the probability that a failed node will set off CFs, which we call its *cascading failure probability* (CFP). Computing the CFP of every node in a large system, for example, with the help of extensive simulation is a difficult and time-consuming task. To the best of our knowledge, there is no computationally efficient method of computing the CFPs; the current state-of-the-art approach relies on the tools available for (multi-type) branching processes under several simplifying assumptions, e.g., [4,5]. Therefore, an important open question is: *how can we identify vulnerable nodes in large systems in a computationally efficient manner, possibly without having to estimate their CFPs?* The goal of our study is to offer a partial answer to this question.

For this study, we focus on scenarios in which CFs do occur occasionally, but they are not frequent, i.e., the CFPs of nodes are small. However, unlike in our earlier studies [4,5], we are not interested in approximating the CFPs of the nodes (or of different types). Moreover, the previous studies fail to distinguish nodes of the same degree; when multi-type branching processes are used to model the spread of failures, only the degree distribution of dependence graph is used and other topological information regarding the dependence graph is lost. As a result, the CFPs of two nodes with the same degree are assumed to be identical. In this paper, we instead aim to identify nodes that are more likely to trigger CFs than others without explicitly computing CFPs.

We first show that there is a close relation between the vector consisting of nodes' CFPs and the leading eigenvector of the *weighted* non-backtracking matrix, where the weights are the probabilities with which failed nodes will cause neighbors to fail as well, which we call *infection probabilities*. This relation is similar to that of the non-backtracking centralities (NBCs) of nodes and Hashimoto or non-backtracking matrix illustrated in [6]. Furthermore, when the infection probabilities are homogeneous, we demonstrate that CFPs are propor-

tional to the NBCs (Theorem 1). As the NBCs can be obtained from the leading eigenvector of a matrix, this greatly simplifies the task of identifying vulnerable nodes. To the best of our knowledge, this is the first study that brings to light this interesting relationship between CFPs and node centralities.

A few words on notation: we will use boldface lowercase letters or symbols to denote (column) vectors or vector functions. For instance, \mathbf{d} denotes a vector, and the j th element of \mathbf{d} is denoted by d_j . Similarly, we use boldface uppercase letters to denote matrices. The set \mathbb{Z}_+ (resp. \mathbb{N}) denotes the set of nonnegative integers $\{0, 1, 2, \dots\}$ (resp. positive integers $\{1, 2, 3, \dots\}$).

2 System Model

We model the interdependence among nodes using a directed dependence graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$: the vertex set $\mathcal{V} = \{1, \dots, n\}$ consists of n nodes in the system. A directed edge from node i to node j , which is denoted by $\langle i, j \rangle$, indicates that node j can fail as a result of the failure of node i . In this sense, we say that node i *supports* node j .

Although not necessary, for the convenience of exposition, we assume that the dependence is mutual in that if $\langle i, j \rangle \in \mathcal{E}$, then $\langle j, i \rangle \in \mathcal{E}$. The mutual dependence between nodes i and j is denoted by an undirected edge (i, j) , and we say that they are (dependence) neighbors. Denote the set of neighbors and degree of node i by \mathcal{N}_i and $d_i := |\mathcal{N}_i|$, respectively, and let \mathbf{D} be an $n \times n$ diagonal matrix with $D_{i,i} = d_i$. Define $m := |\mathcal{E}|$ to be the number of directed edges in \mathcal{E} .

The connectivity between nodes via edges in \mathcal{E} can be represented with the help of a symmetric adjacency matrix $\mathbf{A} \in \{0, 1\}^{n \times n}$. The element $A_{i,j}$ in the i th row and the j th column indicates whether or not there is an edge from node i to node j : $A_{i,j} = 1$ if $\langle i, j \rangle \in \mathcal{E}$ and $A_{i,j} = 0$ otherwise. We adopt the convention $A_{i,i} = 0$ for all $i \in \mathcal{V}$, i.e., there are no self-loops.

2.1 Propagation of Failures Between Nodes

For our study, we adopt a simple model for capturing the spread of failures, in which a failed node causes each of its neighbors to fail with some fixed probability. To be more precise, the probability that node i , when it fails, will cause a neighbor $j \in \mathcal{N}_i$ to fail as well is given by $\beta_{i,j} \geq 0$. We call $\beta_{i,j}$ the infection probability of node j by node i , and denote the vector of all infection probabilities by $\boldsymbol{\beta} := (\beta_{i,j}; \langle i, j \rangle \in \mathcal{E})$.

We do not assume the symmetry of infection probabilities, i.e., $\beta_{i,j}$ and $\beta_{j,i}$ are allowed to be different. Thus, asymmetric connectivity between nodes can be handled by setting one of the infection probabilities to zero.

The infection probability $\beta_{i,j}$ can depend on many factors, including the degree of node j . These infection probabilities may be estimated, for example, using historical data or detailed simulation. For instance, Yang et al. [13] computed the line failure probabilities from detailed simulation and validated them

against historical data with respect to the distribution of cascade sizes. In the remainder of our study, we assume that the estimated infection probabilities are fixed and known.

An infection graph is defined to be a (random) subgraph of \mathcal{G} which describes the sequence of node failures: starting with an initial failure of a node, the infection graph consists of all failed nodes and the directed edges used to contribute to the failures of neighbors. In other words, an infection graph is a directed graph whose vertex set comprises failed nodes, and a directed edge $\langle i, j \rangle \in \mathcal{E}$ belongs to the infection graph if and only if node i fails first and impacts the failure of neighbor j .

2.2 Non-backtracking Centralities

Before we proceed, let us briefly discuss the NBC measure introduced in [6]. The NBCs were proposed to address the issue of localization of eigenvector centralities (ECs); ECs are given as the leading eigenvector of the adjacency matrix and, similarly to other notions of centrality, were proposed to capture the importance of nodes in a network. However, it has been pointed out by several studies [6, 12] that, as the network size increases with heterogeneous node degrees, ECs tend to concentrate on a small number of mostly high degree nodes or a densely connected subgraph, marginalizing the rest.

The authors of [6] argue that a primary reason for the localization of ECs, especially in the presence of high-degree hubs, is that a hub with large EC contributes to those of its neighbors that in turn reflect it back to the hub, in the process inflating the hub's EC. The NBC, by design, mitigates this issue: for each directed edge $\langle i, j \rangle \in \mathcal{E}$, define $v_{i \rightarrow j}$ to be the centrality of node i without the contribution from node j . If $\langle i, j \rangle \notin \mathcal{E}$, we set $v_{i \rightarrow j} = 0$. Then, the NBC of node i is given by

$$x_i = \sum_{j \in \mathcal{N}_i} v_{j \rightarrow i} = \sum_{j \in \mathcal{V}} A_{i,j} v_{j \rightarrow i}. \quad (1)$$

The NBC attempts to prevent the localization of centrality at hub nodes by avoiding double-counting the contribution generated by reflection between two neighbors; when computing the NBC of node i in accordance with (1), the contribution of node i to its neighbors is removed first.

From their definitions, it is clear that $\mathbf{v} := (v_{i \rightarrow j}; \langle i, j \rangle \in \mathcal{E})$ is the leading eigenvector of an $m \times m$ zero-one matrix \mathbf{B} with

$$B_{\langle k, l \rangle, \langle i, j \rangle} = \begin{cases} 1 & \text{if } j = k \text{ and } i \neq l, \\ 0 & \text{otherwise.} \end{cases} \quad (2)$$

Therefore, the NBCs of nodes, $\mathbf{x} := (x_i; i \in \mathcal{V})$, can be computed from (the leading eigenvector of) matrix \mathbf{B} .

It turns out that the NBC vector \mathbf{x} can be obtained without first computing \mathbf{v} : suppose that λ_{\max} is the leading eigenvalue of matrix \mathbf{B} , which is real from

Perron-Frobenius theorem, and define a $2n \times 2n$ matrix \mathbf{M} given by

$$\mathbf{M} = \begin{bmatrix} \mathbf{A} & \mathbf{I}_{n \times n} - \mathbf{D} \\ \mathbf{I}_{n \times n} & \mathbf{0}_{n \times n} \end{bmatrix}. \quad (3)$$

Then, the leading eigenvalue of \mathbf{M} is equal to λ_{\max} and the first n -elements of the associated eigenvector are equal to \mathbf{x} [6]. Since the number of nodes (n) is typically much smaller than the number of directed edges (m), computing the leading eigenvector of \mathbf{M} is computationally cheaper than computing that of \mathbf{B} .

3 Approximation of Cascading Failure Probabilities

Recall that we are interested in scenarios where the probability of suffering CFs, starting with a single failed node, is small regardless of the initial failed node. This in turn implies that, on the average, each failed node causes the failure of only a few neighbors. As a result, the infection graph is likely to be tree-like with few cycles. The study by Yang et al. [13] supports this assumption; even when cascading failures do occur, they found that the fraction of nodes involved in cascades tends to be small. For this reason, in the remainder of the paper, we assume that the infection graph is a directed tree.

For each $i \in \mathcal{V}$, define q_i to be node i 's CFP. Similarly, for each directed edge $\langle i, j \rangle \in \mathcal{E}$, we define c_{i-j} to be the probability that node i will trigger CFs without the failure of node j . This can be viewed as a proxy for the probability that node i will trigger CFs without the directed edge $\langle i, j \rangle$ under the assumption that the infection graph is a directed tree. In addition, under the same assumption, the probability that node i 's failure will not lead to CFs can be approximated using

$$1 - q_i = \prod_{j \in \mathcal{N}_i} (1 - \beta_{i,j} c_{j-i}). \quad (4)$$

Note that $\beta_{i,j} c_{j-i}$ is the probability that node i causes the failure of neighbor j and node j subsequently triggers CFs through its neighbors other than node i .

When $q_i \ll 1$, which implies $\beta_{i,j} c_{j-i} \ll 1$ for all $j \in \mathcal{N}_i$, we can approximate (4) using

$$1 - q_i = 1 - \sum_{j \in \mathcal{N}_i} \beta_{i,j} c_{j-i} + o\left(\sum_{j \in \mathcal{N}_i} \beta_{i,j} c_{j-i}\right) \approx 1 - \sum_{j \in \mathcal{N}_i} \beta_{i,j} c_{j-i}.$$

This suggests that we can approximate q_i using $\sum_{j \in \mathcal{N}_i} \beta_{i,j} c_{j-i}$.

Assumption 1. For all $i \in \mathcal{V}$, the CFP of node i , q_i , is given by

$$q_i = \sum_{j \in \mathcal{N}_i} \beta_{i,j} c_{j-i}. \quad (5)$$

4 Main Results

Define an $m \times m$ matrix \mathbf{B}^* , where

$$B_{\langle k,l \rangle, \langle i,j \rangle}^* = \begin{cases} \beta_{l,k} & \text{if } j = k \text{ and } i \neq l, \\ 0 & \text{otherwise.} \end{cases}$$

Note that this new matrix \mathbf{B}^* is similar to the Hashimoto matrix defined in (2); instead of binary entries, the entries in the row corresponding to the directed edge $\langle j, i \rangle$ are all equal to the infection probability $\beta_{i,j}$. Therefore, we can obtain \mathbf{B}^* as $\mathbf{B}^* = \text{diag}(\boldsymbol{\beta})\mathbf{B}$, where $\text{diag}(\boldsymbol{\beta})$ is a diagonal matrix whose diagonal elements are the infection probabilities $\beta_{i,j}$.

Recall that c_{j-i} is the probability that node j will set off CFs without causing the failure of neighbor i . In other words, at least one of neighbors of node j in $\mathcal{N}_j \setminus \{i\}$ will fail and lead to CFs. Therefore, if nodes i and j do not share a mutual neighbor or when the infection graphs are trees, following a similar argument used at the end of the previous section, we obtain

$$c_{j-i} \approx \sum_{k \in \mathcal{N}_j \setminus \{i\}} \beta_{j,k} c_{k-j}. \quad (6)$$

Although real networks are shown to exhibit higher clustering than random networks (for which the clustering coefficient approaches zero as the network size grows), the clustering coefficient for most real networks is not large [11]. Hence, for real networks with small clustering coefficients in which only a small fraction of nodes fail even when CFs do occur, (6) is likely a reasonable approximation. For this reason, we replace the approximation in (6) with an equality, i.e., we assume $c_{j-i} = \sum_{k \in \mathcal{N}_j \setminus \{i\}} \beta_{j,k} c_{k-j}$. For a reader who is interested in the effects of clustering on CFs, we refer to [5] and references therein.

It is clear from the definition of matrix \mathbf{B}^* and (6) that the vector $\mathbf{c} = (c_{j-i}; \langle j, i \rangle \in \mathcal{E})$ is the leading eigenvector of matrix \mathbf{B}^* . Hence, comparing (1) and (5) suggests a close relation between the NBCs and CFPs of nodes. In fact, the following subsection shows that, in the special case with homogeneous infection probability, they are identical up to a scale factor. As a result, the NBCs help us identify vulnerable nodes more prone to trigger CFs.

4.1 Homogeneous Infection Probability Case

In this subsection, we consider a special case where the infection probability $\beta_{i,j}$ is the same for all directed edges $\langle i, j \rangle$, i.e., $\beta_{i,j} = \beta \in (0, 1]$ for all $\langle i, j \rangle \in \mathcal{E}$. In many real world situations, especially during design and planning phases, we may not have sufficient data to obtain individual infection probabilities. In such cases, it may be reasonable to attempt to estimate a single infection probability β with the interpretation that when a node fails, the probability that it will cause the failure of a neighbor is approximated using β . In other words, when faced with shortage of available data, we pretend that the infection probability is homogeneous and settle for estimating a single quantity.

Let $\mathbf{q} := (q_i; i \in \mathcal{V})$ be the CFP vector. The following theorem illustrates the linear relationship between the CFP vector and the NBC vector. Its proof is a straightforward modification of the argument in [3] and is omitted here.

Theorem 1. *Suppose that the leading eigenvalue of matrix \mathbf{B}^* is λ_{\max}^* . Then, $\lambda_{\max}^* = \beta \lambda_{\max}$, where λ_{\max} is the leading eigenvalue of matrix \mathbf{B} (or, equivalently, \mathbf{M} in (3)), and \mathbf{q} is proportional to the first n elements of the leading eigenvector of matrix \mathbf{M} .*

Remark 1. As explained in Sect. 2.2, the NBC vector of nodes is given by the first n elements of the leading eigenvector of matrix \mathbf{M} . Therefore, Theorem 1 tells us that the CFP vector is a scaled NBC vector of a system with adjacency matrix \mathbf{A} . Also, as expected from the linear relation in (5), the common infection probability β affects only the associated eigenvalue of \mathbf{B}^* without altering the leading eigenvector of \mathbf{M} .

Remark 2. Although Theorem 1 is obtained under the assumption of homogeneous infection probability, it will likely provide a good picture of which nodes are vulnerable and more prone to trigger CFs when the infection probabilities do not vary significantly. In other words, when the infection probabilities are heterogeneous but have small variations, the NBCs, which can be computed efficiently from the sparse matrix \mathbf{M} , will facilitate identifying vulnerable nodes in a system.

5 Numerical Results

In this section, we present some numerical results. The goal of the numerical studies is two-fold: (i) to validate our main result in the previous section, and (ii) to investigate how the accuracy of the proposed approach varies with the network size, n , and the percentage of most vulnerable nodes we are interested in identifying. For our studies, we consider scale-free networks with 100, 200, 300, 400, 600, and 800 nodes.² Although we carried out similar numerical studies with Erdős-Rényi networks, due to a space constraint, we do not report their results. For each considered network size, we aim to identify the top 1, 2, 5, 10, or 20 percent of the most vulnerable nodes with the highest CFPs.

- **Scale-free networks** – For each $n \in \mathcal{N} := \{100, 200, 300, 400, 600, 800\}$, we choose the minimum degree of two and the maximum degree, d_{\max} , of $\lceil 3 \log(n) \rceil$, which is equal to 14, 16, 18, 18, 20, and 21, respectively. We select the minimum degree of two in order to ensure that the random network is connected with high probability. Furthermore, nodes with degree of one are less likely to be vulnerable nodes.

² It is shown that the degree distribution of many real networks can be approximated using a power law [1].

We choose the power law parameter of $\xi = 1.5$. In other words, the fraction of nodes with degree $d \in \{2, 3, \dots, d_{\max}\}$ is proportional to $d^{-1.5}$. For selected parameters, we first generate a degree sequence according to the given degree distribution, and then generate 100 random networks using the configuration model [7,8]. We first check the algebraic connectivity of each network to ensure that it is connected.

- **Approximation of CFPs using our approach** – For each generated network, we compute the leading eigenvector of matrix \mathbf{B}^* and, using equations (5) and (6), estimate nodes' CFPs. In this section, we call them non-backtracking centrality-based CFPs (NBCCFPs) of nodes. We normalize the NBCCFP vector so that its ℓ_1 norm is equal to one.

- **Estimation of CFPs** – In order to estimate the true CFP of a node, we cause the failure of the node and construct the random infection graph described in Sect. 2.1. Since the diameter of scale-free graphs is $O(\log(n))$ [2], which is known as a small world phenomenon, we declare that CFs occurred if the failures spread at least three hops away from the initially failed node. We repeat this 1,000 times for each node and take the fraction of times we see CFs as the CFP of the node.

- **Hit ratio** – We sort the estimated CFP vector (resp. the NBCCFP vector) by decreasing value and identify the K nodes with the K largest CFPs (resp. NBCCFPs). Denote the set of nodes with the K largest CFPs and that of the K largest NBCCFPs by \mathcal{V}_K^{CFP} and \mathcal{V}_K^{NBC} , respectively. Then, we compute the fraction of the K nodes in \mathcal{V}_K^{CFP} , which also belong to \mathcal{V}_K^{NBC} . In other words, we calculate

$$h_K := \frac{|\mathcal{V}_K^{NBC} \cap \mathcal{V}_K^{CFP}|}{K}.$$

Clearly, h_K lies in the unit interval $[0, 1]$ and is equal to one if and only if the two sets are identical. Throughout this section, we refer to h_K as the hit ratio (HR).

In real networks, we expect the infection probabilities of directed edges to vary for several reasons. For example, nodes that invest more in protecting themselves are less likely to suffer failures (property 1). Similarly, depending on how critical their functionality is in supporting the overall system and their neighbors, some nodes may be less likely to cause the failure of neighbors even when they fail (property 2). Therefore, the infection probabilities may vary significantly from one directed edge to another. In order to capture both properties 1 and 2 above, we generate the matrix of infection probabilities as follows:

Step 1: Generate two random vectors – one $n \times 1$ column vector, $\boldsymbol{\beta}^c$, and one $1 \times n$ row vector $\boldsymbol{\beta}^r$. The elements of the vectors are given by independent and identically distributed exponential random variables with mean 0.03.

Step 2: For a directed edge $\langle i, j \rangle$, its infection probability $\beta_{i,j}$ is set to be $\min(1, \max(0.01, \beta_i^c \times \beta_j^r))$. From their construction, it is clear that (i) the i th element of the column vector $\boldsymbol{\beta}^c$ influences the probability that node i will cause the failures of its neighbors when it fails, and (ii) the j th element of the row vector $\boldsymbol{\beta}^r$ shapes the probability that node j will fail when its neighbor does.

Table 1 provides the average HRs. Here, k is the percentage of the most vulnerable nodes we wish to identify, i.e., $k = K/n$. There are a few observations we note from the table. First, the proposed approach yields reasonably high hit ratios, even for small values of k . For a simple comparison, when two sets of K nodes are chosen from the vertex set \mathcal{V} of n nodes according to the uniform distribution over the set of all possible subsets of K nodes, the expected number of common nodes in the two sets is K^2/n . Equivalently, the expected fraction of common nodes or the HR is equal to $K/n = k$. Clearly, the HRs reported in Table 1 are much larger than k .

Table 1. Average HRs for scale-free networks generated by configuration model.

	$k = 0.01$	$k = 0.02$	$k = 0.05$	$k = 0.1$	$k = 0.2$
$n = 100$	0.340	0.385	0.512	0.572	0.631
$n = 200$	0.395	0.452	0.509	0.581	0.645
$n = 300$	0.399	0.463	0.535	0.604	0.656
$n = 400$	0.380	0.449	0.539	0.599	0.649
$n = 600$	0.447	0.476	0.558	0.611	0.657
$n = 800$	0.371	0.471	0.560	0.613	0.666

Second, the HR improves with increasing k . This is somewhat expected as explained earlier. Moreover, in spite of fluctuations, there is a slight tendency for the HR to improve as the network size n increases. We suspect that the improvement with increasing network size is in part due to the fact that the CFPs of the most vulnerable nodes tend to rise somewhat with the network size, making it somewhat easier to identify nodes with larger CFPs.

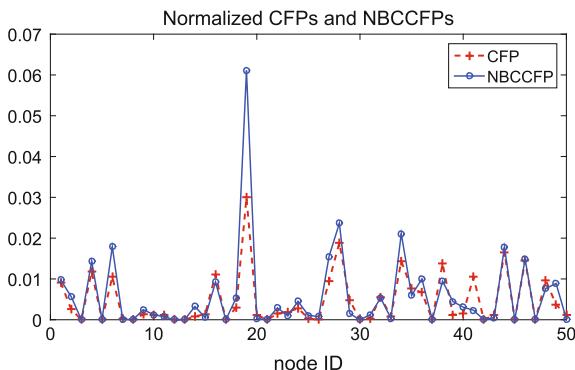


Fig. 1. CFPs and NBCCFPs of nodes 1 through 50 for one random network with $n = 400$. Both CFPs and NBCCFPs are normalized so that their ℓ_1 norms are equal to one.

For a closer look, we plot the normalized CFPs and NBCCFPs of 50 nodes with the largest degrees (ordered by decreasing degree) for one of random networks with $n = 400$ in Fig. 1. While they do not match perfectly as both are estimates of true CFPs, it is clear from the plots that they follow a similar trend and NBCCFPs do show potential to help us identify vulnerable nodes with larger CFPs.

Effects of degree correlations – The random networks generated by the configuration model are neutral in that there are no significant correlations between the degrees of end nodes of edges. But, many real networks exhibit non-negligible degree correlations (also known as network mixing or degree assortativity); it has been shown [9, 10] that engineered networks, e.g., the Internet, tend to be disassortative, whereas social networks are typically assortative. In other words, nodes in engineered systems tend to be connected to other nodes with dissimilar degrees, while those in social networks display a tendency to be neighbors with other nodes with similar degrees. Thus, it is of interest to examine the effects of degree correlations on the performance of the proposed approach.

The degree correlations of a network are often measured using the assortativity coefficient, which is the Pearson correlation coefficient of degrees of end nodes and lies in the interval $[-1, 1]$: let $\mathbf{p} := (p_d; d \in \mathbb{N})$ be the node degree distribution, i.e., p_d is the fraction of nodes with degree d . Similarly, define $\mathbf{p}^r := (p_d^r; d \in \mathbb{Z}_+)$ to be the distribution of the residual degree of a neighbor of a node, where p_d^r is given by $p_d^r = (d + 1)p_{d+1}/d_{\text{avg}}$ and $d_{\text{avg}} = \sum_{d' \in \mathbb{N}} d' p_{d'}$ is the average degree of nodes, and $\mathbf{e} := (e_{d,d'}; d, d' \in \mathbb{Z}_+)$ the joint distribution of the residual degrees of the end nodes of edges [9]. Then, the assortativity coefficient of the network is equal to

$$\gamma_{AC} = \frac{\sum_{d,d' \in \mathbb{Z}_+} d d' (e_{d,d'} - p_d^r p_{d'}^r)}{\sigma_r^2},$$

where $\sigma_r^2 := (\sum_{d \in \mathbb{Z}_+} d^2 p_d^r) - (\sum_{d \in \mathbb{Z}_+} d p_d^r)^2$ is the variance of the residual degree of a neighbor.

In order to generate an assortative or disassortative network, we order the nodes of the scale-free network generated using the configuration model by decreasing degree and rewire some of edges to increase or decrease degree correlations: (a) In order to generate an assortative network, starting with $i = 1$, for each edge (i, j) with $i < j$, we look for another edge (k, ℓ) with $\ell < j < k$. If there exists such an edge, then we rewire the two edges (i, j) and (k, ℓ) to new edges (i, ℓ) and (j, k) . Note that $d_\ell \geq d_j \geq d_k$ and we try to take two existing edges and rewire them to make the degrees of end nodes more similar, thereby increasing degree correlations. (b) For a disassortative network, at each iteration, we look for a pair of edges, one between two small-degree nodes and the other between two high-degree nodes. These two edges are then rewired so that the two new edges are between a high-degree node and a small-degree node.

Table 2. Average assortativity coefficients of 100 random networks (CM = configuration model).

	Neutral (CM)	Assortative	Disassortative
$n = 100$	-0.023	0.238	-0.269
$n = 200$	-0.018	0.261	-0.243
$n = 300$	-0.012	0.275	-0.228
$n = 400$	-0.007	0.276	-0.212
$n = 600$	-0.009	0.284	-0.188
$n = 800$	-0.004	0.285	-0.169

Table 2 shows the average assortativity coefficients of 100 neutral networks generated by the configuration model as well as those of assortative and disassortative networks produced through rewiring of edges as described above. As mentioned earlier, the configuration model produces random networks with negligible degree correlations. The rewired networks, on the other hand, exhibit significant degree correlations as intended.

Table 3. Average HRs for assortative networks.

	$k = 0.01$	$k = 0.02$	$k = 0.05$	$k = 0.1$	$k = 0.2$
$n = 100$	0.340	0.490	0.544	0.598	0.662
$n = 200$	0.430	0.448	0.537	0.599	0.642
$n = 300$	0.420	0.508	0.576	0.613	0.662
$n = 400$	0.412	0.492	0.559	0.603	0.655
$n = 600$	0.415	0.483	0.573	0.623	0.665
$n = 800$	0.444	0.518	0.589	0.624	0.675

Tables 3 and 4 provide the average HRs for the same values of n and k for assortative networks and disassortative networks, respectively. A key observation is that the degree correlations of networks do not affect the performance of proposed approach significantly; the average HRs do not diverge much from those of neutral networks shown in Table 1. Therefore, this suggests that the effectiveness of the proposed approach can be evaluated by considering only neutral networks without having to worry about the degree correlations of networks.

Table 4. Average HRs for disassortative networks.

	$k = 0.01$	$k = 0.02$	$k = 0.05$	$k = 0.1$	$k = 0.2$
$n = 100$	0.350	0.445	0.512	0.586	0.652
$n = 200$	0.410	0.482	0.535	0.579	0.640
$n = 300$	0.413	0.432	0.543	0.583	0.644
$n = 400$	0.445	0.482	0.529	0.583	0.635
$n = 600$	0.405	0.438	0.515	0.577	0.635
$n = 800$	0.424	0.437	0.535	0.589	0.643

6 Conclusion

We explored the problem of identifying vulnerable nodes in complex systems, which are more likely to set off CFs when they fail. We revealed an interesting linear relation between the CFPs and NBCs of nodes in case of homogeneous infection probability. When infection probabilities are heterogeneous, we showed that, under some assumptions, CFPs can be approximated with the help of a weighted Hashimoto matrix. Using numerical studies, we demonstrated that the proposed approach has the potential to facilitate finding vulnerable nodes without having to carry out time-consuming detailed simulation to compute CFPs.

Acknowledgement. This work was supported in part by contract 70NANB16H024 from National Institute of Standards and Technology (NIST).

References

- Albert, R., Jeong, H., Barabasi, A.L.: Error and attack tolerance of complex networks. *Nature* **406**, 378–382 (2000)
- Bollobas, B., Riordan, O.: The diameter of a scale-free random graph. *Combinatorica* **24**(1), 5–34 (2004)
- Krzakala, F., et al.: Spectral redemption: clustering sparse networks. *Proc. Natl. Acad. Sci.* **110**(52), 20935–20940 (2013)
- La, R.J.: Cascading failures in interdependent systems: impact of degree variability and dependence. *IEEE Trans. Netw. Sci. Eng.* **5**(2), 127–140 (2018)
- La, R.J.: Cascading failures in interdependent systems: impact of degree variability and dependence. *IEEE Trans. Netw. Sci. Eng.* (2018). <https://doi.org/10.1109/TNSE.2018.2805720>
- Martin, T., Zhang, X., Newman, M.E.J.: Localization and centrality in networks. *Phys. Rev. E* **90**(052808) (2014)
- Molloy, M., Reed, B.: A critical point for random graphs with a given degree sequence. *Random Struct. Algorithms* **6**(2–3), 161–180 (1995)
- Molloy, M., Reed, B.: The size of the largest component of a random graph on a fixed degree sequence. *Comb. Probab. Comput.* **7**(3), 295–305 (1998)

9. Newman, M.E.J.: Assortative mixing in networks. *Phys. Rev. Lett.* **89**(208701) (2002)
10. Newman, M.E.J.: Mixing patterns in networks. *Phys. Rev. E* **67**(026126) (2003)
11. Newman, M.E.J.: The structure and function of complex networks. *SIAM Rev.* **45**(2), 167–256 (2003)
12. Pastor-Satorras, R., Castellano, C.: Eigenvector localization in real networks and its implications for epidemic spreading. *J. Stat. Phys.* 1–14 (2018). <https://doi.org/10.1007/s10955-018-1970-8>
13. Yang, Y., Nishikawa, T., Motter, A.E.: Small vulnerable sets determine large network cascades in power grids. *Science* **358**(6365) (2017)



Computational Aspects of Fault Location and Resilience Problems for Interdependent Infrastructure Networks

Madhav V. Marathe¹, S. S. Ravi^{1,2(✉)}, Daniel J. Rosenkrantz²,
and Richard E. Stearns²

¹ University of Virginia, Charlottesville, VA 22904, USA

{mmarathe66,ssravi0}gmail.com

² University at Albany – SUNY, Albany, NY 012222, USA

{drosenkrantz,thestearns2}gmail.com

Abstract. Motivated by applications in diagnosing failures in complex infrastructure networks, we consider the configuration sequence completion problem (CSC) for networked systems. The goal of the CSC problem is to choose values for unknown entries in a specified sequence of configurations of a system so that the resulting sequence represents a valid trajectory of the system. This problem generalizes some known decision problems for dynamical systems. We present efficient algorithms for some versions of the CSC problem and computational intractability results for other versions.

1 Introduction

We study inverse problems for networked dynamical systems motivated by their applications in assessing the reliability and fault tolerance of inter-dependent critical societal infrastructures. A key feature of such systems is their network structure, i.e., individual agents/components interact only with a specified set of other agents/components. The networked structure implies that simple lumped models for analyzing cascading failures in such systems is unlikely to yield a correct understanding of the inherent faults in these systems. Here, we will concentrate on the analysis of dynamic/cascading faults.

Complex societal infrastructures exhibit cascading failures. Power systems, the Internet, traffic systems, and chemical plants among others seem to have tipping points where a single additional failure leads to a catastrophic system wide collapse. As pointed out in [21, 25], these systems have three common characteristics: (i) overloading of the remaining system when elements are removed from service, (ii) inherent robustness until the point of collapse, and (iii) the presence of protection systems. Each fault results in a cascading failure of the system once triggered. See [2, 3, 8, 9, 12, 17, 19, 22, 25, 27] for additional discussion and examples of this phenomenon in the context of power systems.

The issue of cascading failures becomes all the more complicated for interdependent critical infrastructures. There are many direct physical linkages among infrastructure networks, especially power, water (e.g., for cooling), and telecommunications (e.g., SCADA control systems) [17, 21]. It is well known that these linkages create interdependencies that may transform an apparently localized single-component failure into a cascading crisis. For example, a localized loss of electric power may interrupt the supply of natural gas to a generator, which might in turn prevent a pump from supplying cooling water to yet another component. Because different infrastructures are designed and managed by many different organizations, both public and private, and because service areas for each component overlap in complicated ways, it is often difficult to analyze all the possible failure modes. Simulation tools are being developed to address such physical interdependencies.

Problem Considered. We use a model called a synchronous dynamical system (SyDS) to represent a networked system. We present an informal description of a SyDS here; a formal description appears in Sect. 2. A SyDS is specified by an undirected graph $G(V, E)$, where $|V| = n$. Each node $v \in V$ has a state value from $\{0, 1\}$ which changes over time. Each node v also has a local function f_v . The inputs to f_v are the current state of v and those of its neighbors, and the output of f_v is the next state of v . The configuration \mathcal{C} of a system at time τ is the n -tuple where the i^{th} component of \mathcal{C} represents the state of node v_i at time τ , $1 \leq i \leq n$. If there is a one-step transition from a configuration \mathcal{C}' to a configuration \mathcal{C} , we say that \mathcal{C} is the **successor** of \mathcal{C}' , and \mathcal{C}' is the **predecessor** of \mathcal{C} . Since our focus is on deterministic systems, each configuration has a unique successor; however, a configuration may have zero¹ or more predecessors. Any configuration \mathcal{C} whose successor is \mathcal{C} itself is called a **fixed point**.

We use the phrase “uniform SyDS” to mean a SyDS in which all the node functions are the same. A simple example of a uniform SyDS is one in which each local function is the AND function. We refer to such a SyDS as an AND-SyDS and use a similar notation for other uniform SyDSs.

The **phase space** $\mathbb{P}_{\mathcal{S}}$ of a SyDS \mathcal{S} is a directed graph defined as follows. There is a node in $\mathbb{P}_{\mathcal{S}}$ for each configuration of \mathcal{S} . There is a directed edge from a node representing configuration \mathcal{C} to that representing configuration \mathcal{C}' if there is a one step transition of \mathcal{S} from \mathcal{C} to \mathcal{C}' . For a SyDS with n nodes, the number of nodes in the phase space is 2^n ; thus, the size of phase space is *exponential* in the size of a SyDS. Each node in the phase space has an outdegree of 1 (since our SyDS model is deterministic). Also, in the phase space, each fixed point of a SyDS is a self-loop and each GE configuration is a node of indegree zero.

To give an informal description of the problem considered in this paper, consider a sequence $Q = \langle \mathcal{C}_0, \mathcal{C}_1, \dots, \mathcal{C}_{t-1}, \mathcal{C}_t \rangle$ consisting of $t + 1$ configurations. Here, each configuration is assumed to be fully specified; that is, each entry in each of the configurations in the sequence is 0 or 1. Given such a sequence, if \mathcal{C}_{i+1} is the successor of \mathcal{C}_i , for $0 \leq i \leq t - 1$, the sequence specifies a directed path in the phase space (i.e., a valid trajectory) of a given SyDS. We consider

¹ Configurations with no predecessors are called **Garden of Eden** configurations.

a more general form of a configuration sequence, where configurations may be *partially specified*. Thus, each entry in a configuration may be a 0, 1 or x , where ‘ x ’ denotes an unknown (or “don’t care”) value. The focus of our work is on the problem of determining whether a 0 or 1 value can be assigned to each x entry in a sequence of partially specified configurations so that the resulting sequence of fully specified configurations represents a valid trajectory of the given SyDS (i.e., a directed path in the phase space of the SyDS). We call this the **Configuration Sequence Completion** (CSC) problem. A formal definition of CSC and a discussion on how it generalizes problems such as predecessor existence and fixed point existence for dynamical systems studied in the literature [6, 14, 18] are presented in Sect. 2.

Summary of Contributions. Our main contributions (listed below) are stated for SyDSs with undirected graphs. Our results hold for SyDSs with directed graphs and also for SyDSs wherein a node does not consider its own current state while computing the next state. Furthermore, multiple and layered networks can be easily accommodated by appropriately considering labeled (colored) edges and nodes and suitable local transition functions.

- (1) For the following classes of uniform SyDSs, namely AND-SyDSs, OR-SyDSs, XOR-SyDSs and XNOR-SyDSs, we show that the CSC problem can be solved efficiently even when the parameter t is specified as part of the problem instance. This result generalizes the corresponding efficient solvability results in [6, 18] for predecessor and t -predecessor existence problems.
- (2) When the time parameter t and the treewidth² of the underlying graph are *fixed* and each local function is r -symmetric³ for some *fixed* integer r , we show that the CSC problem is solvable efficiently using dynamic programming. This result extends the results in [6] where the 1-predecessor problem was considered assuming that the final configuration was completely specified.
- (3) In contrast to the result mentioned in Item (2) above, when the parameter t is part of the problem instance, we show that the CSC problem is **NP**-complete even when the treewidth of the underlying graph is *fixed* and each node computes an r -symmetric function for a fixed r .

Discussion. We refer the reader to [3, 9, 12] for discussions on how Boolean models over multi-networks can be used to model cascading failures in interdependent infrastructure networks.

The results discussed here along with earlier results in [6, 18] suggest that in general, solving inverse problems on general networks is unlikely to be computationally efficient. However, structural properties of networks can often be exploited to solve very general inverse problems. A key network structure that we believe helps is *treewidth*. It generalizes the notion of trees significantly. Nevertheless, unlike the impressive results in graph algorithms showing that several problems expressible in monadic second order logic can be solved efficiently on treewidth bounded graphs, inverse problems for dynamics on such networks require bounded treewidth as well as certain other restrictions on local dynam-

² See [7] for the definition of treewidth.

³ The definition of r -symmetric functions appears in Sect. 2.

ics to be efficiently solvable. Our results extend several earlier results in [6, 18] and provide a formal framework to rigorously study generalized predecessor (or “preimage”) problems that have applications in network reliability and fault location.

Examples of Analyses that can be Performed. We now present several examples of questions that can be studied using our mathematical framework. These examples are based on problems and issues brought up in [3, 9, 23, 24]. See [3] for a good survey on various fault models. The focus here is on *inverse problems* related to these fault models; they capture fault location, isolation and related issues. (i) Can a configuration, in which a specified set of important installations has failed, be reached starting from some initial operating configuration? (ii) How many initial configurations can give rise to a given (faulty) configuration? (A smaller number of such initial configurations is desirable since in such a case the diagnosis can be carried out more quickly. As an extreme case, if a given configuration has a unique initial configuration, then the diagnosis procedure is even quicker.) (iii) The robustness of a system is often quantified by a measure called **resilience**. Informally, resilience indicates the largest number of failures that a system can tolerate and continue to function satisfactorily. In the context of networked dynamical systems, when a failure is described by a configuration \mathcal{C} in which a subset of the nodes are malfunctioning, one can define the resilience with respect to \mathcal{C} as the largest number of node failures such that the system is guaranteed not to reach the configuration \mathcal{C} within a specified number of steps. This measure of resilience can be computed using appropriate linear constraints. (iv) *Fault location and isolation*: Our algorithms can be used to locate faults that lead to system wide failures. For example, given a system state \mathcal{C} , one wants to know if the failure of a particular node or a fixed set of nodes can lead to a system-level fault in less than t steps. As another example, consider the question of assessing if more than (or less than) a certain number of nodes failing could have caused a system to fail. This can also be captured through a linear constraint. (v) *Reliability*: Suppose we know that the failure of any k elements from a subset S of components can lead to a system-wide failure. We want to find out how many initial configurations can lead to such a state. Then one can add a linear constraint to capture this for the final configuration and count the number of initial configurations.

Related Work. Barrett et al. [6] studied of the predecessor existence problem for SyDSs. They addressed only the 1-step predecessor existence and assumed that the final configuration is completely specified. Kawachi et al. [18] studied the generalized predecessor problem for values of $t \geq 1$ assuming that the underlying graph is directed and that the final configuration is completely specified.

Recently, there has been substantial interest in assessing the reliability of inter-dependent critical infrastructures; see e.g., [1, 4, 11, 15, 17, 19, 22, 27]. Reliability of individual infrastructures has been investigated within the respective fields for quite some time now; see [15, 25] and the numerous references cited therein. Theoretical models and analytical results for studying issues related to vulnerability and criticality can be found in several papers including [8, 11, 15, 28].

Recent papers [2, 3, 6, 12, 20] have undertaken Boolean network based dynamical analysis of contagions in socio-technical networks.

There are essentially three progressively more realistic (and usually computationally more expensive) methods for assessing reliability: (i) static methods that usually view the system as a graph or fixed object and look for ways to *break or shatter* the graph by removing some of its components, (ii) a steady state analysis that usually takes the form of looking for flows in these graphs and analyzing the effects of removing one or more components on the resulting flow and (iii) a dynamical analysis which looks for faults and disturbances that are cascading and are inherently dynamic in nature. This paper is primarily concerned with the third approach. Although it is more realistic, there is usually a penalty for this approach in terms of computational costs. Our work is also related to the recent work of Haimes et al. and Lee et al. [15, 16, 26] on models and analysis of interconnected infrastructures, but differs in the following important way. SyDS-based models considered in our work are dynamical models; in contrast, much of the work in the above papers concentrates on steady state equilibrium analysis⁴. The work in [15, 16] by and large does not consider the computational complexity of their models. Crucially, they do not study the effect of network topology on computational tractability – our results for treewidth bounded graphs show that one can exploit certain kinds of inter-connected structures to develop *provably efficient* algorithms.

2 Preliminaries

Synchronous Dynamical Systems and Local Functions. We follow the presentation in [6] to discuss the basic definitions associated with discrete dynamical systems. Let \mathbb{B} denote the Boolean domain $\{0, 1\}$. A **Synchronous Dynamical System** (SyDS) \mathcal{S} over \mathbb{B} is specified as a pair $\mathcal{S} = (G, \mathcal{F})$, where (a) $G(V, E)$, an undirected graph with $|V| = n$, represents the underlying graph of the SyDS, with node set V and edge set E , and (b) $\mathcal{F} = \{f_1, f_2, \dots, f_n\}$ is a collection of functions in the system, with f_i denoting the **local function** associated with node v_i , $1 \leq i \leq n$. Each node of G has a state value from \mathbb{B} . Each function f_i specifies the local interaction between node v_i and its neighbors in G . The inputs to function f_i are the state of v_i and those of the neighbors of v_i in G ; function f_i maps each combination of inputs to a value in \mathbb{B} . This value becomes the next state of node v_i . It is assumed that each local function can be computed efficiently.

At any time τ , the **configuration** \mathcal{C} of a SyDS is the n -vector $(s_1^\tau, s_2^\tau, \dots, s_n^\tau)$, where $s_i^\tau \in \mathbb{B}$ is the state of node v_i at time τ ($1 \leq i \leq n$). Given a configuration \mathcal{C} , the state of a node v in \mathcal{C} is denoted by $\mathcal{C}(v)$. In a SyDS, all nodes compute and update their next state *synchronously*. Other update disciplines (e.g., sequential updates) have also been considered in the literature (e.g., [6]). If a given SyDS can transition in one step from a configuration \mathcal{C}' to a configuration \mathcal{C} , then \mathcal{C} is

⁴ Haimes and Jiang [16] propose a dynamical model similar to the one discussed here but do not discuss the computational aspects which is crucial.

a **successor** of \mathcal{C}' and \mathcal{C}' is a **predecessor** of \mathcal{C} . SyDSs have been considered in the literature under many classes of local interaction functions (see e.g., [5, 18]). We now present an example of a SyDS.

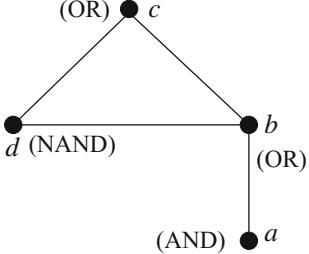


Fig. 1. An Example of a SyDS

initial configuration, the sequence of configurations that the system goes through is given by $(1, 1, 0, 0) \rightarrow (1, 1, 1, 1) \leftrightarrow (1, 1, 1, 0)$ where we have used “ \leftrightarrow ” to indicate that the system cycles through the last two configurations.

We now provide a formal definition of the configuration sequence completion (CSC) studied in this paper.

Configuration Sequence Completion (CSC)

Instance: A SyDS \mathcal{S} , integer t , and a sequence $Q = \langle \mathcal{C}_0, \mathcal{C}_1, \dots, \mathcal{C}_{t-1}, \mathcal{C}_t \rangle$ of $t+1$ partially specified configurations, where each entry in any configuration in Q is from $\{0, 1, \mathbf{x}\}$.

Question: Is there an assignment of 0 or 1 value to each \mathbf{x} entry in Q such that the resulting (fully specified) sequence of configurations represents a directed path in the phase space $\mathbb{P}_{\mathcal{S}}$ of \mathcal{S} ?

We now discuss how our formulation of CSC generalizes other problems considered in the literature. For example, given a configuration \mathcal{C} , the problem of determining whether \mathcal{C} has a predecessor has been studied in the context of discrete dynamical systems and cellular automata (see e.g., [6, 14]). This can be seen to be a special case of CSC by choosing the sequence Q to consist of two configurations $\langle \mathcal{C}', \mathcal{C} \rangle$, with \mathcal{C} being fully specified and making each entry of \mathcal{C}' to be \mathbf{x} . Further, instead of a one-step predecessor, a more general problem of obtaining a t -step predecessor for a given value of $t \geq 1$ has also been studied [18]. Given a configuration \mathcal{C} , a **t -step predecessor** of \mathcal{C} is a configuration \mathcal{C}_0 such that there is a sequence of t one-step transitions starting with \mathcal{C}_0 and ending with \mathcal{C} . The t -step predecessor problem can be represented as an instance of the CSC problem by making each entry of the configurations \mathcal{C}_0 through \mathcal{C}_{t-1} to have the value \mathbf{x} and the (fully specified) configuration \mathcal{C}_t to be the given configuration \mathcal{C} for which a t -predecessor must be found (if one exists). We note that the CSC problem is more general than the t -predecessor problem since the former allows some state values in the intermediate configurations to be specified as well. Also,

Example: Consider the graph of a SyDS shown in Fig. 1. The local function at each node is shown within parentheses. Assume that initially, nodes a and b are in state 1 while nodes c and d are in state 0; that is, the initial configuration \mathcal{C}_0 of the system is $(1, 1, 0, 0)$. At time 1, the state of a remains 1 since the local function is AND and its two inputs, namely the states of a and b are both 1. In a similar manner, it can be seen that the states of c and d change to 1 at time 1. At time step 2, the state of d changes to 0 since its local function is NAND and all the inputs to the function are 1. Using this line of reasoning, it can be seen that starting with the

the fixed point existence problem (FPE) considered in the literature can also be thought of as an instance of CSC with a specific constraint. Recall that in the FPE problem, we are given a SyDS \mathcal{S} and the goal is to determine whether \mathcal{S} has a fixed point. This problem can be represented by an instance of CSC along with a simple constraint as follows: the configuration sequence consists of two identical configurations, each of whose entries has the value \mathbf{x} , and the constraint is that for each node v , the $\{0, 1\}$ value assigned to v in the first and second configurations must be the same. We again note that CSC formulation allows us to consider versions of FPE where some of the nodes have pre-specified $\{0, 1\}$ values. In this paper, our focus is on the version of CSC without any constraints. A study of CSC with constraints is left for future work.

Additional Terminology. Here, we introduce some additional graph theoretic and Boolean function terminology that will be used throughout the paper.

Given a graph $G(V, E)$ and a node $v \in V$, the **closed neighborhood** of v , denoted by $N[v]$, consists of v and all its neighbors. The definitions of *tree decomposition* and *treewidth* are standard; they can be found in many references (e.g., [7]).

A **symmetric** Boolean function [10] is one whose value does not depend on the order in which the input bits are specified; that is, the function value depends only on how many of its inputs are 1. A Boolean function f is r -**symmetric** [6] if the inputs to f can be partitioned into at most r classes such that the value of f depends only on how many of the inputs in each of the r classes are 1. Note that any Boolean function with d inputs is d -symmetric. We will be mainly concerned with r -symmetric functions where r is a *fixed* integer. We say that a SyDS is r -symmetric if each of its local functions is r' -symmetric for some $r' \leq r$.

3 Results for Uniform SYDSs

In this section, we consider uniform SyDSs, that is, SyDSs where all the local functions are identical. Recall that when each local function is f , we use the notation f -SyDS to refer to the corresponding uniform SyDS. We use XOR to denote the “exclusive or” function and XNOR to denote the complement of XOR.

Theorem 1. *For uniform f -SyDSs, where f is from {AND, OR, XOR, XNOR}, the CSC problem can be solved efficiently even when the time parameter t is specified as part of the problem instance.*

Proof. Throughout this proof, we assume that the given sequence Q has $t + 1$ configurations denoted by $\mathcal{C}_0, \mathcal{C}_1, \dots, \mathcal{C}_t$. For any node v and configuration \mathcal{C} , we use $\mathcal{C}(v)$ to denote the state value of v in \mathcal{C} . Note that the value $\mathcal{C}(v)$ is from $\{0, 1, \mathbf{x}\}$.

Part 1 – AND-SyDSs: Here, we consider SyDSs in which each local function is AND. Our algorithm consists of two phases discussed below. During any of these phases, if there is any inconsistency (i.e., for some node v , the value assigned to

$\mathcal{C}_j(v)$ by these steps differs from the current value of $\mathcal{C}_j(v)$), the algorithm stops and reports that there is no solution to the CSC instance.

Phase 1: This phase consists of two iterative steps which may be carried out in any order. Each of these iterative steps changes a x value to 0 or 1. The correctness of the steps is a simple consequence of the fact that each node computes the AND function.

Step 1 (Forward Propagation of 0's): This step relies on the fact that if for some node v , $\mathcal{C}_i(v) = 0$, then for every node $w \in N[v]$ (the closed neighborhood of v), $\mathcal{C}_j(w) = 0$ for all \mathcal{C}_j , $i < j \leq t$.

Step 2 (Backward Propagation of 1's): This step relies on the fact that if for some node v , $\mathcal{C}_i(v) = 1$, then for every node $w \in N[v]$ (the closed neighborhood of v), $\mathcal{C}_j(w) = 1$ for all \mathcal{C}_j , $0 < j < i$.

Phase 2: After Phase 1 is completed (i.e., all forward and backward propagation steps have been completed), this phase sets all the remaining x values to 0. The correctness of this step can be readily verified.

It is easy to see that each of the phases runs in polynomial time.

Part 2 – OR-SyDSs: The algorithm for this case is similar to the one for AND-SyDSs except for the following: (i) The forward propagation step is for 1's, (ii) the forward propagation step is for 0's and (iii) Phase 2 sets all the x values to 1.

Part 3 – XOR-SyDSs: We obtain an algorithm for this case reducing the CSC instance to a system of linear equations over $\{0, 1\}$. It is well known that such a system of equations can be solved in polynomial time using standard methods such as Gaussian elimination [13]. To obtain these equations, we introduce a variable x_{ij} for the value of node v_j in configuration \mathcal{C}_i , $0 \leq i \leq t$ and $1 \leq j \leq n$. For each pair of configurations \mathcal{C}_{i-1} and \mathcal{C}_i and each node v_j , $1 \leq j \leq n$, we have the equation $x_{ij} = \bigoplus_{k \in N[v_j]} x_{i-1,k}$ where \oplus denotes the XOR function. This equation merely expresses how the local function at v_j computes the next state of v_j using the state values of the nodes in $N[v_j]$ in the previous configuration. When some of the values of the variables are known, the equations can be simplified by substituting those values. Since there are $t+1$ configurations and n values in each configuration, the total number of equations is $O(nt)$. It can be seen that there is a solution to the CSC instance iff there is a solution to the system of linear equations. It follows that the CSC problem can be solved efficiently for XOR-SyDSs.

Part 4 – XNOR-SyDSs: The algorithm for this case is similar to that for XOR-SyDSs, except that each equation uses XNOR instead of the XOR. \square

Additional Comments: It is known for NAND-SyDSs and NOR-SyDSs, CSC can be solved efficiently when $t = 1$. For these cases, we will show in a complete version of this paper that CSC becomes NP-hard for $t \geq 2$. For other classes of uniform SyDSs and many classes of non-uniform SyDSs (i.e., those where different nodes may have different local functions), NP-completeness of CSC follows from the corresponding results for the predecessor existence (PRE) problem. For example, it is known that the PRE problem is NP-complete when each node

computes the k -threshold function⁵ for any $k \geq 2$ [6]. Thus, the CSC problem is **NP**-complete for SyDSs where each node computes the k -threshold function for any $k \geq 2$. Likewise, the PRE problem is **NP**-complete for (non-uniform) SyDSs where each local function is from {AND, OR}. Thus, the CSC problem is also **NP**-complete for such SyDSs. We note that all of these hardness results hold even when t is fixed at 1.

4 Efficient Solvability for Sequences of Fixed Length

In this section, we consider SyDSs where the underlying graph is treewidth bounded and each local function is r -symmetric for some *fixed* integer r . We also assume that the value of t is *fixed*. The main result of this section is the following.

Theorem 2. *For any fixed trajectory length $t + 1$, symmetry bound r , and treewidth bound k , the CSC problem can be solved efficiently when the underlying graph of the given SyDS is treewidth bounded by k and each local function is r -symmetric.*

Proof. (Sketch): Let \mathcal{S} be a SyDS whose underlying graph $G(V, E)$ has a treewidth of k . It is well known that a tree decomposition $(\{X_i \mid i \in I\}, T = (I, F))$ of G can be constructed in time that is a polynomial in the size of G . Moreover, this can be done so that T is a binary tree; that is, each node of T has at most two children [7].

Our algorithm relies on definitions of **explicit nodes**, **inherited nodes**, **hidden nodes** and **originating nodes** in a tree decomposition presented in [6]. For space reasons, we will assume familiarity with these concepts.

Given a set of nodes $Y \subseteq V$, a Y -**configuration** \mathcal{C}_Y assigns a Boolean value to each node in Y . Given a configuration \mathcal{C} , we use $\mathcal{C}(y)$ to denote the value of node y in that configuration. Similarly, given a Y -configuration \mathcal{C}_Y and a node y in Y , we use $\mathcal{C}_Y(y)$ to denote the value of node y in \mathcal{C}_Y . We extend this notation to subsets of nodes as follows. Given a configuration \mathcal{C} and a set of nodes $W \subseteq V$, $\mathcal{C}(W)$ denotes the combination of values assigned to the nodes in W . Similarly, given a Y -configuration \mathcal{C}_Y and a subset of nodes $W \subseteq Y$, $\mathcal{C}_Y(W)$ denotes the combination of values assigned to the nodes in W .

Let $Q = \langle \mathcal{C}_0, \mathcal{C}_1, \dots, \mathcal{C}_{t-1}, \mathcal{C}_t \rangle$ be the $t+1$ partially specified configurations specified in the given instance of the CSC problem for \mathcal{S} . Given a set of nodes $Y \subseteq V$, a Y -**configuration ensemble** is a function h_Y that maps each j , $0 \leq j \leq t$, into a Y -configuration. We let \mathcal{H}_Y denote the set of all Y -configuration ensembles. Note that the cardinality of \mathcal{H}_Y is $2^{(t+1)s}$, where s is the number of nodes in Y . If t is fixed, and s is bounded by a constant, then the cardinality of \mathcal{H}_Y is also bounded by a constant. We say that a given Y -configuration ensemble h_Y is **conforming** if for every j , $0 \leq j \leq t$, and every node $v \in$

⁵ The value of the k -threshold function is 1 iff at least k of the inputs are 1 [10].

Y such that $\mathcal{C}_j(v)$ is either 0 or 1, $h_Y(j)(v) = \mathcal{C}_j(v)$. Consider a given Y -configuration \mathcal{C}_Y and a given Z -configuration \mathcal{C}_Z . We say that \mathcal{C}_Y and \mathcal{C}_Z are **consistent** if for every node x of $Y \cap Z$, it is the case that $\mathcal{C}_Y(x) = \mathcal{C}_Z(x)$. We say that a given Y -configuration ensemble h_Y and a given Z -configuration ensemble h_Z are **consistent** if for every j , $0 \leq j \leq t$, configurations $h_Y(j)$ and $h_Z(j)$ are consistent. For disjoint subsets Y and Z of V , suppose that h_Y is a Y -configuration ensemble and h_Z is a Z -configuration ensemble. We use the notation $h_Y \cup h_Z$ to denote the $(Y \cup Z)$ -configuration ensemble that maps each j , $0 \leq j \leq t$, into the $(Y \cup Z)$ -configuration $h_Y(j) \cup h_Z(j)$.

For simplicity, we subsequently consider the case where r is 1, so that all the local functions are symmetric. We refer to all the members of the closed neighborhood of a node v of V as generalized neighbors of v . For not necessarily disjoint sets of nodes Y and W of G , a (Y, W) -**signature** is a function $g : W \rightarrow \mathbb{N}$, such that for each w in W , the value of $g(w)$ does not exceed the number of generalized neighbors of w that are in Y . Given subsets Y and W of V , a (Y, W) -**signature ensemble** is a function that maps each j , $0 \leq j \leq t$, into a (Y, W) -signature. We let $\Gamma_{Y,W}$ denote the set of all possible (Y, W) -signatures, and let $\Gamma_{Y,W}^H$ denote the set of all possible (Y, W) -signature ensembles. Note that $\Gamma_{Y,W}^H$ is isomorphic to $(\Gamma_{Y,W})^{(t+1)}$. Also note that since t is fixed, if the cardinality of W is bounded, then the cardinality of $\Gamma_{Y,W}^H$ is bounded by a polynomial function of the the number of nodes in G .

To solve the CSC problem we use bottom-up dynamic programming on the decomposition tree to compute an appropriate table of signature ensembles at each node of the decomposition tree. Details will appear in a complete version of this paper.

□

5 Complexity When Sequence Length is Not Fixed

In this section, we consider the CSC problem under the same restrictions as the ones in Sect. 4, except that the value of t is *not fixed*. We show that, in contrast to the result of the previous section, this version of the CSC problem is **NP**-complete.

Theorem 3. *When t is not fixed and given in unary, the CSC problem is NP-complete even when the underlying graph of the given SyDS is treewidth bounded and each local function is r -symmetric for some fixed integer r .*

Proof. (Idea): Due to space restrictions, we will limit ourselves to the main idea behind the proof. We use a reduction from the well known **NP**-complete problem 3SAT [13]. Our proof uses an intermediate step that involves the construction of a 1-tape Turing Machine (TM) from the given 3SAT instance. We obtain an instance of the CSC problem by constructing a sequence of configurations which represent successive instantaneous descriptions⁶ (IDs) of the TM.

⁶ The instantaneous description of a 1-tape machine consists of the state of the TM, contents of the tape and the head position.

The reason for doing the reduction in this manner is that we can obtain a SyDS whose underlying graph has bounded pathwidth (and hence bounded treewidth) and whose local functions are all r -symmetric for some fixed r . The details will appear in a complete version of this paper.

□

6 Summary and Future Research Directions

Our results can be readily extended to the SyDS model with directed graphs considered in [18]. Our work also provides some new research directions. For example, it is of interest to consider the CSC problem for other uniform SyDSs. We observed that modeling fixed point existence problems through CSC involves constraints on the don't-care values. Identifying appropriate constraints that can model other problems for SyDSs and studying the CSC problem under those constraints is an interesting research direction.

Acknowledgments. We thank Professors Arun Phadke and late Jim Thorp (Virginia Tech) for discussions related to problems studied in this paper. We thank the referees for providing helpful comments. This work has been partially supported by DARPA Cooperative Agreement D17AC00003 (NGS2), DTRA CNIMS (Contract HDTRA1-11-D-0016-0001), NSF DIBBS Grant ACI-1443054, NSF BIG DATA Grant IIS-1633028 and NSF EAGER Grant CMMI-1745207.

References

1. Amin, M.: Toward secure and resilient interdependent infrastructures. *J. Infrastruct. Syst.* **8**(3), 67–75 (2002)
2. Banerjee, J., Basu, K., Sen, A.: Analysing robustness in intra-dependent and inter-dependent networks using a new model of interdependency. *Int. J. Crit. Infrastruct.* **14**(2), 156–181 (2018)
3. Banerjee, J., Das, A., Sen, A.: A survey of interdependency models for critical infrastructure networks. [arXiv:1702.05407](https://arxiv.org/abs/1702.05407) (2017)
4. Barrett, C., Eubank, S., Kumar, V.A., Marathe, M.V.: Understanding large scale social and infrastructure networks: a simulation based approach. *SIAM News* **37**(4), 1–5 (2004)
5. Barrett, C., Hunt III, H.B., Marathe, M.V., Ravi, S.S., Rosenkrantz, D.J., Stearns, R.E.: Modeling and analyzing social network dynamics using stochastic discrete graphical dynamical systems. *Theoret. Comput. Sci.* **412**(30), 3932–3946 (2011)
6. Barrett, C., Hunt III, H.B., Marathe, M.V., Ravi, S.S., Rosenkrantz, D.J., Stearns, R.E., Thakur, M.: Predecessor existence problems for finite discrete dynamical systems. *Theoret. Comput. Sci.* **386**(1), 3–37 (2007)
7. Bodlaender, H.: Treewidth: algorithmic techniques and results. In: Proceedings 22nd MFCS, pp. 29–36 (1997)
8. Brummitt, C.D., D’Souza, R.M., Leicht, E.A.: Suppressing cascades of load in interdependent networks. *Proc. Nat. Acad. Sci.* **109**(12), E680–E689 (2012)
9. Buldyrev, S.V., Parshani, R., Paul, G., Stanley, H.E., Havlin, S.: Catastrophic cascade of failures in interdependent networks. *Nature* **464** (2010)

10. Crama, Y., Hammer, P.: Boolean Functions: Theory, Algorithms, and Applications. Cambridge University Press, New York, NY (2011)
11. Dobson, I., Chen, J., Thorp, J., Carreras, B.A., Newman, D.E.: Examining criticality of blackouts in power system models with cascading events. In: Proceedings 35th Annual Hawaii International Conference on System Sciences (HICSS) (2002). 10 pages
12. Galbusera, L., Giannopoulos, G., Argyroudis, S., Kakderi, K.: A Boolean networks approach to modeling and resilience analysis of interdependent critical infrastructures. In: Computer-Aided Civil and Infrastructure Engineering, pp. 1–15 (2018)
13. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-completeness. W. H. Freeman & Co., San Francisco (1979)
14. Green, F.: NP-complete problems in cellular automata. *Complex Syst.* **1**(3), 453–474 (1987)
15. Haimes, Y.Y., Horowitz, B.M., Lambert, J.H., Santos, J.R., Lian, C., Crowther, K.G.: Inoperability input-output model for interdependent infrastructure sectors I: theory and methodology. *J. Infrastruct. Syst.* **11**(2), 67–79 (2005)
16. Haimes, Y.Y., Jiang, P.: Leontief-based model of risk in complex interconnected infrastructures. *J. Infrastruct. Syst.* **7**(1), 1–12 (2001)
17. Horowitz, S.H., Phadke, A.G.: Boosting immunity to blackouts. *IEEE Power Energy Mag* **99**(5), 47–53 (2003)
18. Kawachi, A., Ogihara, M., Uchizawa, K.: Generalized predecessor existence problems for boolean finite dynamical systems. In: 42nd International Symposium on Mathematical Foundations of Computer Science (MFCS 2017), pp. 8:1–8:13 (2017)
19. Little, R.G.: Controlling cascading failure: understanding the vulnerabilities of interconnected infrastructures. *J. Urban Technol.* **9**(1), 109–123 (2002)
20. Parandehgheibi, M., Modiano, E.: Robustness of interdependent networks: the case of communication networks and the power grid. In: Global Communications Conference (GLOBECOM), 2013 IEEE, pp. 2164–2169. IEEE (2013)
21. Phadke, A., Thorp, J.S.: Expose hidden failures to prevent cascading outages [in power systems]. *IEEE Comput. Appl. Power* **9**(3), 20–23 (1996)
22. Rinaldi, S.M., Peerenboom, J.P., Kelly, T.K.: Identifying, understanding, and analyzing critical infrastructure interdependencies. *IEEE Control Syst.* **21**(6), 11–25 (2001)
23. Rosato, V., Issacharoff, L., Tiriticco, F., Meloni, S., Porcellinis, S., Setola, R.: Modelling interdependent infrastructures using interacting dynamical models. *Int. J. Crit. Infrastruct.* **4**(1–2), 63–79 (2008)
24. Shao, J., Buldyrev, S.V., Havlin, S., Stanley, H.E.: Cascade of failures in coupled network systems with multiple support-dependence relations. *Phys. Rev. E* **83**(3), 036,116 (2011)
25. Tamronglak, S., Horowitz, S., Phadke, A., Thorp, J.: Anatomy of power system blackouts: preventive relaying strategies. *IEEE Trans. Power Deliv.* **11**(2), 708–715 (1996)
26. Vespignani, A.: The fragility of interdependency. *Nature* **464**, 984–985 (2010)
27. Wallace, W.A., Mendonça, D., Lee, E., Mitchell, J., Chow, J.: Managing disruptions to critical interdependent infrastructures in the context of the 2001 World Trade Center attack. In: Impacts of and Human Response to the September 11, 2001 Disasters: What Research Tells Us (2001)
28. Wang, W., Yang, S., Hu, F., Stanley, H.E., He, S., Shi, M.: An approach for cascading effects within critical infrastructure systems. *Phys. A Stat. Mech. Appl.* **510**, 164–177 (2018)

Author Index

A

- Adiga, Abhijin, 524
Ajwani, Deepak, 28
Aleskerov, Fuad, 94
Alghamdi, Elham, 316
An, Chuanhai, 15
Anand, Avishek, 104
Ando, Keiya, 798
Aparício, David, 143
Arora, Shiraj, 731
Arora, Viplove, 743
Arquam, Md, 418

B

- Babu, Suresh, 841
Balev, Stefan, 769
Bary, Sophie, 130
Bean, Nigel, 304
Benbadis, Farid, 118
Bernard, Jocelyn, 578
Berthouze, Luc, 718
Bertier, Clément, 118
Bishop, A., 376
Bocharov, Pavel, 553
Boekhout, Hanjo D., 565
Bothorel, Cécile, 183
Bouthinon, Dominique, 130
Bouvry, Pascal, 171
Brown, Violet, 66
Brust, Matthias R., 171

C

- Caldarelli, Guido, 820
Campedelli, Gian Maria, 292
Cann, Tristan J. B., 267

- Carley, Kathleen M., 292
Cazabet, Rémy, 81
Centeno, Virgilio, 155
Chen, Andi, 652
Chen, Guanrong, 470
Chen, Tzu-Yi, 652
Chen, Xi, 66
Cherifi, Hocine, 782
Cimini, Giulio, 820
Conan, Vania, 118
Connes, Victor, 256
Cowan, Sarah K., 694
Cruickshank, Iain, 292
Cummings, Kayla S., 652

D

- Danoy, Grégoire, 171
Dao, Vinh-Loc, 183
Dey, Kuntal, 353
Dias de Amorim, Marcelo, 118
Duan, Zhisheng, 470
Dubaniowski, Mateusz Iwo, 482
Dugué, Nicolas, 256
Dutot, Antoine, 769

E

- Eeti, 491
El Faouzi, Nour-Eddin, 52
El Hassouni, Mohammed, 782
Eubank, Stephen, 524, 537
Evans, John, 445

F

- Faustino, Josemar, 328
Feng, Yuting, 470

Fernandes, Sara, 505
 Fiscarelli, Antonio Maria, 171
 Furno, Angelo, 52
 Fushimi, Takayasu, 3

G

Gama, João, 512
 Gawrychowski, Paweł, 28
 Gläser, Jochen, 219
 Glonek, Max, 304
 Goryashko, Alexander, 553
 Grácio, Clara, 505
 Greene, Derek, 316
 Guille, Adrien, 256
 Gupta, Shubham, 353

H

Hariri, Salim, 578
 Hartman, Ryan, 328
 Havemann, Frank, 219
 Hecking, Tobias, 195
 Hedayati, Maryam, 66
 Heidari, Farzaneh, 457
 Heinemann, Hans R., 482
 Heinz, Michael, 219
 Helling, Thomas J., 244
 Henry Tsang, K. Y., 854
 Holzmann, Helge, 104
 House, T., 376
 Huntsman, Steve, 433

I

Ikeda, Tetsuo, 3

J

Jain, Abhishek, 731
 Jiang, Wenxin, 280
 Jones, Timothy, 694
 Jurek-Loughrey, Anna, 405

K

Kanaan, Mohamad, 603
 Kaperick, Bryan, 524
 Karlsen, Matthew R., 828
 Kaushik, Saroj, 353
 Kazama, Kazuhiro, 3
 Kheddouci, Hamamache, 578, 603
 Khosla, Megha, 104
 Kiss, István Z., 365, 376, 718
 Kosters, Walter A., 565
 Kralj, Jan, 757

L

La, Richard J., 866
 Lamba, Hemank, 353
 Lavrač, Nada, 757
 Lenca, Philippe, 183
 Li, Bo, 854
 Liben-Nowell, David, 66
 Lopes, Luís Mário, 505
 Lytvyniuk, Kateryna, 405

M

Malliaros, Fragkiskos D., 392
 Marathe, Achla, 524
 Marathe, Madhav V., 155, 524, 879
 Marik, Radek, 207
 Menezes, Ronaldo, 328
 Metzler, Saskia, 680
 Meyer, Ulrich, 28
 Meyur, Rounak, 155
 Michael Wong, K. Y., 854
 Miettinen, Pauli, 680
 Miller, Joel C., 365
 Mitchell, Lewis, 304
 Moschoyiannis, Sotiris K., 828
 Mourchid, Youssef, 782

N

Nagar, Seema, 353
 Nakamura, Yasuyuki, 798
 Narayan, Onuttom, 41
 Nath, Madhurima, 524, 537
 Neal, Janie L., 652

O

O’Malley, A. James, 15
 Olivier, Damien, 769
 Overbury, Peter, 718

P

Pal, Anamitra, 155
 Paliouras, Georgios, 338
 Panagopoulos, George, 392
 Panduranga Rao, M. V., 731
 Papagelis, Manos, 457
 Papatheodorou, Dimitrios, 338
 Pinheiro, Diego, 328
 Pizzuti, Clara, 807
 Poulin, Valérie, 231

R

Ramesh, Yenda, 731
 Rapisardi, Giacomo, 820

- Ravi, S. S., 879
Reinert, Gesine, 590
Ren, Yihui, 537
Renoust, Benjamin, 782
Rockmore, Daniel N., 15
Rosenkrantz, Daniel J., 879
- S**
- Sahai, Swapnil, 694
Saito, Kazumi, 3
Samokhin, Leonid, 553
Sanjeev, Iraj, 41
Santini, Guillaume, 130
Sarantopoulos, Ilias, 338
Saucan, Emil, 706
Schickendantz, Alexander, 28
Scholtes, Johannes C., 244
Seyednezhad, S. M. Mahdi, 328
Shao, Sicong, 578
Sharma, Rajesh, 52, 405, 418, 491
Shvydun, Sergey, 94
Sikes, Camden, 66
Silva, Fernando, 143
Silva, Jorge, 143
Simon, Péter L., 365
Singh, Anurag, 418, 491
Sinha, Aakash, 81
Škrlj, Blaž, 757
Socievole, Annalisa, 807
Soldano, Henry, 130
Stearns, Richard E., 879
Strand, Julia, 66
- T**
- Tabassum, Shazia, 512
Tagawa, Takahiro, 798
Takes, Frank W., 244, 565
Théberge, François, 231

- Tirico, Michele, 769
Treuer, Jan, 619, 635, 663
Tuke, Jonathan, 304
Tunc, Cihan, 578
Tzortzis, Grigoris, 338
- U**
- Ulrich Hoppe, H., 195
- V**
- Vaudaine, Rémi, 81
Vazirgiannis, Michalis, 392
Venkatramanan, Srinivasan, 524
Ventresca, Mario, 743
Vogiatzis, Dimitrios, 338
Vullikanti, Anil, 155

- W**
- Ward, Jonathan A., 445
Weaver, Iain S., 267
Weber, Melanie, 706
Williams, Hywel T. P., 267
Wilson, Tegan, 66
- X**
- Xu, Xiaochuan, 590
- Y**
- Yadav, Gitanjali, 841
Yasutake, Koichi, 798
Youssef, Mina, 155

- Z**
- Zheng, Tian, 694
Zhu, Ruimin, 280
Zikmund, Tomas, 207
Zimeo, Eugenio, 52