

Commenced Publication in 1973

Founding and Former Series Editors:

Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen

Editorial Board

David Hutchison

Lancaster University, UK

Takeo Kanade

Carnegie Mellon University, Pittsburgh, PA, USA

Josef Kittler

University of Surrey, Guildford, UK

Jon M. Kleinberg

Cornell University, Ithaca, NY, USA

Alfred Kobsa

University of California, Irvine, CA, USA

Friedemann Mattern

ETH Zurich, Switzerland

John C. Mitchell

Stanford University, CA, USA

Moni Naor

Weizmann Institute of Science, Rehovot, Israel

Oscar Nierstrasz

University of Bern, Switzerland

C. Pandu Rangan

Indian Institute of Technology, Madras, India

Bernhard Steffen

University of Dortmund, Germany

Madhu Sudan

Massachusetts Institute of Technology, MA, USA

Demetri Terzopoulos

University of California, Los Angeles, CA, USA

Doug Tygar

University of California, Berkeley, CA, USA

Gerhard Weikum

Max-Planck Institute of Computer Science, Saarbruecken, Germany

Marco Dorigo Mauro Birattari
Christian Blum Maurice Clerc
Thomas Stützle Alan F.T. Winfield (Eds.)

Ant Colony Optimization and Swarm Intelligence

6th International Conference, ANTS 2008
Brussels, Belgium, September 22-24, 2008
Proceedings



Springer

Volume Editors

Marco Dorigo

IRIDIA, CoDE, Université Libre de Bruxelles
Avenue F. Roosevelt 50, CP 194/6, 1050 Brussels, Belgium
E-mail: mdorigo@ulb.ac.be

Mauro Birattari

IRIDIA, CoDE, Université Libre de Bruxelles
Avenue F. Roosevelt 50, CP 194/6, 1050 Brussels, Belgium
E-mail: mbiro@ulb.ac.be

Christian Blum

ALBCOM, LSI, Universitat Politècnica de Catalunya
Jordi Girona 1-3, Omega 112 Campus Nord, 08034 Barcelona, Spain
E-mail: cblum@lsi.upc.edu

Maurice Clerc

204 Route de la Nerulaz, 74570 Groisy, France
E-mail: Maurice.Clerc@WriteMe.com

Thomas Stützle

IRIDIA, CoDE, Université Libre de Bruxelles
Avenue F. Roosevelt 50, CP 194/6, 1050 Brussels, Belgium
E-mail: stuetzle@ulb.ac.be

Alan F.T. Winfield

Bristol Robotics Laboratory, University of the West of England
Coldharbour Lane, Bristol BS16 1QY, UK
E-mail: Alan.Winfield@uwe.ac.uk

Library of Congress Control Number: 2008934397

CR Subject Classification (1998): F.2.2, F.1.1, G.1, G.2, I.2, C.2.4, I.1

LNCS Sublibrary: SL 1 – Theoretical Computer Science and General Issues

ISSN 0302-9743

ISBN-10 3-540-87526-3 Springer Berlin Heidelberg New York

ISBN-13 978-3-540-87526-0 Springer Berlin Heidelberg New York

This work is subject to copyright. All rights are reserved, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, re-use of illustrations, recitation, broadcasting, reproduction on microfilms or in any other way, and storage in data banks. Duplication of this publication or parts thereof is permitted only under the provisions of the German Copyright Law of September 9, 1965, in its current version, and permission for use must always be obtained from Springer. Violations are liable to prosecution under the German Copyright Law.

Springer is a part of Springer Science+Business Media

springer.com

© Springer-Verlag Berlin Heidelberg 2008
Printed in Germany

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 12519514 06/3180 5 4 3 2 1 0

Preface

The series of biannual international conferences “ANTS – International Conference on Ant Colony Optimization and Swarm Intelligence”, now in its sixth edition, was started ten years ago, with the organization of ANTS’98. As some readers might recall, the first edition of ANTS was titled “ANTS’98 – From Ant Colonies to Artificial Ants: First International Workshop on Ant Colony Optimization.” In fact, at that time the focus was mainly on ant colony optimization (ACO), the first swarm intelligence algorithm to go beyond a pure scientific interest and to enter the realm of real-world applications.

Interestingly, in the ten years after the first edition there has been a growing interest not only for ACO, but for a number of other studies that belong more generally to the area of swarm intelligence. The rapid growth of the swarm intelligence field is attested by a number of indicators. First, the number of submissions and participants to the ANTS conferences has steadily increased over the years. Second, a number of international conferences in computational intelligence and related disciplines organize workshops on subjects such as swarm intelligence, ant algorithms, ant colony optimization, and particle swarm optimization. Third, IEEE started organizing, in 2003, the IEEE Swarm Intelligence Symposium (in order to maintain unity in this growing field, we are currently establishing a cooperation agreement between IEEE SIS and ANTS so as to have IEEE SIS in odd years and ANTS in even years). Last, the *Swarm Intelligence*¹ journal was born.

Continuing a tradition started in 2002 with the third edition of ANTS, in 2008 the proceedings of the conference are also published in the Springer LNCS series.

The current edition of the proceedings contains 17 full-length papers and 24 short papers. These were selected out of 91 submissions, which means that 45% of the submitted papers were accepted for publication. In addition to this, 10 submissions were accepted as extended abstracts: these represent potentially interesting research that is still in its initial stage.

All the contributions to the proceedings were presented at the conference in the form of poster presentations. This choice was intended to promote discussion among the participants. Additionally, there were a few oral presentation, chosen among the full-length papers.

Finally, we would like to thank all the people who helped in organizing ANTS 2008. We are very grateful to the authors who submitted their works; to the members of the international Program Committee and to the additional referees for their detailed reviews; to the people of IRIDIA for their enthusiasm in helping with organizational matters; to the Université Libre de Bruxelles for providing rooms and logistic support; and, more generally, to all those contributing

¹ See <http://www.springer.com/11721>

to the organization of the conference. We would like to also thank our sponsors: the IEEE Computational Intelligence Society for technical co-sponsorship, the company AntOptima, the Belgian Fund for Scientific Research–FNRS, and the French community of Belgium, for their financial support.

June 2008

Marco Dorigo
Mauro Birattari
Christian Blum
Maurice Clerc
Thomas Stützle
Alan F.T. Winfield

Organization

ANTS 2008 was organized by IRIDIA, Université Libre de Bruxelles, Belgium.

Conference Chair

Marco Dorigo IRIDIA, Université Libre de Bruxelles, Belgium

Technical Program Chairs

Christian Blum
Universitat Politècnica de Catalunya, Spain
Maurice Clerc
Independent Consultant on Optimisation,
France
Alan F.T. Winfield
University of the West of England, Bristol, UK

Publication Chair

Mauro Birattari IRIDIA, Université Libre de Bruxelles, Belgium

Publicity Chair

Thomas Stützle IRIDIA, Université Libre de Bruxelles, Belgium

Program Committee

Ashraf Abdelbar	American University in Cairo, Egypt
Carl Anderson	Archimedes, Inc., USA
Payman Arabshahi	University of Washington, USA
Bart Baesens	Katholieke Universiteit Leuven, Belgium
Thomas Bartz-Beielstein	Fachhochschule Köln, Germany
Leonora Bianchi	USI-SUPSI, Switzerland
Tim Blackwell	University of London, UK
Jürgen Branke	Universität Karlsruhe (TH), Germany
Marco Chiarandini	University of Southern Denmark, Denmark
Carlos Coello Coello	CINVESTAV-IPN, Mexico
Oscar Cordón	European Centre for Soft Computing, Spain
Carlos Cotta	Universidad de Málaga, Spain
Xiaohui Cui	Oak Ridge National Laboratory, USA
Gianni Di Caro	USI-SUPSI, Switzerland
Karl Doerner	Universität Wien & Salzburg Research, Austria
Hai-Bin Duan	Beihang University, China

VIII Organization

Andries P. Engelbrecht	University of Pretoria, South Africa
Mudassar Farooq	NUCES, Pakistan
Dario Floreano	EPFL, Switzerland
Alex Freitas	University of Kent, UK
Luca Gambardella	USI-SUPSI, Switzerland
Deborah Gordon	Stanford University, USA
Roderich Gross	EPFL, Switzerland
Walter Gutjahr	Universität Wien, Austria
Julia Handl	University of Manchester, UK
Richard Hartl	Universität Wien, Austria
Beat Hirsbrunner	University of Fribourg, Switzerland
Colin Johnson	University of Kent, UK
James Kennedy	Bureau of Labor Statistics, USA
Franziska Klügl	Universität Würzburg, Germany
Joshua Knowles	University of Manchester, UK
William B. Langdon	University College London, UK
Kristina Lerman	University of Southern California, USA
Vittorio Maniezzo	Università di Bologna, Italy
David Martens	Katholieke Universiteit Leuven, Belgium
Alcherio Martinoli	EPFL, Switzerland
Monaldo Mastrolilli	USI-SUPSI, Switzerland
Ronaldo Menezes	Florida Institute of Technology, USA
Daniel Merkle	Universität Leipzig, Germany
Bernd Meyer	Monash University, Australia
Martin Middendorf	Universität Leipzig, Germany
Francesco Mondada	EPFL, Switzerland
Nicolas Monmarché	Université de Tours, France
Frank Neumann	Max-Planck-Institut für Informatik, Germany
Ann Nowé	Vrije Universiteit Brussel, Belgium
Luis Paquete	Universidade de Coimbra, Portugal
Rafael Stubs Parpinelli	Universidade do Estado de Santa Catarina, Brazil
Konstantinos Parsopoulos	University of Patras, Greece
Riccardo Poli	University of Essex, UK
Marc Reimann	University of Warwick, UK
Andrea Roli	Università di Bologna, Italy
Martin Roth	Google, UK
Ruben Ruiz	Universidad Politécnica de Valencia, Spain
Erol Sahin	Middle East Technical University, Turkey
Michael Sampels	Université Libre de Bruxelles, Belgium
Giovanni Sebastiani	Ist. Applicazioni del Calcolo "Mauro Picone", Italy
Yuhui Shi	Xi'an Jiaotong-Liverpool University, China
Christine Solnon	Université Claude Bernard, France
William M. Spears	University of Wyoming, USA
Kasper Støy	University of Southern Denmark, Denmark

Ponnuthurai Suganthan	Nanyang Technological University, Singapore
David Sumpter	Uppsala University, Sweden
Fatih Tasgetiren	Sultan Qaboos University, Oman
Guy Théraulaz	Université Paul Sabatier, France
Vito Trianni	Ist. Scienze e Tecnologie della Cognizione, Italy
Richard T. Vaughan	Simon Fraser University, Canada
Michael N. Vrahatis	University of Patras, Greece
Carsten Witt	Universität Dortmund, Germany
Jun Zhang	Sun Yat-sen University, China

Local Arrangements

Marco Montes de Oca	IRIDIA, Université Libre de Bruxelles, Belgium
Carlotta Piscopo	IRIDIA, Université Libre de Bruxelles, Belgium

Additional Referees

Prasanna Balaprakash	Giacomo Di Tollo	Rehan O'Grady
Arne Brutschy	Jens Gimmeler	Steffen Wolf
Alexandre Campo	Amin Mantrach	Zhi Yuan

Sponsoring Institutions

AntOptima, Lugano, Switzerland http://www.antoptima.com
Fund for Scientific Research–FNRS, Belgium http://www.fnrs.be
French Community of Belgium (through the research project ANTS) http://www.cfwb.be
IEEE Computational Intelligence Society (as a technical co-sponsor) http://www.ieee-cis.org

Table of Contents

A Combined Ant Colony and Differential Evolution Feature Selection Algorithm.....	1
<i>Rami N. Khushaba, Ahmed Al-Ani, Akram AlSukker, and Adel Al-Jumaily</i>	
An Improved ACO Based Plug-in to Enhance the Interpretability of Fuzzy Rule Bases with Exceptions	13
<i>Pablo Carmona and Juan Luis Castro</i>	
Ant Colony Optimization for Energy-Efficient Broadcasting in Ad-Hoc Networks	25
<i>Hugo Hernández, Christian Blum, and Guillem Francès</i>	
Ant Colony Optimization for Genome-Wide Genetic Analysis	37
<i>Casey S. Greene, Bill C. White, and Jason H. Moore</i>	
cAnt-Miner: An Ant Colony Classification Algorithm to Cope with Continuous Attributes	48
<i>Fernando E.B. Otero, Alex A. Freitas, and Colin G. Johnson</i>	
Finding Minimum Spanning/Distances Trees by Using River Formation Dynamics	60
<i>Pablo Rabanal, Ismael Rodríguez, and Fernando Rubio</i>	
Gathering Multiple Robotic Agents with Crude Distance Sensing Capabilities	72
<i>Noam Gordon, Yotam Elor, and Alfred M. Bruckstein</i>	
Integration of ACO in a Constraint Programming Language	84
<i>Madjid Khichane, Patrick Albert, and Christine Solnon</i>	
Learning from House-Hunting Ants: Collective Decision-Making in Organic Computing Systems	96
<i>Arne Brutschy, Alexander Scheidler, Daniel Merkle, and Martin Middendorf</i>	
Modeling Phase Transition in Self-organized Mobile Robot Flocks	108
<i>Ali Emre Turgut, Cristián Huepe, Hande Çelikkanat, Fatih Gökçe, and Erol Şahin</i>	
Molecular Structure Elucidation Using Ant Colony Optimization: A Preliminary Study	120
<i>Caroline Farrelly, Douglas B. Kell, and Joshua Knowles</i>	

Rigorous Analyses for the Combination of Ant Colony Optimization and Local Search	132
<i>Frank Neumann, Dirk Sudholt, and Carsten Witt</i>	
Simple Dynamic Particle Swarms without Velocity	144
<i>Jorge Peña</i>	
Swarming in a Virtual World: A PSO Approach to Virtual Camera Composition	155
<i>Luca Di Gaspero, Andrea Ermetici, and Roberto Ranon</i>	
The Binary Bridge Selection Problem: Stochastic Approximations and the Convergence of a Learning Algorithm	167
<i>Armand M. Makowski</i>	
Two-Level ACO for Haplotype Inference Under Pure Parsimony	179
<i>Stefano Benedettini, Andrea Roli, and Luca Di Gaspero</i>	
What Hides in Dimension X? A Quest for Visualizing Particle Swarms	191
<i>Namrata Khemka and Christian Jacob</i>	

Short Papers

A Dynamic Swarm for Visual Location Tracking	203
<i>Marcel Kronfeld, Christian Weiss, and Andreas Zell</i>	
A Simulation Study of Routing Performance in Realistic Urban Scenarios for MANETs	211
<i>Gianni A. Di Caro, Frederick Ducatelle, and Luca M. Gambardella</i>	
ACO-Based Scheduling of Parallel Batch Processing Machines with Incompatible Job Families to Minimize Total Weighted Tardiness	219
<i>Li Li, Fei Qiao, and Qidi Wu</i>	
Adaptive Particle Swarm Optimization	227
<i>Zhi-hui Zhan and Jun Zhang</i>	
Ant Based Heuristics for the Capacitated Fixed Charge Location Problem	235
<i>Harry Venable and Alfredo Moscardini</i>	
Ant Colony Optimization and the Single Round Robin Maximum Value Problem	243
<i>David C. Uthus, Patricia J. Riddle, and Hans W. Guesgen</i>	
Artificial Ants to Extract Leaf Outlines and Primary Venation Patterns	251
<i>Robert J. Mullen, Dorothy Monekosso, Sarah Barman, Paolo Remagnino, and Paul Wilkin</i>	

Autonomous Reconfiguration in a Self-assembling Multi-robot System	259
<i>Rehan O'Grady, Anders Lyhne Christensen, and Marco Dorigo</i>	
Beanbag Robotics: Robotic Swarms with 1-DoF Units	267
<i>David M.M. Kriesel, Eugene Cheung, Metin Sitti, and Hod Lipson</i>	
BlåtAnt: Bounding Networks' Diameter with a Collaborative Distributed Algorithm	275
<i>Amos Brocco, Fulvio Frapolli, and Béat Hirsbrunner</i>	
Dependency by Concentration of Pheromone Trail for Multiple Robots	283
<i>Ryusuke Fujisawa, Shigeto Dobata, Daisuke Kubota, Hikaru Imamura, and Fumitoshi Matsuno</i>	
Dissemination of Information with Fair Load Distribution in Self-organizing Grids	291
<i>Agostino Forestiero, Carlo Mastroianni, and Giandomenico Spezzano</i>	
Emergent Sorting in Networks of Router Agents	299
<i>Alexander Scheidler, Christian Blum, Daniel Merkle, and Martin Middendorf</i>	
Enhancing the Cooperative Transport of Multiple Objects	307
<i>Antoine Decugnière, Benjamin Poulain, Alexandre Campo, Carlo Pinciroli, Bruno Tartini, Michel Osée, Marco Dorigo, and Mauro Birattari</i>	
Formal Modeling of <i>BeeAdHoc</i> : A Bio-inspired Mobile Ad Hoc Network Routing Protocol	315
<i>Muhammad Saleem, Syed Ali Khayam, and Muddassar Farooq</i>	
Incorporating Heuristics in a Swarm Intelligence Framework for Inferring Gene Regulatory Networks from Gene Expression Time Series	323
<i>Kyriakos Kentzogloukakis, Matthew Poole, and Carl Adams</i>	
Incorporating Preferences to a Multi-objective Ant Colony Algorithm for Time and Space Assembly Line Balancing	331
<i>Manuel Chica, Óscar Cordón, Sergio Damas, Jordi Pereira, and Joaquín Bautista</i>	
KANTS: Artificial Ant System for Classification	339
<i>Carlos Fernandes, Antonio Miguel Mora, Juan Julián Merelo, Vitorino Ramos, Juan Luis Laredo, and Agostinho Rosa</i>	
Lattice Formation in Space for a Swarm of Pico Satellites	347
<i>Carlo Pinciroli, Mauro Birattari, Elio Tuci, Marco Dorigo, Marco del Rey Zapatero, Tamas Vinko, and Dario Izzo</i>	

Merging Groups for the Exploration of Complex State Spaces in the CPSO Approach	355
<i>Stefanie Thiem, Jörg Lässig, and Peter Köchel</i>	
Parallel Ant Colony Optimization for the Quadratic Assignment Problems with Symmetric Multi Processing	363
<i>Shigeyoshi Tsutsui</i>	
Social Odometry in Populations of Autonomous Robots	371
<i>Álvaro Gutiérrez, Alexandre Campo, Francisco C. Santos, Carlo Pinciroli, and Marco Dorigo</i>	
The Architecture of Ant-Based Clustering to Improve Topographic Mapping	379
<i>Lutz Herrmann and Alfred Ultsch</i>	
The Small World of Pheromone Trails	387
<i>Paola Pellegrini and Andrea Ellero</i>	
Extended Abstracts	
A Particle Swarm Optimization Algorithm for Multiuser Scheduling in HSDPA	395
<i>Mehmet E. Aydin, Raymon Kwan, Cyril Leung, and Jie Zhang</i>	
AntLib v1.0: A Generic C++ Framework for Ant Colony Optimization	397
<i>Francisco Javier Diego Martín, José Ángel González Manteca, Ruth Carrasco-Gallego, and Javier Carrasco Arias</i>	
Applying a Distributed Swarm-Based Algorithm to Solve Instances of the RCPSP	399
<i>Paulo R. Ferreira Jr. and Ana L.C. Bazzan</i>	
bicACO: An Ant Colony Inspired Biclustering Algorithm	401
<i>Fabrício O. de França, Guilherme P. Coelho, and Fernando J. Von Zuben</i>	
Dynamic Routing and Travel Time Prediction with Ant Based Control	403
<i>Bogdan Tatomir, Adriana-Camelia Suson, and Leon Rothkrantz</i>	
Network Formation Using Ant Colony Optimization	405
<i>Steven C. Oimoen, Gilbert L. Peterson, and Kenneth M. Hopkinson</i>	
On the Stability and the Parameters of Particle Swarm Optimization	407
<i>Keiichiro Yasuda, Nobuhiro Iwasaki, and Genki Ueno</i>	

Regional Traffic Assignment by ACO: Preliminary Results	409
<i>Vittorio Maniezzo, Matteo Roffilli, Roberto Gabrielli, Alessandra Guidazzi, Manuel Otero, and Rolando Trujillo</i>	
SwarmClass: A Novel Data Clustering Approach by a Hybridization of an Ant Colony with Flying Insects	411
<i>Amira Hamdi, Nicolas Monmarché, M. Adel Alimi, and Mohamed Slimane</i>	
The Differential Ant-Stigmergy Algorithm for Large Scale Real-Parameter Optimization	413
<i>Peter Korošec and Jurij Šilc</i>	
Author Index	415

A Combined Ant Colony and Differential Evolution Feature Selection Algorithm

Rami N. Khushaba, Ahmed Al-Ani, Akram AlSukker, and Adel Al-Jumaily

Faculty of Engineering, University of Technology, Sydney, Australia
`{rkhushab,ahmed,alsukker,adel}@eng.uts.edu.au`

Abstract. Feature selection is an important step in many pattern recognition systems that aims to overcome the so-called curse of dimensionality problem. Although Ant Colony Optimization (ACO) proved to be a powerful technique in different optimization problems, but it still needs some improvements when applied to the feature selection problem. This is due to the fact that it builds its solutions sequentially, where in feature selection this behavior will most likely not lead to the optimal solution. In this paper, a novel feature selection algorithm based on a combination of ACO and a simple, yet powerful, Differential Evolution (DE) operator is presented. The proposed combination enhances both the exploration and exploitation capabilities of the search procedure. The new algorithm is tested on two biosignal-driven applications. The performance of the proposed algorithm is compared with other dimensionality reduction techniques to prove its superiority.

1 Introduction

Pattern recognition is a multi-disciplinary field of research with the goal of classifying a set of objects into a number of categories or classes. Among the several parameters that affect the performance of pattern recognition systems, feature representation of patterns can be the most important. Feature selection (FS) aims to reduce the feature set dimensionality through selecting a subset of features that performs the best under some classification criterion [1]. This is done by eliminating irrelevant and redundant features, thus providing a better representation of the original patterns. This will significantly reduce the computational cost and will result in a better generalization for the classifier.

As a part of any feature subset selection algorithm, there are several factors that need to be considered, the two most important are the evaluation measure and the search procedure [2]. The existing evaluation measures utilized in feature selection techniques are divided into two categories according to their dependency on the classification algorithms namely: filters and wrappers. Filter based feature selection methods are in general faster than wrapper based methods. This is due to the fact that the filter based methods depend on some type of estimation of the importance of individual features or subset of features. Comparing to filter methods, wrapper based methods are more accurate as the importance of feature subsets is measured using a classification algorithm. On

the other hand, a search strategy is needed to explore the feature space. Various search algorithms that differ in their optimality and computational cost have been developed to search the solution space. These methods include: Tabu Search (TS), Simulated Annealing (SA), and Genetic Algorithm (GA) [3]. Another trend of search procedures is based on swarm intelligence, which adopts the social insect metaphor that emphasizes distributedness and direct or indirect interactions among relatively simple agents. Swarm intelligence methods, particularly the Ant Colony Optimization (ACO) [4] and Particle Swarm Optimization (PSO) [5] were also utilized as search procedures in feature selection problems [2,6].

Ant colony optimization is a promising approach to solve discrete optimization problems. It was initially used to solve the well known travelling salesman problem. There were few attempts in the literature that utilized ACO in feature selection, where it was used to reduce the dimensionality in face, medical diagnostic, speech, and texture classification problems [2]. However, the main limitation of those methods is the sequential selection of features, which in most cases will not lead to an optimal solution.

This paper presents a novel feature selection algorithm based on a combination of ACO and a Differential Evolution (DE) [7] operator. Although DE optimization technique was originally developed to optimize problems with real valued variables, an extension of the original DE algorithm to discrete problems is presented. The new algorithm, termed ANTDE, will be tested on the Brain Computer Interface (BCI) and the multifunction myoelectric control (MEC) problems and the performance will be compared with other state of the art feature selection and projection techniques.

2 Ant Colony Optimization and Feature Selection

In real ant colonies, a pheromone, which is an odour substance, is used as an indirect communication medium. When a source of food is found, the ants lay some pheromone to mark the path. The quantity of the laid pheromone depends upon the distance, quantity and quality of the food source. While an isolated ant that moves at random detects a laid pheromone, it is very likely that it will decide to follow its path. This ant will itself lay a certain amount of pheromone, and hence enforce the pheromone trail of that specific path. Accordingly, the path that has been used by more ants will be more attractive to follow. Dorigo et. al. [4] adopted this concept and proposed an artificial algorithm based on real ant colonies behavior, to solve hard combinatorial optimization problems. The ACO metaheuristic was originally applied to solve the classical Travelling Salesman Problem (TSP), where it was shown to be an effective tool in finding good solutions.

2.1 Application of ACO in Feature Selection

The feature selection problem differs from TSP as the distance between cities are fixed in TSP. When adding one more city, the change in the objective function is

affected only by the distance between last two cities. In contrast to TSP, adding a feature to an existing subset of features can have an impact on the overall performance. A relevant feature will produce a better subset, and hence improve the performance, while an irrelevant feature may degrade the performance of the original subset. When adding a feature to the current feature subset the local performance measure should take into account the relationship with all previously selected features and not only the last one. This makes the problem of feature selection more complicated.

Various ACO based feature selection algorithms were presented in the literature. Some of them employed a hybrid filter and wrapper techniques to estimate the heuristic information and overall performance. As an example, Al-Ani [2] proposed an ACO based feature selection algorithm that estimates the pheromone values by means of mutual information measure, and the overall performance using a neural network classifier. The method was tested on two different classification problems achieving higher results than a GA based feature selection approach. Zhang et al [8] has also used the hybrid of ACO and mutual information for selection of features in a forecasting problem. As a different approach, Gao et al [9] utilized the Fisher Discrimination Rate (FDR) as heuristic information in an ACO-based feature selection method used for selection of features in a network intrusion problem. Jensen et al [10] on the other hand, employed ACO for finding rough set reducts. On the other hand, the classifier performance was adopted as heuristic information for ACO in both Kanan et al [11] and Yan et al [12] experiments.

This paper presents a variation of the approach proposed by Al-Ani [2]. Due to the fact that the original algorithm searches for the global optimal by forming the solutions in a semi-sequential way, there is a chance for the ants to be trapped in a local minima. To overcome this limitation, a parallel search mechanism will be required. The most well known parallel search algorithms are GA, PSO, and DE algorithms. In feature selection problems with both GA and PSO, binary strings are employed usually in which every bit represents an attribute. The value of '1' means that the attribute is selected while '0' means not selected. This increases the computational cost for large problems. As an example consider a problem with 256 features. If a subset of 20 feature is required, then for a population of 50 elements, the total size of the population matrix will be 50×256 . On the other hand, DE was introduced to solve problems with real values. The DE optimization technique can be viewed as an enhanced version of the real valued GA that employs a differential mutation operator with faster convergence properties.

We modified the original DE algorithm to make it suitable for the problem of feature selection without converting into binary strings. Thus, for the example mentioned earlier, the size of the population matrix will be 50×20 , hence, a lower memory requirement than both GA and PSO. Since DE proved to present good performance in different problems [7], it was adopted here to form with ACO a novel feature selection algorithm.

2.2 Differential Evolution

Differential Evolution (DE) is an optimization method, capable of handling non-differentiable, nonlinear and multimodal objective functions. It is a simple, parallel, direct search, easy to use, having good convergence, and fast implementation properties [7]. The crucial idea behind DE is a new scheme for generating trial parameter vectors by adding the weighted difference vector between two population members (r_1 and r_2) to a third member (r_3). The following equation shows how to combine three different, randomly chosen vectors to create a mutant vector, $V_{i,g}$ from the current generation g :

$$V_{i,g} = X_{r0,g} + F \times (X_{r1,g} - X_{r2,g}) \quad (1)$$

where $F \in (0, 1)$ is a scale factor that controls the rate at which the population evolves.

Extracting both distance and direction information from the population to generate random deviations results in an adaptive scheme that has excellent convergence properties. In addition, DE also employs uniform crossover, also known as discrete recombination, in order to build trial vectors out of parameter values that have been copied from two different vectors. In particular, DE crosses each vector with a mutant vector, as given in Eq. (2) below:

$$U_{j,i,g} = \begin{cases} V_{j,i,g} & \text{if } \text{rand}(0,1) \leq C_r \text{ or} \\ X_{j,i,g} & \text{Otherwise} \end{cases} \quad (2)$$

where $U_{j,i,g}$ is the j^{th} trial vector along i^{th} dimension from the current population g . The crossover probability $C_r \in [0, 1]$ is a user defined value that controls the fraction of parameter values that are copied from the mutant. If the newly generated vector results in a lower objective function value (better fitness) than the predetermined population member, then the resulting vector replaces the vector with which it was compared.

As a novel contribution of this paper:

1. The population upon which the DE operators are performed are actually drawn from the solutions that the ACO finds. Hence, DE is utilized to further explore and exploit around the solutions that each of the ants found. This is controlled by the values of the scale factor F .
2. Initially the value of F is made to linearly increase from 0.4 to 0.9, thus first exploiting around the solutions provided by ACO and gradually increasing to 0.9 thus further exploring around the solutions. If during any iteration a new global minimum (higher fitness) is found then the value of F is reset to 0.4 and made to increase again. Also for the mutant vector generation, either Eq. (1) or the one presented below can be used:

$$V_{ig} = X_{best,g} + F \times (X_{r1,g} - X_{r2,g}) \quad (3)$$

where $X_{best,g}$ is the best solution found in the current generation g .

3. Since DE is a numerical optimizer, it will need certain amendments before being suitable for combinatorial optimization problems. This is best understood with the following example. Consider the same problem mentioned earlier with 256 features from which we seek a subset of 5 features. When using DE directly the solutions produced will be float numbers, while in FS problems we need positive decimal numbers. Rounding the solution of DE is the first operation applied. Secondly, when optimizing a problem with a numerical optimizer, nothing can prevent two or more dimensions from settling at the same number. As an example if S (the subset of selected features by a specific ant) is [1.11 202.56 35.98 36.32 90.07] then the rounded value of S would be [1 203 36 36 90]. This is completely unacceptable as feature (36) is repeated. In order to overcome such a problem, the redundancies in the solutions produced by DE are removed by utilizing the pheromone intensities from the ants. In other words, the feature indices are sorted in a descending manner according to their pheromone values. The repeated features only will be replaced by the first few features with high pheromone intensities. Thus S could be for example [1 203 36 150 90] if 150 has the highest pheromone value (i.e., it is used by most of the ants).

3 The Proposed Feature Selection Algorithm

A hybrid evaluation measure that is able to estimate the overall importance of subsets as well as the local importance of features is proposed. A Linear Discriminant Analysis (LDA classifier) is used to estimate the performance of subsets (i.e., a wrapper evaluation function). On the other hand, the local importance of a given feature is measured using the mutual information. For this purpose we adopted the approach proposed in [13] known as the mutual information evaluation function (MIEF).

The following parameters are used in the algorithm:

- n : number of features that constitute the original set, $F = \{f_1, \dots, f_n\}$
- na : number of artificial ants to search through the feature space
- τ_i : intensity of pheromone trail associated with feature f_i
- PL :list of the previously tested subsets.
- BL :list of the best k subsets.
- k :where the best k subsets ($k < na$) will be used to influence the feature subsets of the next iteration.
- $S_j = \{s_1, \dots, s_m\}$:a list that contains the selected feature subset for ant j

In the first iteration, each ant will randomly choose a subset of m features, where m is the desired number of features. In the second and following iterations, each ant will start with $m - p$ features that are randomly chosen from the previously selected k -best subsets, where p is an integer that ranges between 1 and $m - 1$. In this way, the features that constitute the best k subsets will have more chance to be present in the subsets of the next iteration. Nevertheless, it will still be possible for each ant to consider other features as well. For

instance, ant j will consider those features that achieve the best compromise between previous knowledge, i.e., pheromone trails, and local importance. The local importance of feature f_i is measured with respect to the features of S_j (features that have already been selected by ant j). The Selection Measure (SM) used for this purpose is defined as:

$$SM_j^{S_j} = \begin{cases} \frac{(\tau_i)^\eta (LI_i^{S_j})^\psi}{\sum_{g \notin S_j} (\tau_g)^\eta (LI_g^{S_j})^\psi} & \text{if } i \in S_j \\ 0 & \text{Otherwise} \end{cases} \quad (4)$$

where $LI_i^{S_j}$ is the local importance of feature f_i given subset S_j . The parameters η and ψ control the effect of trial intensity and local feature importance respectively.

$$LI_i^{S_j} = I(C; f_i) \times \left[\frac{2}{1 + \exp(-\alpha \times D_i^{S_j})} - 1 \right] \quad (5)$$

where

$$D_i^{S_j} = \min_{f_s \in S_j} \left[\frac{H(f_i) - I(f_i, f_s)}{H(f_i)} \right] \times \frac{1}{|S_j|} \sum_{f_i \in S_j} \left[\beta \left(\frac{I(C; f_i, f_s)}{I(C; f_i) + I(C; f_s)} \right)^\gamma \right] \quad (6)$$

The parameters α , β , and γ are constants, $H(f_i)$ is the entropy of f_i , $I(f_i; f_s)$ is the mutual information between f_i and f_s , $I(C; f_i)$ is the mutual information between the class labels and f_i , and $|S_j|$ is the cardinal of S_j .

Below are the steps of the algorithm:

1. Initialization:
 - Start with a fixed small amount of pheromone for all ants, $\tau_i = cc$, where cc is a constant.
 - Define the maximum number of iterations.
 - Define k , where the k -best subsets will influence the subsets of next iteration.
 - Define p , where $m - p$ is the number of features each ant will start with in the second and following iterations.
2. If in the first iteration,
 - For $j = 1$ to na ,
 - Randomly assign a subset of m features to S_j .
 - Goto step 4.
3. Select the remaining p features for each ant:
 - For $mm = m-p+1$ to m ,
 - For $j = 1$ to na ,
 - * Given subset S_j , Choose feature f_i that maximizes $SM_i^{S_j}$
 - * $S_j = S_j \cup f_i$
4. Evaluate the selected subset of each ant using the chosen classification algorithm:
 - For $j = 1$ to na ,
 - Compute the Mean Square Error (MSE_j) of the classification results obtained by classifying the features of S_j using an LDA classifier.

- Sort the subsets according to their MSE . Update the minimum MSE (if achieved by any ant), and store the corresponding subset of features.
 - Update the list of the previously tested subsets. $PL = [PL; S_j]$, where $(j = 1 : na)$.
5. Apply the DE operator represented by Eqs. (2) and (3) once in each iteration.
- If redundancies exist in the feature subsets.
 - Sort the features according to the pheromone intensities τ_i associated with each feature.
 - Replace redundant features by the first few features with highest pheromone intensities
 - Evaluate the new solutions and decide whether to keep the original solution found by ACO or the new ones resulting from the DE operator.
6. Update BL (the list of the k best subsets).
7. For each feature f_i , update the pheromone trail according to the following formula:

$$\tau_i = a_1 R_{1i} + a_2 R_{2i} + a_3(1 - R_{3i}) + a_4 \quad (7)$$

where

- a_1, a_2, a_3 , and a_4 are constants.
- R_{1i} : ratio indicating the occurrence of f_i in BL .
- R_{2i} : ratio between the occurrence of f_i in the best half subsets and the overall occurrence of f_i .
- R_{3i} : ratio indicating the overall occurrence of f_i .

8. Using the feature subsets of the best k ant:
- For $j = 1$ to na ,
 - Randomly produce $m - p$ feature subset for ant j , to be used in the next iteration, and store it in S_j .
 - 9. If the stopping criterion is not met, goto 3.

The rationale behind Eq. (7) is to estimate the pheromone intensity of f_i . R_{1i} shows the contribution of f_i toward the best k subsets. R_{2i} indicates the degree that f_i contributes in forming good subsets. Hence, a new subset formed by combining f_i with other features might become the best subset. The term $(1 - R_{3i})$ aims at favoring exploration, where this term will be close to 1 if the overall usage of f_i is very low. The reader can refer to [2,13] for the selection of all the parameters mentioned above.

4 Experiments and Practical Results

Two biosignal-driven applications were used to prove the effectiveness of the ANTDE algorithm. The first application involves the utilization of the Electroencephalogram (EEG) signal from human brain in a brain-computer-interface problem (BCI). The second application chosen is the myoelectric control (MEC) problem, in which the human muscles activity, known as the Myoelectric Signal (MES), is utilized in a noninvasive manner as a control signal for external

devices. Both of these applications witnessed a great focus of research in the last few years. Due to the fact that such biosignals driven applications usually utilize a multichannel approach, the feature vector size can become very large. This in turn will increase the computational cost for such systems, while at the same time affecting the generalization capability of the classifier. The large size of the extracted feature sets would also introduce a time delay which hinders the development of continuous real time control systems.

4.1 A Comparison with Other Feature Selection Techniques in BCI Problem

The data used here was obtained from the University of Technology, Graz, Austria.¹ EEG signals were recorded for three right-handed females with 56 Ag/AgCl Electrodes using monopolar montage, with reference electrode on the right ear. The subjects were placed in an armchair and asked to imagine right or left finger movements according to stimuli on screen. A total of 406 trials were used, 208 for left movement and 198 for right. The wavelet packet transform was used in this paper to extract features from this dataset. The total number of features extracted were 168 features (56 channels \times 3 features/channel). For more information about the feature extraction process the reader can refer to [14]. The first 300 patterns were used for training and the rest of the data, 106 patterns, were used for testing.

The proposed ANTDE was tested against all of the following methods: the original ant colony feature subset selection by Al-Ani [2] (referred to as ANT), Genetic Algorithm (referred to as GA), and the binary particle swarm optimization (referred to as BPSO). The results of the comparison are given in Fig. 1. The desired number of selected features was varied between 3 and 99 features. Each of the mentioned algorithms was executed for ten times when selecting each feature subset. For example when selecting 9 features, each method was used ten times and the average result is reported here. It is also worth to mention that the same initial population was used for all the methods.

In order to analyze the results, one can start by first looking at the performance of the methods when selecting a small feature subset. The figure shows that both ANT and ANTDE achieved higher classification accuracies than GA and BPSO despite the fact that all methods started from the same initial population. This is expected as both ANT and ANTDE employ mutual information (MI) based heuristic measure. The ants are guided using MIEF into the vicinity with features that best interact together. Since the MIEF measure uses a sequential procedure to evaluate the importance of features, this is expected to give good results when selecting small number of features. However, one of the reasons why the performance of the ANT algorithm becomes very near to that of GA when selecting large number of features is due to the fact that the MIEF becomes less accurate in estimating the true MI when the number of features

¹ The authors would like to thank the Department of Medical Informatics, University of Technology, Graz, Austria for providing the EEG data.

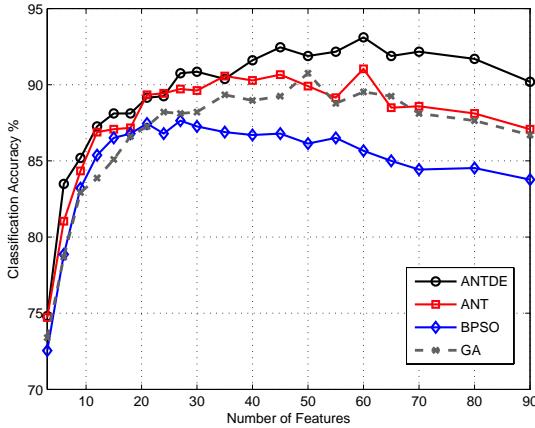


Fig. 1. A comparison of ANTDE with other feature selection techniques like ANT, GA, and BPSO on the BCI EEG dataset

increase. The other reason is the fact that ACO actually builds its solutions using a sequential approach, which can be good when dealing with small number of features, but with large dimensionalities this may not lead to the optimal solution. The ANTDE on the other hand can further exploit and explore around the initial solution provided by the ants, as it also employs a parallel DE based search procedure, thus providing a powerful mixture of both. In general, within all of the selected feature subsets the performance of BPSO and GA was almost always worse than ANT, while ANTDE performance was almost always the best. The ANTDE was the only algorithm that achieved a maximum accuracy of 93.11% while the maximum accuracies achieved by the other methods were 91.03%, 90.7%, and 87.64% for ANT, GA, and BPSO respectively.

4.2 A Comparison with Feature Projection Techniques in MEC Problem

The reason behind selecting the MEC problem for evaluating the successfulness of ANTDE is that within the MEC problem the focus had always been on utilizing feature projection techniques as a dimensionality reduction stage. Some of the proposed dimensionality reduction techniques in myoelectric control include principal components analysis (PCA) [15], the combination of PCA and a self organizing feature map (SOFM) [16], linear discriminant analysis (LDA) based feature projection [17], and the uncorrelated linear discriminant analysis (ULDA) [18]. This is due to the fact that these methods are able to compress the whole variance in a subset of few features.

In this paper, we apply the ANTDE algorithm in MEC to re-evaluate the significance of feature selection in MES classification problems. This is based on the fact that the proposed method can select subsets of features that best interact together, and thus produce high classification accuracies. Testing is performed

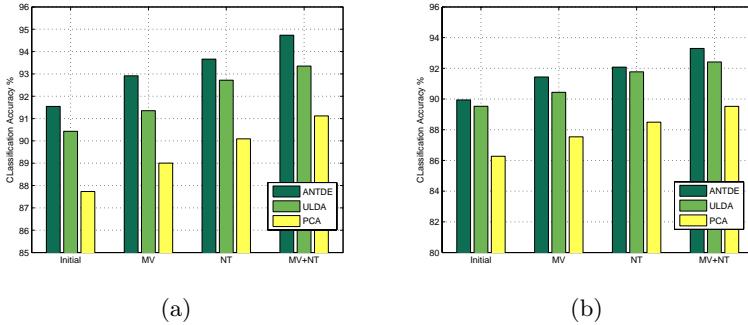


Fig. 2. Classification accuracies averaged across 30 subjects with different dimensionality reduction techniques (a) Using the validation set and (b) Using the testing set

by adopting a three way data split scheme in which the datasets were divided into a training set, validation set, and testing set. The features that minimize both of the training error and validation error are chosen as the members of the best solution. Then generalization capability of the classifier is tested based upon the completely unseen testing data (unseen during training and validation).

The MES datasets utilized in this experiment was originally collected by Goge et al [19].² Eight channels of surface MES were collected from the right arm of thirty normally limbed subjects (twelve males and eighteen females). Each subject underwent four sessions, with one to two days separation between sessions. Each session consisted of six trials. Seven distinct limb motions were used, hand open (HO), hand close (HC), supination (S), pronation (P), wrist flexion (WF), wrist extension (WE), and rest state (R). Similar to Goge's original research, we only used session four here. Data from the first two trials were used as training set and data from the remaining four trials were divided equally into two trials for validation (trails 1 and 2) and two trials for testing (trails 3 and 4).

The extracted feature set included the mean of the square values of the wavelet coefficients using a Symmlet wavelet (WT) family with five levels of decomposition (total of 48 features = 8 channels × 6 features/channel). The desired number of features was set to be equal to only 10 features. Classification is performed using a linear discriminant classifier (LDA). The advantage of this classifier is that it does not require iterative training, avoiding the potential for under- or over-training. The classification results averaged across thirty subjects are shown in Fig. 2. It should be mentioned here that the output of the MES pattern recognition system is usually smoothed using a majority vote post processing technique [15]. It has been found that applying majority vote in MES classification problems represents a necessary step as it can achieve an enhancement in the MES classification accuracy of about 2%. Another step that is usually utilized in MES recognition problems is to remove the transitional data between classes. This is due to the fact that the system is in an undetermined state between

² The authors would like to thank Dr. Adrian D. C. Chan from Carleton University for providing the MES datasets.

contractions [14]. The results shown for both the validation and the testing sets were given first without a majority vote (referred to as Initial), then with a majority vote (MV), followed by excluding the transitional data between classes (NT), and finally with both majority voting and the excluding of transitional data (MV+NT).

When analyzing the results, it was obvious that the hit rates obtained by both the proposed ANTDE and ULDA algorithms highly outperform PCA. This is expected as the latter does not take into account the relation between features and class labels. On the other hand, ULDA projects the data into the direct that maximizes the ratio of the between class scatter matrix to the within class scatter matrix. One issue to be mentioned regarding ULDA is that the resulting dimensionality is limited to $C-1$, where C represents the number of classes. Although this might be an advantage since it highly reduces the number of projected features; but it could also serve as a limitation to this technique, as this small number of features may not give an optimal solution. In comparison, the ANTDE has the flexibility of selecting feature subsets of different sizes by means of a hybrid technique that maximizes the discrimination capability of the classifier. The figure shows that for both validation and testing data ANTDE achieved the highest classification accuracies across all subjects. The accuracy achieved by the proposed ANTDE was 94.73% comparing to 93.35% and 91.11% for ULDA and PCA respectively for the validation set and 93.39% for ANTDE and 92.41% and 89.52% for ULDA and PCA respectively for the testing set.

5 Conclusion

This paper presented a novel feature selection algorithm based on a combination of ant colony and differential evolution optimization techniques. This was inspired from the fact that the ACO algorithm builds the solutions in a sequential manner, which may not lead to the optimal solution. The new mixture with DE provides further exploitation and exploration around the solutions found by the ants. The proposed ANTDE was compared with other well-known feature selection and projection techniques using two different biosignal-driven applications. The results indicated the significance of the proposed method in achieving higher classification accuracies than the other methods.

References

1. Liu, H., Motoda, H.: Computational Methods of Feature Selection. Taylor & Francis Group, Abington (2008)
2. Al-Ani, A.: Feature subset selection using ant colony optimization. Int. Journal of Computational Intelligence 2, 53–58 (2005)
3. Frohlich, H., Chapelle, O., Scholkopf, B.: Feature selection for support vector machines by means of genetic algorithms. In: 15th IEEE International Conference on Tools with Artificial Intelligence (ICTAI 2003), pp. 142–148 (2003)
4. Dorigo, M., Stützle, T.: Ant Colony Optimization. MIT Press, London (2004)

5. Kennedy, J., Eberhart, R.C., Shi, Y.: *Swarm Intelligence*. Morgan Kaufmann, London (2001)
6. Firpi, H., Goodman, E.: Swarm feature selection. In: Proceedings of the 33rd Applied Imagery Pattern Recognition Workshop (AIPR 2004), pp. 112–118 (2004)
7. Price, K., Storn, R., Lampinen, J.: *Differential Evolution: A Practical Approach to Global Optimization*. Springer, Heidelberg (2005)
8. Zhang, C., Hu, H.: Feature selection using the hybrid of ant colony optimization and mutual information for the forecaster. In: Proceedings of International Conference on Machine Learning and Cybernetics, pp. 1728–1732 (2005)
9. Gao, H., Yang, H., Wang, X.: Ant colony optimization based network intrusion feature selection and detection. In: Proceedings of 2005 International Conference on Machine Learning and Cybernetics, pp. 3871–3875 (2005)
10. Jensen, R.: Combining Rough and Fuzzy Sets for Feature Selection. PhD thesis, University of Edinburgh (2005)
11. Kanan, H., Faez, K., Taheri, S.: Feature selection using ant colony optimization (aco): A new method and comparative study in the application of face recognition system. In: Perner, P. (ed.) ICDM 2007. LNCS (LNAI), vol. 4597, pp. 63–76. Springer, Heidelberg (2007)
12. Yan, Z., Yuan, C.: Ant colony optimization for feature selection in face recognition. In: Zhang, D., Jain, A.K. (eds.) ICBA 2004. LNCS, vol. 3072, pp. 221–226. Springer, Heidelberg (2004)
13. Al-Ani, A., Deriche, M., Chebil, J.: A new mutual information based measure for feature selection. *Intelligent Data Analysis* 7, 43–47 (2003)
14. Al-Ani, A., Al-Sukker, A.: Effect of feature and channel selection on eeg classification. In: Proceedings of The 28th IEEE EMBS Annual International Conference, New York City, USA, pp. 2171–2174 (2006)
15. Englehart, K.: Signal Representation for Classification of The Transient Myoelectric Signal. PhD thesis, University of New Brunswick (1998)
16. Chu, J., Moon, I., Mun, M.: A real-time emg pattern recognition system based on linear-nonlinear feature projection for a multifunction myoelectric hand. *IEEE Trans. on Biomedical Engineering* 53(11), 2232–2239 (2006)
17. Chu, J., Moon, I., Mun, M.: A supervised feature projection for real-time multifunction myoelectric hand control. In: Proceedings of The 28th IEEE EMBS Annual International Conference, New York City, USA, pp. 2417–2420 (2006)
18. Chan, A., Green, G.: Myoelectric control development toolbox. In: Proceedings of The 30'th Conference of the Canadian Medical & Biological Engineering Society, Toronto, ON (2007)
19. Goge, A., Chan, A.: Investigating classification parameters for continuous myoelectrically controlled prostheses. In: Proceedings of The 28th Conference of the Canadian Medical & Biological Engineering Society, Quebec City, Canada, pp. 141–144 (2004)

An Improved ACO Based Plug-in to Enhance the Interpretability of Fuzzy Rule Bases with Exceptions

Pablo Carmona¹ and Juan Luis Castro²

¹ Department of Computer and Telematics Systems Engineering
Industrial Engineering School, University of Extremadura, Spain
pablo@unex.es

² Department of Computer Science and Artificial Intelligence
Computer and Telecommunication Engineering School, University of Granada, Spain
castro@decsai.ugr.es

Abstract. In a previous work, the authors proposed, on one hand, an extension on the syntax of fuzzy rules by including new predicates and exceptional rules and, on the other hand, the use of an ant colony optimization algorithm to obtain an optimal set of such rules that describes an initial fuzzy model. The present work proposes several extensions on that algorithm in order to improve the interpretability of the obtained fuzzy model, as well as the computational cost of the algorithm. Experimental results on several initial fuzzy models reveal the gain obtained with each extension and when applied altogether.

1 Introduction

Despite the interpretability is one of the distinctive features of fuzzy models, it is often underestimated in the search for an accuracy improvement. Nevertheless, in the last years, research efforts have been redirected to preserve and enhance the interpretability power of this type of models in order to obtain a good interpretability-accuracy trade-off [1].

One way to improve the interpretability of a fuzzy model consists of trying to identify rules as general as possible, so that each rule positively covers the highest number of examples [2], and, ultimately, the number and complexity of the rule diminish. In addition, by extending the syntax of the rules with new predicates different from the usual *equal-to* predicate more compact rules can be provided. Moreover, recently the authors proposed the use of exceptions to further increase the interpretability of models described with fuzzy rules [3,4].

However, finding the optimal set of such general rules with exceptions is not an easy task. In a previous work [5], the authors propose a method to search for the best combination of general rules with exceptions that describes an initial fuzzy model. In order to solve this combinatorial problem, an ant colony optimization (ACO) algorithm [6] is proposed. This technique is a global optimization method which lies on the emergent behavior that rises from the cooperative search of a

set of agents called artificial ants. These ants communicate among them in an indirect way —*stigmergy*— by means of a shared memory which emulates the pheromones that real ants deposit along the paths they trace between the nest and the food. For further readings, see [7].

In this work, several extensions to the algorithm described in [5] are proposed in order to improve the interpretability of the obtained fuzzy model, as well as the computational cost of the algorithm. The first extension consists of replacing the AS model used in the original method with the more advanced ACS model. The second extension adds a local search to refine each solution, a common practice in this and other metaheuristics. Finally, a third extension proposes to use a candidate list in order to restrict the number of steps considered to select the next one, trying that way to diminish the computational cost of the algorithm.

Next section introduces the syntax of the general rules with exceptions which allows to enhance the interpretability of fuzzy models. Section 3 outlines the ACO algorithm proposed in [5]. Section 4 constitutes the core of the paper, where different extensions to the algorithm are detailed. In Section 5, some experimental results are provided to show the gains obtained separately for each extension and when applied altogether. Section 6 summarizes the conclusions.

2 Single, Compound and Exceptional Rules

We consider Multiple-Input–Single-Output (MISO) systems with n input variables $\mathbf{X} = \{X_1, \dots, X_n\}$ defined over the universe of discourse $\mathcal{X} = \mathcal{X}_1 \times \dots \times \mathcal{X}_n$, and one output variable Y defined over the universe of discourse \mathcal{Y} . The fuzzy domain of X_i is denoted as $\widetilde{\mathcal{X}}_i = \{LX_{i,1}, \dots, LX_{i,p_i}\}$, where $LX_{i,j}$ represents the linguistic label of the j th value, and p_i is the granularity of the fuzzy domain. Analogously, $\widetilde{\mathcal{Y}} = \{LY_1, \dots, LY_q\}$ is the output fuzzy domain.

Usually, the fuzzy rules for MISO systems contain in their antecedent a premise for each input variable which associates it with a label from its fuzzy domain. We refer to this type of rules as *single rules* and they are denoted as:

$$R_{LY^i}^{i_1 \dots i_n} : LX_{1,i_1}, \dots, LX_{n,i_n} \rightarrow LY^i.$$

In order to improve the interpretability of the fuzzy rules, it is possible to extend their syntax both by associating more than one label to an input variable and by using the negation and other predicates different from the usual *equal-to* predicate. We call them *compound rules* and they are denoted as:

$$R^i : [\neg]X_1 op_i SX_1^i, \dots, [\neg]X_n op_n SX_n^i \rightarrow LY^i$$

where $op_i \in \{=, \leq, \geq, \div, \in\}$ (\div means *between*), and where each SX_j^i is one label if $op_i \in \{=, \leq, \geq\}$, two labels if op_i is \div , and a set of labels associated disjunctively if op_i is \in .

Moreover, the compactness of the fuzzy model can be further improved by introducing the concept of *exceptional rule*. An exceptional rule is associated

with another one —the *excepted rule*— and excludes it from being fired in a region of its coverage, being the exceptional rule fired instead. This idea was firstly introduced by the authors in [3,4]. Here, we also propose to take advantage of the context introduced by the antecedent of the excepted rule to express the antecedent of the exceptional rule in a further compact way.

For example, suppose a 3-input/1-output system with the same fuzzy domain $\{NL, NS, Z, PS, PL\}$ for all the variables. The excepted and exceptional rules

$$R^1 : X_1 \in \{NL, Z, PS\}, X_2 \div [NS, PS] \rightarrow NL \\ \text{exc: } X_1 \geq Z, X_3 = Z \rightarrow NS$$

comprise $3 \times 3 \times 5 = 45$ single rules and equal to the set of non-excepted compound rules

$$R^1 : X_1 = NL, X_2 \div [NS, PS] \rightarrow NL, \\ R^2 : X_1 \in \{Z, PS\}, X_2 \div [NS, PS], \neg X_3 = Z \rightarrow NL, \\ R^3 : X_1 \in \{Z, PS\}, X_2 \div [NS, PS], X_3 = Z \rightarrow NS.$$

3 The ACO Algorithm

In this section, the original algorithm proposed in [5] is presented. However, in order to make comparable the experimental results of the proposed extensions with this original algorithm, some changes have been included here as part of it. Specifically, the use of negation (\neg), the expression of the exceptional rules taken advantage of the context introduced by the antecedent of their excepted rules—as described in Section 2—the change explained below in the footnote of Section 3.1, and the initial amount of pheromone defined below in (7) are not actually included in the original algorithm described in [5].

The algorithm searches for the best transformation of a set of initial rules (SIR) into a set of compound rules (SCR) possibly with exceptional rules. In order to do that, ants can take *inclusion* steps, that add rules from the SIR to the SCR, or *amplification* steps, that extend the coverage of the rules in the SCR by adding a label to one of the premises in the antecedent. Amplifications are denoted by $< i, j, k >$, which represents the addition of the k th label, $LX_{j,k}$, to the j th premise in the antecedent of R^i . Inclusions are denoted by $< i, 0, 0 >$, which represents the inclusion of the i th initial rule (IR) into the SCR, giving the compound rule R^i .

3.1 Solution Construction Process

In order to build a solution, firstly the ant is randomly located in an IR, so that this IR is included in the SCR of the ant. Subsequently, the ant will select one step among the *feasible* transitions for its state. A step is feasible:¹

¹ Some modifications to the definition of feasible step presented in [5] are included here in order to allow that consistent rules (i.e., rules with the same consequent) overlap.

1. If it is an inclusion, when the IR is not yet positively covered by the SCR.
2. If it is an amplification of a compound rule, R^i :
 - (a) if R^i is not exceptional, when:
 - i. the amplification zone (AZ) —i.e., the new zone covered by $R^{i'}$ (the amplified R^i)— does not only negatively cover IRs, and
 - ii. each R^j that overlaps with $R^{i'}$ either:
 - is consistent with $R^{i'}$ and, either subsumes it or the overlapping zone does not negatively cover IRs, or
 - is in conflict with $R^{i'}$, subsumes it, is not exceptional of other rule and does not cover negatively any IR.
 - (b) If R^i is exceptional of R^j , when the AZ is included in the coverage of R^j , does not cover negatively any IR nor overlaps with any rule in conflict with R^i but R^j .

After each amplification, each consistent rule subsuming with R^i is deleted and each conflicting rule overlapping with R^i becomes exceptional of it.

3.2 Heuristic Information

It guides the search toward paths containing promising steps and is defined as

$$\eta = \gamma \frac{Reg_{cov}}{Reg_{max}} + (1 - \gamma) \frac{Reg_+ / (Reg_- + 1)}{Reg_{max}}, \quad (1)$$

where Reg_{cov} is the number of input regions covered by the AZ, Reg_+ (Reg_-) is the number of IR positively (negatively) covered by the AZ and Reg_{max} is the maximum number of input regions covered by the AZ when amplifying the variable considered in the step (i.e., if $s = < i, j, k >$, $Reg_{max} = \prod_{l \neq j} p_l$)².

Eq. (1) is evaluated just over the state of the ant preceding the step, since the AZ of an amplification depends on the previous steps.

3.3 Pheromone Update

It refers to the pheromones deposited by an ant once it has completed a solution (i.e., a final SCR). However, a process is launched before this pheromone update in order to delete those rules that subsume not in an unique compound rule, but in the whole rule base. During this process, the excepted rules are analyzed before its exceptional rules, so that if an excepted rule is deleted, its exceptional rules become non exceptional rules and then their subsumption in the rule base are analyzed. In this respect, an excepted rule subsumes in the rule base if all its coverage but the ones of its exceptional rules is covered by other rules in the CRC.

Regarding the pheromone update, given a solution, the interpretability of each final compound rule R^i is evaluated and an amount of pheromone proportional

² If $s = < i, 0, 0 >$, $Reg_{max} = \prod_l p_l / \min_l p_l$ and (1) is divided by 2 in order to favor amplifications.

to this value is deposited in every step $< i, \cdot, \cdot >$ taken by the ant. Both excepted and exceptional rules are equally dealt.

The evaluation of a compound rule is regarded as a function of its syntactic simplicity. Due to this, the complexity of each premise P_j is defined as³

$$c(P_j) = \begin{cases} 0 & \text{if } \text{void premise} \\ 3 & \text{if } X_j(=, \leq, \geq) LX_{j,k} \\ 4 & \text{if } X_j \div [LX_{j,k}, LX_{j,k+l}] \\ l + 2 & \text{if } X_j \in \{LX_{j,k_1}, \dots, LX_{j,k_l}\} \end{cases} \quad (2)$$

the complexity of a rule is defined as

$$C(R^i) = \min \left(1, \frac{E + \sum_j c(\neg P_j)}{\sum_j p_j + 1} \right), \quad (3)$$

where E is the number of exceptional rules of R^i , and its interpretability is defined as

$$I(R^i) = \delta(1 - C(R^i)) + (1 - \delta) \frac{Reg_{cov}}{Reg_{max}}, \quad (4)$$

where Reg_{cov} is the number of input regions covered by R^i but those covered by its exceptional rules, $Reg_{max} = \prod_l p_l$, and $\delta \in [0, 1]$.

Besides, since the suitability of a step also depends on the previous steps, the amount of pheromone deposited in a step $s : < i, \cdot, \cdot >$ is calculated as

$$\Delta\tau_s = I(R^i) \times \frac{L - l + 1}{L}, \quad (5)$$

where L is the length of the path and l is the locus in the path of $< i, 0, 0 >$.

Finally, the measure used to select the best solution is defined as

$$I(\mathcal{RB}) = \frac{NIR - \sum_i C(R^i)}{NIR}, \quad (6)$$

where NIR is the number of IRs and equals to the highest complexity of the model (the highest number of rules with the highest interpretability).

The construction graph is initialized with an amount of pheromone equal to

$$\tau_0 = \overline{IR}_{greedy}/NR_{greedy} \quad (7)$$

where \overline{IR}_{greedy} is the averaged interpretability of the fuzzy rules obtained with a greedy algorithm equivalent to the proposed ant colony algorithm (without the extensions in 4.2 and 4.3) but always selecting the step with the highest heuristic value, and NR_{greedy} is the number of rules in such fuzzy model.

³ In order to consider negation, both P_j and $\neg P_j$ are evaluated (where $c(\neg P_j) = c(P_j) + 1$), being the simplest considered in the rest of equations.

4 Extensions to the Algorithm

Several extensions have been developed in order to improve the interpretability of the results, as well as the computational cost of the algorithm. Those mechanisms are described in this section.

4.1 ACS Model

The first extension consists of replacing the AS model [6] used in the original algorithm with the more advanced ACS model proposed by Dorigo and Gambardella in [8]. The novelties of this model are:

- The random proportional rule used in the AS model for selecting a step is replaced in the ACS model with a pseudorandom proportional rule, which uses a new parameter q_0 that allows to establish the probability for selecting the best step (from the heuristic and pheromone information) instead of using the original random proportional rule.
- After each cycle, only the pheromones of each step s of the global best solution built until that moment are updated by applying the global updating rule $\tau_s = (1 - \alpha) \cdot \tau_s + \alpha \cdot \Delta\tau_s$, instead of being updated the steps taken by all the ants in the cycle, as it is the case of the AS model.
- During the travel of each ant k , the pheromones of each step s taken by it are locally updated by applying the local updating rule $\tau_s^k = (1 - \rho) \cdot \tau_s^k + \rho \cdot \tau_0$. This update reduces the probability that the remaining ants select this step.

4.2 Local Search

A local search is proposed to be added to the method in Section 3, focused on enhancing the interpretability of each complete solution.

It should be noted that the wideness of the rule coverage is favored in (1) even although the corresponding amplification does not obtain an immediate benefit, with the hope that future amplifications on the same premise provide it. This can lead to final rules with unnecessary labels that not only do not provide any benefit, but even increase the complexity of the rules. Due to this, a local search was introduced in order to explore each final rule and analyze the suitability of each label in the antecedent. Concretely, the local search can be described as follow:

For each rule R^i of the final model:

For each step $s : < i, j, k >$ of the path:

Let $R^{i'}$ be equal to R^i but $LX_{j,k}$

If $cond$ then

Replace R^i with $R^{i'}$

Remove step s from the path

where $cond$ are the conditions to consider $LX_{j,k}$ as unnecessary and comprise:

1. $I(R^{i'}) \geq I(R^i)$
2. the SIR remains covered after replacing R^i with $R^{i'}$

3. if R^i is an excepted rule, none of its exceptions covers the AZ delimited by s
4. if R^i is an exceptional rule of R^j , the rules that could overlap with R^i in the AZ delimited by s are all (if any) exceptional rules of R^j too.

This local search is launched after each construction of a solution and before the pheromone update. Besides, the resultant rule will be deleted if it subsumes in the whole rule base.

4.3 Candidate List

An usual problem of metaheuristics, in general, and ACO paradigm, in particular, is their computational cost. Trying to mitigate this drawback, it is proposed here to use a subset of candidate *feasible* steps among which the next step is selected, with the aim to reduce the number of evaluations of the condition of feasible step, which is the main contribution to the computational cost of the algorithm.

In order to do that, for each state of an ant, a number λ of candidate feasible steps will be considered in the selection of the next step (if the number of feasible steps is less than λ , all of them will be considered). A random proportional rule is used for completing the list of candidate feasible steps, which randomly selects a step to be included in the candidate list proportionally to its current pheromone level. Specifically, the probability of a step of being included in the candidate list is given by (8), that gives the probability with which an ant k at the state S chooses to take the step s^* ,

$$p_k(S, s^*) = \begin{cases} \frac{\tau_{s^*}}{\sum\limits_{s \in J_k(S)} \tau_s}, & \text{if } s^* \in J_k(S) \\ 0 & \text{otherwise} \end{cases} \quad (8)$$

where τ_s is the pheromone associated with step s and $J_k(S)$ is the set of possible steps —feasible and not feasible ones— but those tabu-listed that can be taken by ant k at the state S .

5 Experimental Results

Several experiments were conducted in order to evaluate the performance of the extensions proposed in Section 4. With this aim, the performance of each of them was separately evaluated and then evaluated altogether. The results were compared with those obtained with the original method described in Section 3. Each setup of the algorithm was run 10 times and the averaged results were obtained.

The parameters were set as $\gamma = 0.25$, $\delta = 0.5$, and a number of ants $M = 5$. For the random proportional rule the parameter $\beta = 1$ was used. In the AS model, the pheromone decay parameter was set as $\rho = 0.1$ and, in the ACS model, the parameters $\alpha = \rho = 0.1$ and $q_0 = 0.9$ were used. The number of

Table 1. Automatically generated compound rule base RB^3 with $n/p_i = 3/7$

- $$\begin{aligned}
R^1 : X_1 &\in \{PS, PL\}, X_2 \in \{NS, PS\}, X_3 \in \{NL, Z\} \rightarrow NM \\
R^2 : X_1 &= PL, X_2 \in \{NL, NM, Z, PL\}, X_3 \in \{NS, PM\} \rightarrow Z \\
R^3 : X_1 &= NS, X_2 \in \{NS, PM\}, X_3 \in \{NS, Z\} \rightarrow PS \\
&\text{exc: } X_2 = PM, X_3 = NS \rightarrow PS \\
R^4 : X_1 &= NS, \neg X_2 \in [NS, PS], X_3 \in \{NL, NM, PL\} \rightarrow PM \\
&\text{exc: } X_2 \geq PM, X_3 = NL \rightarrow PM \\
R^5 : X_1 &\in \{NM, PM\}, X_2 \geq NS, X_3 \in \{NL, NM, Z, PS\} \rightarrow PL \\
&\text{exc: } X_1 = NM, X_2 \in \{NS, PM, PL\}, X_3 = NM \rightarrow PL \\
&\text{exc: } X_1 = NM, X_2 = Z, X_3 \in \{NL, PS\} \rightarrow PL \\
R^6 : X_1 &\leq NM, \neg X_2 \in \{NM, PS\}, X_3 \in \{NM, NS, PM\} \rightarrow PL \\
&\text{exc: } X_1 = NM, X_2 \in \{NS, PM, PL\}, X_3 = NM \rightarrow PL \\
&\text{exc: } X_2 \in \{NL, Z, PL\}, X_3 = NS \rightarrow PL
\end{aligned}$$

cycles was bounded to $NC_{max} = 80$ and a stagnation condition was satisfied and the corresponding run finished if all the ants in a cycle built the same set of compound rules.

An automatic generator was designed to provide compound rule bases with exceptions with different dimensionalities —number of input variables, n — and cardinalities for the fuzzy domains—the same cardinality p_i for all the domains. This generator of compound rule bases randomly selects a set of labels for each premise in the antecedent of the rules with the restriction that the final model contains neither conflicting nor subsumed rules. Then, some of these rules are excepted by randomly generated exceptional compound rules.

The ACO algorithm was applied to two n/p_i setups, namely: 2/9, and 3/7. For each setup, three different rule bases were generated (RB^1 , RB^2 , and RB^3). Table 1 shows the third automatically generated rule base RB^3 with $n/p_i = 3/7$, that describes 98 single rules with only 12 compound rules (including 6

Table 2. Results obtained from the variety of versions of the proposed method

	BR_1	BR_2	BR_3
$n = 2, p_i = 9$	(4.40) $\langle 10.0 \rangle \{49.0\}$	(3.75) $\langle 9.0 \rangle \{47.0\}$	(3.85) $\langle 9.0 \rangle \{40.0\}$
AS	(4.40/0.42) $\langle 10.9 \rangle [140s]$	(3.61/0.27) $\langle 9.4 \rangle [130s]$	(3.47/0.26) $\langle 9.1 \rangle [38s]$
ACS	(3.78/0.14) $\langle 11.0 \rangle [149s]$	(3.38/0.03) $\langle 9.0 \rangle [120s]$	(2.96/0.03) $\langle 8.5 \rangle [114s]$
ACS+LS	(3.43/0.09) $\langle 11.0 \rangle [152s]$	(3.27/0.06) $\langle 9.2 \rangle [126s]$	(2.71/0.02) $\langle 8.1 \rangle [120s]$
ACS $_{\lambda=5}$	(3.76/0.10) $\langle 10.3 \rangle [99s]$	(3.35/0.11) $\langle 9.5 \rangle [95s]$	(3.11/0.11) $\langle 8.6 \rangle [81s]$
ACS $_{\lambda=5}$ +LS	(3.38/0.03) $\langle 11.0 \rangle [97s]$	(3.11/0.08) $\langle 9.4 \rangle [97s]$	(2.78/0.13) $\langle 8.5 \rangle [84s]$
$n = 3, p_i = 7$	(4.50) $\langle 10.0 \rangle \{88\}$	(5.88) $\langle 13.0 \rangle \{145\}$	(5.54) $\langle 12.0 \rangle \{98\}$
AS	(5.13/0.74) $\langle 13.2 \rangle [618s]$	(6.18/0.72) $\langle 16.0 \rangle [805s]$	(4.50/0.47) $\langle 11.0 \rangle [351s]$
ACS	(4.77/0.24) $\langle 12.3 \rangle [500s]$	(6.46/0.38) $\langle 15.1 \rangle [791s]$	(4.40/0.31) $\langle 11.2 \rangle [443s]$
ACS+LS	(4.20/0.25) $\langle 12.0 \rangle [485s]$	(5.69/0.33) $\langle 13.7 \rangle [726s]$	(3.91/0.35) $\langle 11.2 \rangle [360s]$
ACS $_{\lambda=10}$	(4.23/0.36) $\langle 11.9 \rangle [306s]$	(5.74/0.47) $\langle 15.9 \rangle [423s]$	(3.44/0.15) $\langle 10.8 \rangle [262s]$
ACS $_{\lambda=10}$ +LS	(3.58/0.34) $\langle 11.8 \rangle [297s]$	(5.18/0.30) $\langle 15.5 \rangle [399s]$	(3.24/0.16) $\langle 11.0 \rangle [236s]$

exceptional rules), includes among them all the extended relational operators, and provides an accumulated complexity equals to 5.54.

Table 2 summarizes the obtained results. For each setup n/p_i , the first row shows the data related with each initial fuzzy model used as the input of the ACO algorithm, and the remaining rows show the averaged results related with the different versions of the method: ‘AS’ corresponds to the original ACO algorithm presented in Section 3 that uses the AS model; ‘ACS’ implement the ACS model proposed in Section 4.1; ‘ACS+LS’ applies the local search proposed in Section 4.2 to the ACS version; ‘ $ACS_{\lambda=x}$ ’ corresponds to the ACS model with the candidate list proposed in Section 4.3 using x candidate steps; and, finally, ‘ $ACS_{\lambda=x}+LS$ ’ join altogether the ACS model, the local search and the candidate list.

That table shows, for each fuzzy model to which the ACO algorithm was applied, the following items:

- in parenthesis, the averaged accumulated complexity of the—input or output—fuzzy model and the standard deviation. This accumulated complexity equals to the sum of the complexity of the rules in the fuzzy model and, therefore, it is inverse to the interpretability of a rule base defined in (6),
- in angular parenthesis, the number of compound rules of the —input or output—fuzzy model,
- in braces, the number of initial single rules of the fuzzy model provided as the input of the ACO algorithm, and
- in brackets, the number of seconds used by the ACO algorithm to obtain a solution.

It must be stressed that the ACO algorithm does not start from the set of compound rules generated automatically, but from the set of singles rules them contain. Regarding this, although it is not assured that the generator of compound rule bases provides the optimal description that cover the underlying set of single rules, it at least generates reasonable good descriptions, on the basis that the ratio NIR/NCR —number of initial rules vs. number of compound rules (including exceptional rules)—is high. For example, for $n/p_i = 3/7$ the ratios were 88/10, 145/13, and 98/12 (see Table 2). Due to these facts, we set as a goal to obtain with the ACO algorithm, at least, a compound rule base with the same interpretability than the automated generated one.

5.1 Original Method

Regarding the original ACO algorithm ‘AS’, it can be observed that, although it achieves rule bases with better interpretability than the initial set of single rules, it did not reach our goal of finding solutions, on average, as interpretable as the automatically generated rule bases in all the cases. The original ACO algorithm provides a higher accumulated complexity than the initial compound rule bases for RB^3 and RB^4 with $n/p_i = 3/7$. Besides, according to the standard deviations, the variability of the results is also considerable, due to an inadequate exploration versus exploitation balance that leads to local optima. Therefore, it is justified the need to include some improvements in order to reach our goal.

5.2 ACS Model

The inclusion of this extension yields a double benefit. On one hand, it achieves a reduction on the averaged accumulated complexity of the rule bases with respect to the solutions obtained with the ‘AS’ version in all but one case. On the other hand, the variability of the results is also significantly reduced, providing a more robust algorithm. Besides, the computational cost is roughly maintained, excepting the case for RB^3 with $n/p_i = 2/9$, for which the incidental premature convergence of the AS model reduced the time for obtaining a solution.

Nevertheless, still for the same cases than in Section 5.1 (RB^3 and RB^4 with $n/p_i = 3/7$) the interpretability of the solutions provided by the ‘ACS’ version is, on average, worse than that of the corresponding original fuzzy model. Therefore, despite this extension provides important benefits, it is not enough by itself.

5.3 Local Search

In all the cases, the local search clearly increases the interpretability results with respect to the ‘ACS’ version. Furthermore, the inclusion of this extension allows to reach the goal of finding solutions, on average, as interpretable as those of the initial fuzzy models (in fact, more interpretable than them).

Of course, the extra-evaluation derived from the local search increases the computational cost. However, the final averaged computational cost of the algorithm shows different results depending on the dimensionality of the rule bases. On one hand, the final computational cost increases with the local search for $n/p_i = 2/9$, although this increase is almost negligible (5.66% on average, meaning 100% increasing the cost twice). On the other hand, the final computational cost decreases in a higher extent for $n/p_i = 3/7$ (9.98% on average, meaning 50% reducing the cost by the half). This latter result can be explained since the local search allows to increase the focus of the ACO process on the best solutions and it compensates the cost of the local search process.

5.4 Candidate List

In the experiments of this extension, we set $\lambda = 5$ for $n/p_i = 3/7$ and $\lambda = 10$ for $n/p_i = 2/9$. As it was mentioned in Section 4.3, the goal of using a list of candidate feasible steps is not to increase the interpretability of the results but to diminish the computational cost without reducing the interpretability results in excess.

Taken this into account, the candidate list extension exceeds the expectations when its results are compared with the ‘ACS’ version. By restricting the number of evaluations of the feasible-step condition, this extension does not only achieve the expected reduction in the computational cost, but it also increases the interpretability of the obtained fuzzy models in almost all the experiments. This latter effect shows an intensification of the exploitation against the exploration since, although it is favored the presence of the better steps (in terms of pheromone levels) in the list, their presence is not guaranteed, which favors the exploration of new paths.

Besides, this extension allows to improve again the interpretability of the initial rule bases in all the cases.

Table 3. Rule base obtained with the ‘ $\text{ACS}_{\lambda=10}+\text{LS}$ ’ version

$R^{11} : X_1 \in \{Z, PS, PL\} \rightarrow Z$
exc: $X_2 \in \{NS, PS\} \rightarrow NM$
$R^5 : X_1 = NS \rightarrow PM$
exc: $X_2 = NS \rightarrow PS$
exc: $X_2 \geq PM, X_3 = NL \rightarrow NM$
exc: $X_3 = Z \rightarrow PS$
exc: $X_2 = PM, X_3 = NS \rightarrow Z$
$R^1 : X_1 \in \{NL, NM, PM\} \rightarrow PL$
exc: $X_1 = NM, X_2 \in \{NS, PM, PL\}, X_3 = NM \rightarrow Z$
exc: $X_2 \in \{NL, Z, PL\}, X_3 = NS \rightarrow NL$
exc: $X_1 = NM, X_2 = Z, X_3 \in \{NL, PS\} \rightarrow PS$

5.5 Application of the Extensions Altogether

Finally, when the extensions are applied altogether, the interpretability improvement with respect to both the ‘AS’ and the ‘ACS’ versions is clear in all the experiments. Besides, our goal of obtaining compound rule bases with, at least, the same interpretability than the initial fuzzy models is reached, even achieving in all the cases an increase in the interpretability level.

The computational cost is also reduced (except for the prematurely converged case RB^3 with $n/p_i = 2/9$). This is mainly due to the use of the candidate list.

As an illustrative example, from the 98 initial single rules contained in the compound rule base RB^3 with $n/p_i = 3/7$ shown in Table 1, the ‘ $\text{ACS}_{\lambda=10}+\text{LS}$ ’ version achieved the fuzzy model shown in Table 3 with only 11 compound rules (including 8 exceptional rules), and which reduces the accumulated complexity to 3.00. It can be observed that the rules obtained with the ACO algorithm uses a simpler syntax in their antecedents than the initial fuzzy model, providing a more interpretable description of the model.

6 Conclusions

An initial ant colony optimization algorithm was proposed by the authors in [5] that tries to increase the interpretability of initial single fuzzy rule bases by searching for good combinations of compound rules with exceptional rules. Despite this original algorithm achieved interpretable models, it still was not able to find near to optimal descriptions in general. We propose here several extensions focused on either increasing the interpretability results of the original algorithm or reducing its computational cost.

With the aim to evaluate the suitability of the proposed extensions, we implemented a generator of random fuzzy models described by means of compound rules with exceptional rules. Since these fuzzy models, even though not optimal, provides good descriptions in the sense of high ratios NIR/NCR —number of single rules vs. number of compound rules—we consider as a goal to achieve with our algorithm descriptions as interpretable as the randomly generated fuzzy models.

As a conclusion, all the proposed extensions provide some benefit and the highest performance was obtained when all the extensions were jointly applied. All the models provided by this last version not only reach the interpretability levels of the randomly generated compound rule bases, but they improve them.

Acknowledgments. This work has been supported by the Research Project TIN2006-03122.

References

1. Casillas, J., Cordón, O., Herrera, F., Magdalena, L. (eds.): Accuracy Improvements in Linguistic Fuzzy Modelling. Studies in Fuzziness and Soft Computing, vol. 129. Springer, Heidelberg (2003)
2. Castro, J., Castro-Schez, J., Zurita, J.: Learning maximal structure rules in fuzzy logic for knowledge acquisition in expert systems. *Fuzzy Sets Syst.* 101, 331–342 (1999)
3. Carmona, P., Castro, J., Zurita, J.: FRIwE: Fuzzy rule identification with exceptions. *IEEE Trans. Fuzzy Syst.* 12(1), 140–151 (2004)
4. Carmona, P., Castro, J., Zurita, J.: Learning maximal structure fuzzy rules with exceptions. *Fuzzy Sets Syst.* 146(1), 63–77 (2004)
5. Carmona, P., Castro, J.: An Ant Colony Optimization plug-in to enhance the interpretability of fuzzy rule bases with exceptions. In: Analysis and Design of Intelligent Systems Using Soft Computing Techniques. Advances in Soft Computing, vol. 41, pp. 436–444. Springer, Heidelberg (2007)
6. Dorigo, M., Colorni, A., Maniezzo, V.: The ant system: Optimization by a colony of cooperating agents. *IEEE Trans. Syst. Man Cybern. B* 26(1), 29–41 (1996)
7. Dorigo, M., Stützle, T.: Ant Colony Optimization. MIT Press, Cambridge (2004)
8. Dorigo, M., Gambardella, L.: Ant colony system: A cooperative learning approach to the traveling salesman problem. *IEEE Trans. Evol. Comput.* 1(1), 53–66 (1997)

Ant Colony Optimization for Energy-Efficient Broadcasting in Ad-Hoc Networks*

Hugo Hernández, Christian Blum, and Guillem Francès

ALBCOM, Dept. Llenguatges i Sistemes Informàtics
Universitat Politècnica de Catalunya, Barcelona, Spain
`{hhernandez,cblum,gfrances}@lsi.upc.edu`

Abstract. In wireless ad-hoc networks, nodes are generally equipped with batteries, making energy a scarce resource. Therefore, power consumption of network operations is critical and subject to optimization. One of the fundamental problems in ad-hoc networks is broadcasting. In this work we consider the so-called minimum energy broadcast (MEB) problem, which can be stated as a combinatorial optimization problem. We develop an ant colony optimization algorithm for two scenarios: networks in which nodes are equipped with omni-directional, respectively directional, antennas. The results show that our algorithm consistently outperforms other methods for this problem.

1 Introduction

Wireless networks such as ad-hoc and sensor networks are useful in many practical scenarios. While sensor networks find applications, for example, in healthcare and weather forecast, ad-hoc networks are often used, for example, to connect laptops or PDAs. The flexibility and low infrastructure cost they offer make them quite popular, and, as a consequence, they have received much attention from the research community in recent years. Nodes in ad-hoc networks, acting potentially both as routers and hosts, are generally equipped with either omni-directional or directional antennas for sending and receiving information. They have a packet-forwarding capability in order to communicate via shared and limited radio channels. Communication may be performed by one-to-one transmissions (single-hop) or using other nodes as relay stations (multi-hop). In both cases each sender node must adjust its emission power in order to reach the respective receiver node. In cases where energy is supplied by batteries, the network lifetime is limited by the batteries of the wireless devices. Therefore, energy saving is critical in all network operations.

A fundamental problem in ad-hoc networks arises when one node is required to transmit data to all other nodes of the network. This scenario is known as

* This work was supported by grants TIN2005-08818 (OPLINK) and TIN2007-66523 (FORMALISM) of the Spanish government, and by the EU project FRONTS (FP7-ICT-2007-1). Christian Blum acknowledges support from the *Ramón y Cajal* program of the Spanish Ministry of Science and Technology, whereas Guillem Francès acknowledges support from a UPC research grant.

broadcasting. In this work we study this problem for two cases. In the first one, each network node is equipped with an omni-directional antenna. However, omni-directional antennas often waste energy, for example, when information must only be sent to one neighboring node. Therefore, we also study the case of directional antennas, which additionally provide the advantage of reducing the interference with other nodes due to the reduced submission area.

As most other works in this area we consider the following wireless communication model (see [1]). Given a set of network nodes V , each node $i \in V$ can choose an emission power p_i such that $0 \leq p_i \leq p_{\max}$, where p_{\max} is the maximum emission power possible. By setting p_{\max} to ∞ we ensure that broadcasting is always possible. Signal power diminishes at a rate proportional to $r^{-\alpha}$, where r is the distance to the signal source, and α is a parameter that, depending on the environment, takes typically values between 2 and 4. In our work we choose (as in most other works; [2]) $\alpha = 2$. A sender node i is able to successfully transmit a signal to a receiver node j if $p_i \geq k \cdot d(i, j)^\alpha$, where p_i is the emission power, $d(i, j)$ is the Euclidean distance between i and j , and k is the receiving node's power threshold for signal detection which is usually normalized to 1.

In the case of directional antennas we use an idealized model (as used, for example, in [2]) in which we assume that the transmitted energy is concentrated uniformly in a beam of width θ , that is, we neglect fading effects at the borders of the beam. We assume that the beam-width θ can be chosen for each antenna so that $\theta_{\min} \leq \theta \leq 360$. As in [2] we chose $\theta_{\min} = 30$. Furthermore, we assume that each antenna beam can be pointed in any desired direction in order to provide connectivity to a set of the nodes that are within communication range and within the sector covered by the beam. The energy spent by a node i transmitting to a node j at a beam-width of θ_i is:

$$p_{ij}^{\theta_i} := \frac{\max\{\theta_i, \theta_{\min}\}}{360} \cdot d(i, j)^\alpha \quad (1)$$

This shows that a node i equipped with a directional antenna only uses 1/12 of the energy used by an omni-directional antenna to transmit information to just one other node j .

1.1 Minimum Energy Broadcast (MEB)

The MEB problem is an NP -hard optimization problem [3] both in case of omni-directional and directional antennas. It can be stated as follows. Given is a set V of nodes with fixed positions in a 2-dimensional area. Introducing a directed link (i, j) between all (ordered) pairs $i \neq j$ of nodes such that $d(i, j)^\alpha \leq p_{\max}$, where $d(i, j)$ is the Euclidean distance between i and j , induces a directed network $G = (V, E)$. In the following we first deal with the case of omni-directional antennas. Given a source node $s \in V$, one must find emission powers for all nodes such that a broadcast from s to all other nodes is possible, and such that the sum of all emission powers is minimal. This corresponds to finding a directed spanning tree $T = (V, E_T)$ with root node s in G such that function $P_o()$ is minimized:

$$P_o(T) := \sum_{i \in V} \max_{(i,j) \in E_T} d(i,j)^\alpha \quad (2)$$

In the case of directional antennas, in addition to an emission power, a beam-width θ_i and a beam direction must be chosen for each node $i \in V$. Again, this corresponds to finding a directed spanning tree $T = (V, E_T)$ with root node s in G such that function $P_d()$ is minimized:

$$P_d(T) := \sum_{i \in V} \frac{\max\{\theta_i, \theta_{\min}\}}{360} \cdot \max_{(i,j) \in E_T} d(i,j)^\alpha , \quad (3)$$

where θ_i is set to the minimum beam-width possible such that all children of i in T are reached. The beam direction follows automatically from the known locations of all the children of i in T .

1.2 Existing Work

The MEB problem in case of omni-directional antennas has been tackled with centralized heuristics as, for example, [4,5,6]. The most popular constructive technique is the *broadcast incremental power* (BIP) algorithm by Wieselthier et al. [4]. Moreover, local search methods including tree-based methods such as [7,8] and power-based methods such as [9] have been developed. More recently the MEB problem was also tackled by metaheuristics [10,11,12,13]. The case of directional antennas is less studied. Constructive algorithms include, for example, the version of BIP for directional antennas: DBIP [2]. Other approaches can be found, for example, in [14,15]. Finally, in [16] is proposed a mixed integer linear programming (MILP) formulation that solves the case of directional antennas, which also includes—in case $\theta_{\min} = 360$ —the case of omni-directional antennas. A comprehensive survey on existing work is given in [17].

Our paper is organized as follows. First, a detailed description of our algorithm proposal will be given in Sec. 2. In Sec. 3 we present an experimental evaluation of our algorithm, comparing the results to recent techniques. Finally, in Sec. 4 we provide conclusions and an outlook to future work.

2 The Algorithm

In this work we propose an ant colony optimization (ACO) algorithm [18] for the MEB problem. In the following we give an algorithm description that covers both the case of omni-directional and the case of directional antennas. Differences between both cases will be pointed out when necessary.

Local Search: r-shrink. The local search procedure *r-shrink* was originally developed in [9] for the case of omni-directional antennas. Here we use an adaptation that can also be applied for directional antennas. Given a solution $T = (V, E_T)$ and a parameter $r \leq |V| - 1$ as input, our version of *r-shrink* works as follows. First, a permutation of all nodes is produced. Nodes with $k \geq r$ children

Algorithm 1. Variable neighborhood descent (VND)

```

1: INPUT: the network  $G = (V, E)$ , a source node  $s \in V$ , a spanning tree
    $T = (V, E_T)$  of  $G$  rooted in  $s$ , a parameter  $r_{\max}$ 
2:  $r := 1$ 
3: while  $r \leq r_{\max}$  do
4:    $T' := r\text{-shrink}(T)$ 
5:   if  $P(T') < P(T)$  then  $T := T'$  and  $r := 1$  else  $r := r + 1$ 
6: end while
7: OUTPUT: a (possibly) improved tree  $T$ 

```

are treated as explained in the following, in the order given by the permutation. When a node i is treated, first, the children of i are ordered in a decreasing manner concerning the emission power reduction achieved by disconnection. Note that an emission power reduction of i may result from a possible distance reduction and, in the case of directional antennas, also from a possible beam-width reduction. Then the first r children are disconnected from i , and the algorithm tries to reconnect them to any of their non-descendants in the best way possible, that is, in a way that is least energy consuming.

In [9] only the 1-shrink procedure was experimentally evaluated. In contrast, we decided to utilize the general r -shrink procedure within a variable neighborhood descent (VND) algorithm [19], which is outlined in Alg. 1. Note that $P()$ stands either for $P_o()$ (in the case of omni-directional antennas) or $P_d()$ (in the case of directional antennas). The VND algorithm requires an appropriate setting of the parameter r_{\max} (see Sec. 3).

ACO for the MEB Problem. The specific ACO algorithm that we implemented for the MEB problem is a *MAX-MIN* Ant System (*MMAS*) in the Hyper-Cube Framework [20]. It works roughly as follows. At each iteration $n_a = 10$ artificial ants construct a tree rooted at the source node s . Local search is applied to each of these trees. The pheromone model \mathcal{T} used by our ACO algorithm contains a pheromone value τ_e for each link $e \in E$. After the initialization of the variables T^{bs} (i.e., the best-so-far solution), T^{rb} (i.e., the restart-best solution), and cf (i.e., the convergence factor), all the pheromone values are set to 0.5. At each iteration, after the generation of solutions, some of them are used for updating the pheromone values. The details of the algorithmic framework shown in Alg. 2 are explained in the following.

ConstructBroadcastTree(G, s): A solution construction starts with the partial solution $S = (V_S, E_S)$ where $V_S := \{s\}$ and $E_S := \emptyset$. Remember that s is the source node of the directed spanning tree to be constructed. Henceforth we denote by $\overline{V_S}$ the set of nodes which are not included in the current partial solution, that is, $\overline{V_S} := V \setminus V_S$. At each construction step, one link (and one node) is added to the current partial solution. The set E_{add} of potential links that can be added to S is defined as follows: $E_{\text{add}} := \{(i, j) \in E | i \in V_S, j \in \overline{V_S}\}$. In words, E_{add}

Algorithm 2. ACO for the MEB problem

```

1: INPUT: the network  $G = (V, E)$  and a source node  $s \in V$ 
2:  $T^{bs} := \text{NULL}$ ,  $T^{rb} := \text{NULL}$ ,  $cf := 0$ ,  $bs\_update := \text{FALSE}$ 
3: forall  $e \in E$  do  $\tau_e := 0.5$  end forall
4: while termination conditions not satisfied do
5:   for  $j = 1$  to  $n_a$  do
6:      $T^j := \text{ConstructBroadcastTree}(G, s)$ 
7:      $T^j := \text{LocalSearch}(T^j)$ 
8:   end for
9:    $T^{ib} := \text{argmin}\{f(T^1), \dots, f(T^{n_a})\}$ 
10:   $\text{Update}(T^{ib}, T^{rb}, T^{bs})$ 
11:   $\text{ApplyPheromoneValueUpdate}(cf, bs\_update, T, T^{ib}, T^{rb}, T^{bs})$ 
12:   $cf := \text{ComputeConvergenceFactor}(T, T^{rb}, T^{bs})$ 
13:  if  $cf \geq 0.99$  then
14:    if  $bs\_update = \text{TRUE}$  then
15:      forall  $e \in E$  do  $\tau_e := 0.5$  end forall
16:       $T^{rb} := \text{NULL}$ ,  $bs\_update := \text{FALSE}$ 
17:    else
18:       $bs\_update := \text{TRUE}$ 
19:    end if
20:  end if
21: end while
22: OUTPUT:  $T^{bs}$ 

```

consists of those links whose source node is in S and whose goal node is not in S . From these links, one link is chosen according to the following probabilities:

$$\mathbf{p}(e) := \frac{\tau_e \cdot \eta(e)}{\sum_{e' \in E_{\text{add}}} \tau_{e'} \cdot \eta(e')} , \quad (4)$$

where $\eta(e)$ is the heuristic information of a link $e = (i, j)$ which is computed as follows: $\eta(e) := (P_o(S') - P_o(S))^{-1}$ (respectively, $\eta(e) := (P_d(S') - P_d(S))^{-1}$ in the case of directional antennas), where $S' = (V_S \cup \{j\}, E_S \cup \{e\})$. In words, the heuristic information accounts for the increase of emission power spent by the partial solution when adding link e . After choosing a link $e = (i, j)$ for the expansion of the current partial solution S , all the other links of E_{add} (if any) that can be added to S without any further increase of emission powers are also added to S (in addition to e). For example, in the case of omni-directional antennas, this concerns all links $e' = (i, l) \in E_{\text{add}}$ with $d(i, l) \leq d(i, j)$. In the case of directional antennas, the node l is additionally required to be within the beam implicitly defined by all links $e(i, *)$ that are already in S (including $e = (i, j)$).

As E_{add} might contain quite a lot of bad-quality links (especially at the beginning of the construction process) we decided to study also ways of restricting E_{add} . In the following we refer to the use of the unrestricted set E_{add} as *mode 1*.

In contrast, *mode 2* works as follows. $\forall j \in \overline{V_S}$ let $E_{\text{add},j} := \{(*, j) \in E_{\text{add}}\}$, that is, $E_{\text{add},j}$ contains all links of E_{add} with goal node j . Moreover, let $b(E_{\text{add},j}) := \text{argmax}_{e \in E_{\text{add},j}} \{\eta(e)\}$, that is, $b(E_{\text{add},j})$ is the link of $E_{\text{add},j}$ with the best heuristic information value. Given these definitions, in *mode 2* set E_{add} is restricted as follows: $E_{\text{add}}^{\text{m2}} := \cup_{j \in \overline{V_S}} \{b(E_{\text{add},j})\}$, that is, only the best link to each node in $\overline{V_S}$ is considered. In the following, let $l := \text{argmax}_{j \in \overline{V_S}} \{\eta(b(E_{\text{add},j}))\}$. Note that l can be regarded as the *best node* in $\overline{V_S}$. Then, the restriction of set E_{add} in *mode 3* is obtained as follows: $E_{\text{add}}^{\text{m3}} := \{(*, l) \in E_{\text{add}}\}$, that is, $E_{\text{add}}^{\text{m3}}$ contains all the links of E_{add} that have l (the best node) as goal node. In addition to the 3 modes outlined above we tested a fourth mode that is obtained by assigning *mode 2* and *mode 3* each to half of the ants used by the algorithm.

LocalSearch(T^j): In the case of omni-directional antennas, solutions constructed by the ants may contain nodes whose emission powers can be reduced without destroying the broadcast property of the solution. Therefore, in case of omni-directional antennas, we first apply the so-called SWEEP procedure (see [4]) in order to detect and fix these cases. Then, the VND algorithm as outlined before is applied in both antenna cases.

Update(T^{ib}, T^{rb}, T^{bs}): In this procedure T^{rb} and T^{bs} are set to T^{ib} (i.e., the iteration-best solution), if $P_o(T^{ib}) < P_o(T^{rb})$ and $P_o(T^{ib}) < P_o(T^{bs})$ (respectively, $P_d(T^{ib}) < P_d(T^{rb})$ and $P_d(T^{ib}) < P_d(T^{bs})$ in the case of directional antennas).

ApplyPheromoneUpdate($cf, bs_update, T, T^{ib}, T^{rb}, T^{bs}$): Our ACO algorithm may use three different solutions for updating the pheromone values: (i) the iteration-best solution T^{ib} , (ii) the restart-best solution T^{rb} and, (iii) the best-so-far solution T^{bs} . Their influence depends on the convergence factor cf , which provides an estimate about the state of convergence of the system. To perform the update, first an update value ξ_e for each link $e \in E$ is computed: $\xi_e := \kappa_{ib} \cdot \delta(T^{ib}, e) + \kappa_{rb} \cdot \delta(T^{rb}, e) + \kappa_{bs} \cdot \delta(T^{bs}, e)$, where κ_{ib} is the weight of T^{ib} , κ_{rb} the weight of T^{rb} , and κ_{bs} the weight of T^{bs} such that $\kappa_{ib} + \kappa_{rb} + \kappa_{bs} = 1.0$. The δ -function is the characteristic function of the set of links in a tree T , that is, $\delta(T, e) = 1$ if $e \in E(T)$, and $\delta(T, e) = 0$ otherwise. Then, the following update rule is applied to all pheromone values τ_e :

$$\tau_e := \min \{ \max\{\tau_{\min}, \tau_e + \rho \cdot (\xi_e - \tau_e)\}, \tau_{\max} \} ,$$

where $\rho \in (0, 1]$ is the learning rate, set to 0.1. The upper and lower bounds $\tau_{\max} = 0.99$ and $\tau_{\min} = 0.01$ keep the pheromone values always in the range $(\tau_{\min}, \tau_{\max})$, thus preventing the algorithm from converging to a solution. After tuning, the values for κ_{ib} , κ_{rb} and κ_{bs} are chosen as shown in Table 1.

ComputeConvergenceFactor(T, T^{rb}, T^{bs}): This function computes, at each iteration, the convergence factor as

$$cf := \frac{\sum_{e \in E(T^{rb})} \tau_e}{(|E(T^{rb})| - 1) \cdot \tau_{\max}} , \text{ respectively } cf := \frac{\sum_{e \in E(T^{bs})} \tau_e}{(|E(T^{bs})| - 1) \cdot \tau_{\max}} ,$$

Table 1. The schedule used for values κ_{ib} , κ_{rb} and κ_{bs} depending on cf (the convergence factor) and the Boolean control variable bs_update

$bs_update = \text{FALSE}$			$bs_update = \text{TRUE}$
	$cf < 0.7$	$cf \in [0.7, 0.9)$	$cf \geq 0.9$
κ_{ib}	2/3	1/3	0
κ_{rb}	1/3	2/3	1
κ_{bs}	0	0	0

if $bs_update = \text{FALSE}$, respectively if $bs_update = \text{TRUE}$. Here, τ_{\max} is the upper limit for the pheromone values. The convergence factor cf can therefore only assume values between 0 and 1. The closer cf is to 1, the higher is the probability to produce the solution T^{rb} (or T^{bs} analogously).

3 Experimental Evaluation

We implemented our algorithm in ANSI C++ using GCC 3.2.2 for compiling the software. Our experimental results were obtained on a PC with an AMD64X2 4400 processor and 4 GB of memory. We used the same set of benchmark instances as in [12,13]. This set consists of 30 problem instances with 20 nodes each, and further 30 problem instances with 50 nodes.

Results for Omni-Directional Antennas. First, we conducted tuning experiments concerning parameter r_{\max} of the VND algorithm (see Alg. 1), the construction mode, and a so-called candidate list strategy, which is a mechanism that reduces the set of all possible choices (in a solution construction step) to the best k choices, where k is a parameter. We tested $k \in \{4, 8, \text{all}\}$, where "all" means that no candidate list strategy is used. The best results were achieved with $r_{\max} = |V| - 1$, construction mode 2, and a candidate list of size 8. See [21] for more details.

We applied our algorithm 30 times for 5 seconds to each of the 30 problem instances with 20 nodes and for 20 seconds to each of the 30 problem instances with 50 nodes. Concerning the problem instances with 20 nodes our algorithm produced an optimal solution for each instance in each run. On average our algorithm needed 0.001 seconds to find these solutions. The ELS algorithm published in [13] finds (on average) in 24 cases (out of 30) an optimal solution. Moreover, the average computation time needed by ELS is 0.43 seconds on a computer with a 2.8 GHz Pentium IV processor.

The results for the problem instances with 50 nodes are shown (in comparison to NP [12], ELS [13], ILS [11], and BIP + VND) in Table 2. For comparison reasons we decided to use the same way of presenting the results as in [13]. The first table column contains the instance names. The second table column provides the values of the best solutions known. In case values are not marked with a \leq -sign, they are known to be optimal. For algorithms NP, ELS, and ACO we provide three values: The column headed by **excess** gives the excess (in percent) of the average of the values of the best solutions found in 30 trials

over the optimal (respectively, best known) value. The column with heading **found** provides the number of trials in which the optimal (respectively, best known) value was found. Finally, the column titled **time (s)** contains the average computation times (in seconds) over 30 runs. For ILS we were not able to provide the computation time as it was not given in [13]. For algorithm BIP + VND we only provide the excess over the optimal (respectively, best known) solutions. Finally, the last table row gives averages over all instances. The results show that, first, our algorithm outperforms all other algorithms in terms of solution quality. Only in 2 cases our algorithm is not able to find the optimal (respectively, best known) solution in all trials. On average it is found in 29.7 out of 30 trials. The second-best algorithm, ELS, only finds an optimal (respectively, best known) solution in 17.3 trials on average.

Results for Directional Antennas. We conducted the same tuning experiments as outlined in the case of omni-directional antennas also for the case of directional antennas. With respect to these experiments we chose $r_{\max} = |V| - 1$, mode 1 for solution construction, and a candidate list of size 8.

The results for the instances with 20 nodes are shown in Table 3 and for the instances with 50 nodes in Table 4. The format of these tables is as follows. The first column provides the instance name, the second column the result of DBIP + VND, and the remaining columns give the results of the ACO algorithm. Concerning ACO we provide the following information. The column with heading **best** gives the value of the best solution found in 30 trials, whereas the column titled **deviation** provides the improvement over DBIP + VND (in percent). Furthermore, the column **saving** shows how much emission power (in percent) could be saved in comparison to the case of omni-directional antennas. The remaining columns provide the average of the best solutions found in 30 trials (+ standard deviation), and the average of the computation times over 30 trials (+ standard deviation). The last table row provides the average improvement (over 30 instances) over DBIP + VND, the average emission power saving, and the average computation time over 30 instances.

The results show that both for instances with 20 nodes and instances with 50 nodes the improvement of ACO over DBIP + VND is on average more than 12%. Moreover, the average computation times are very low (0.08 seconds for the small instances, and 4.42 seconds for the big instances). It is interesting to note that the computation times are higher than in the case of omni-directional antennas. This accounts for the fact that the computation of the heuristic information is more complicated for directional antennas. Finally, it is also worth mentioning that the saving of emission power is enormous: on average 85.70% in the case of 20 nodes, and 85.86% in the case of 50 nodes.

For showing the difference between solutions in the case of omni-directional antennas in comparison to solutions for the same instance in the case of directional antennas we graphically present the best solutions that we found in Fig. 1. For example, while node 47 has a very high emission power in the solution for omni-directional antennas, it only sends information to two neighboring nodes in the solution for directional antennas.

Table 2. Results for instances with 50 nodes (omni-directional antennas)

Instance	Best known	NP				ELS				ILS				BIP + VND				ACO	
		excess	found	time (s)	excess	found	time (s)	excess	found	time (s)	excess	found	time (s)	excess	found	time (s)	excess	found time (s)	
p50.00	399074.64	6.22%	0/30	11.4	0.41%	15/30	57.0	3.31%	0/30	9.89%	0.0%	30/30	0.52						
p50.01	≤ 373565.15	3.68%	0/30	7.1	0.16%	5/30	47.0	3.30%	0/30	22.79%	0.0%	30/30	1.97						
p50.02	393641.09	10.11%	0/30	10.3	0.28%	13/30	46.0	12.06%	0/30	17.39%	0.0%	30/30	2.84						
p50.03	316801.09	6.43%	0/30	6.1	1.71%	11/30	57.0	6.48%	0/30	18.43%	0.0%	30/30	0.36						
p50.04	≤ 325774.22	5.22%	0/30	7.5	0.30%	25/30	40.0	7.22%	0/30	17.04%	0.0%	30/30	1.89						
p50.05	382235.90	3.28%	1/30	10.9	0.83%	16/30	31.0	2.57%	0/30	7.69%	0.0%	30/30	3.42						
p50.06	≤ 384438.46	1.19%	5/30	10.2	0.0%	30/30	29.0	0.14%	18/30	13.50%	0.0%	30/30	3.54						
p50.07	≤ 401836.85	6.70%	0/30	8.9	0.54%	24/30	64.0	7.80%	0/30	11.50%	0.0%	30/30	0.54						
p50.08	≤ 334418.45	0.10%	27/30	4.6	0.0%	30/30	29.0	0.0%	30/30	11.43%	0.0%	30/30	0.73						
p50.09	≤ 346732.05	9.20%	0/30	12.9	3.29%	0/30	102.0	11.42%	0/30	15.22%	0.0%	30/30	1.88						
p50.10	416733.45	2.14%	0/30	8.9	1.16%	13/30	40.0	2.05%	0/30	12.80%	0.0%	30/30	0.85						
p50.11	≤ 369869.41	4.34%	0/30	7.7	2.87%	1/30	28.0	4.62%	0/30	9.99%	0.0%	30/30	4.47						
p50.12	≤ 392326.01	3.18%	0/30	13.5	0.57%	7/30	66.0	0.44%	0/30	6.82%	0.0%	30/30	0.42						
p50.13	≤ 400563.83	6.63%	0/30	11.2	0.04%	29/30	74.0	1.20%	0/30	21.16%	0.0%	30/30	1.38						
p50.14	388714.91	0.08%	22/30	6.6	0.34%	3/30	11.0	0.0%	30/30	34.93%	0.0%	29/30	2.26						
p50.15	371634.65	0.49%	0/30	8.1	0.20%	5/30	35.0	1.40%	0/30	9.62%	0.0%	30/30	1.12						
p50.16	≤ 414587.42	5.28%	0/30	15.1	0.30%	26/30	81.0	6.06%	1/30	5.51%	0.0%	30/30	0.48						
p50.17	355937.07	2.17%	1/30	11.9	1.88%	17/30	33.0	4.94%	2/30	9.00%	0.0%	30/30	0.38						
p50.18	376617.33	5.96%	0/30	11.3	0.24%	8/30	65.0	0.98%	12/30	7.28%	0.0%	30/30	0.55						
p50.19	335059.72	2.27%	5/30	9.8	0.0%	30/30	28.0	10.49%	1/30	33.60%	0.0%	30/30	0.39						
p50.20	414708.96	3.14%	0/30	10.4	0.15%	0/30	35.0	6.33%	0/30	9.54%	0.1%	21/30	5.85						
p50.21	≤ 361354.27	2.93%	2/30	10.4	0.0%	30/30	41.0	0.0%	30/30	9.01%	0.0%	30/30	0.29						
p50.22	329043.51	0.0%	30/30	7.2	0.0%	30/30	14.0	4.61%	0/30	25.49%	0.0%	30/30	0.74						
p50.23	383321.04	6.39%	0/30	11.1	0.0%	30/30	109.0	2.47%	0/30	10.95%	0.0%	30/30	1.06						
p50.24	404855.92	5.69%	0/30	10.0	0.07%	17/30	37.0	0.37%	3/30	10.42%	0.0%	30/30	0.53						
p50.25	363200.32	0.0%	30/30	3.2	0.0%	30/30	7.0	0.0%	30/30	24.65%	0.0%	30/30	2.16						
p50.26	406631.51	9.59%	0/30	11.5	2.17%	2/30	60.0	11.63%	0/30	12.67%	0.0%	30/30	2.16						
p50.27	451059.62	4.18%	0/30	9.5	0.18%	22/30	40.0	3.35%	0/30	9.09%	0.0%	30/30	1.50						
p50.28	≤ 415832.44	4.48%	0/30	11.6	0.47%	23/30	78.0	0.28%	0/30	8.63%	0.0%	30/30	0.38						
p50.29	380492.77	0.0%	30/30	5.9	0.08%	27/30	18.0	0.60%	0/30	14.62%	0.0%	30/30	1.92						
	4.03%	5.1/30	9.5	0.61%	17.3/30	46.0	3.89%	5.23/30	46.0	14.37%	0.004%	29.7/30	14.49						

Table 3. Results for instances with 20 nodes (directional antennas)

Instance	DBIP + VND	ACO				
		best	deviation	saving	average	(std.) time (s) (std.)
p20.00	64822.56	63199.93	-2.51%	84.48%	63199.93 (0.00)	0.06 (0.05)
p20.01	76436.97	68398.52	-10.52%	84.70%	68398.52 (0.00)	0.02 (0.01)
p20.02	65690.63	47978.29	-26.97%	85.68%	47978.29 (0.00)	0.07 (0.07)
p20.03	68942.34	57398.42	-16.75%	88.25%	57398.42 (0.00)	0.04 (0.02)
p20.04	69845.03	66887.32	-4.24%	87.04%	66887.32 (0.00)	0.01 (0.01)
p20.05	51662.20	45238.84	-12.44%	84.96%	45238.84 (0.00)	0.04 (0.02)
p20.06	60070.64	51729.84	-13.89%	79.35%	51729.84 (0.00)	0.07 (0.06)
p20.07	55895.63	51571.88	-7.74%	85.16%	51571.88 (0.00)	0.03 (0.02)
p20.08	54367.14	49937.22	-8.15%	87.22%	49937.22 (0.00)	0.15 (0.21)
p20.09	65553.86	59488.68	-9.26%	86.71%	59488.68 (0.00)	0.07 (0.06)
p20.10	56304.30	48060.81	-14.65%	84.83%	48060.81 (0.00)	0.06 (0.06)
p20.11	61803.98	52576.87	-14.93%	81.82%	52576.87 (0.00)	0.06 (0.08)
p20.12	47662.65	46448.20	-2.55%	85.23%	46448.20 (0.00)	0.01 (0.01)
p20.13	73334.40	57650.44	-21.39%	83.35%	57650.44 (0.00)	0.03 (0.02)
p20.14	48983.67	39342.53	-19.69%	86.95%	39342.53 (0.00)	0.12 (0.12)
p20.15	63590.04	63346.56	-0.39%	86.15%	63346.56 (0.00)	0.01 (0.01)
p20.16	84518.63	70879.81	-16.14%	85.37%	70879.81 (0.00)	0.05 (0.03)
p20.17	95860.33	69089.46	-27.93%	81.83%	69089.46 (0.00)	0.06 (0.06)
p20.18	47485.69	46477.60	-2.13%	85.49%	46477.60 (0.00)	0.03 (0.02)
p20.19	72365.45	64180.43	-11.32%	86.09%	64180.43 (0.00)	0.07 (0.03)
p20.20	76200.32	66069.32	-13.30%	83.63%	66069.32 (0.00)	0.07 (0.08)
p20.21	45734.98	43714.14	-4.42%	83.93%	43714.14 (0.00)	0.02 (0.01)
p20.22	49408.07	47693.55	-3.48%	85.49%	47693.55 (0.00)	0.03 (0.02)
p20.23	53427.61	51555.92	-3.51%	84.22%	51555.92 (0.00)	0.02 (0.01)
p20.24	57184.69	49804.65	-12.91%	87.42%	49804.65 (0.00)	0.45 (0.45)
p20.25	97379.75	75132.03	-22.85%	83.43%	75132.03 (0.00)	0.46 (0.40)
p20.26	112701.77	83211.94	-26.17%	81.97%	83211.94 (0.00)	0.06 (0.04)
p20.27	66935.67	52136.30	-22.11%	86.60%	52136.30 (0.00)	0.01 (0.01)
p20.28	55028.32	52897.18	-3.88%	81.06%	52897.18 (0.00)	0.09 (0.05)
p20.29	57176.02	52294.99	-8.54%	82.54%	52294.99 (0.00)	0.06 (0.02)
		-12.16% 84.70%				
		0.08				

Table 4. Results for instances with 50 nodes (directional antennas)

Instance	DBIP + VND	ACO				
		best	deviation	saving	average	(std.) time (s) (std.)
p50.00	67637.45	57143.89	-15.52%	85.68%	57148.02 (2.97)	5.37 (5.12)
p50.01	60089.71	55020.28	-8.44%	85.27%	55022.15 (10.26)	4.33 (2.96)
p50.02	63117.46	55592.02	-11.93%	85.88%	55668.98 (77.17)	8.84 (5.25)
p50.03	49914.53	44151.86	-11.55%	86.06%	44151.86 (0.00)	1.29 (0.83)
p50.04	52739.20	49258.11	-6.61%	84.88%	49281.38 (27.07)	6.54 (5.36)
p50.05	62925.57	54624.04	-13.20%	85.71%	54628.36 (23.65)	3.36 (3.35)
p50.06	59134.80	51440.18	-13.02%	86.62%	51440.18 (0.00)	3.07 (2.17)
p50.07	60529.65	53677.16	-11.33%	86.64%	53677.16 (0.00)	0.84 (0.43)
p50.08	59745.12	53915.98	-9.76%	83.88%	53922.06 (33.31)	6.41 (4.58)
p50.09	55269.83	45721.73	-17.28%	86.81%	45721.73 (0.00)	2.26 (1.55)
p50.10	71832.64	60454.61	-15.84%	85.50%	60472.57 (79.99)	9.94 (5.68)
p50.11	58924.18	50641.36	-14.06%	86.31%	50641.36 (0.00)	1.41 (1.41)
p50.12	71670.15	52365.95	-26.94%	86.65%	52462.37 (150.22)	11.04 (4.95)
p50.13	56645.94	50693.12	-10.51%	87.34%	50693.12 (0.00)	3.32 (2.82)
p50.14	72412.21	58263.43	-19.54%	85.01%	58263.43 (0.00)	4.25 (2.34)
p50.15	55379.00	47647.14	-13.97%	87.18%	47647.14 (0.00)	1.77 (1.25)
p50.16	62602.95	53963.57	-13.81%	86.98%	53963.57 (0.00)	2.39 (1.58)
p50.17	62421.55	50597.19	-18.95%	85.78%	50597.24 (0.22)	4.66 (3.45)
p50.18	59991.97	53211.01	-11.31%	85.87%	53241.50 (56.21)	5.51 (5.21)
p50.19	57547.85	52356.87	-9.03%	84.37%	52387.81 (98.69)	7.37 (4.54)
p50.20	62215.55	55986.52	-10.02%	86.50%	55986.52 (0.00)	1.51 (0.94)
p50.21	57325.62	51102.38	-10.86%	85.86%	51136.06 (25.56)	9.07 (5.48)
p50.22	60044.05	52484.70	-12.59%	84.05%	52484.70 (0.00)	1.96 (1.40)
p50.23	58439.32	51386.64	-12.07%	86.59%	51386.64 (0.00)	5.59 (4.08)
p50.24	68262.05	57834.24	-15.28%	85.71%	57834.24 (0.00)	2.20 (1.52)
p50.25	70594.71	60369.77	-14.49%	83.38%	60406.47 (65.84)	9.93 (5.37)
p50.26	61786.70	56527.63	-8.52%	86.10%	56527.63 (0.00)	5.16 (3.21)
p50.27	66161.21	59330.71	-10.33%	86.85%	59330.71 (0.00)	1.29 (0.53)
p50.28	61571.83	55646.64	-9.63%	86.62%	55646.64 (0.00)	0.72 (0.26)
p50.29	61768.29	54511.41	-11.75%	85.67%	54511.41 (0.00)	1.21 (0.72)
		-12.94% 85.86%				
		4.42				

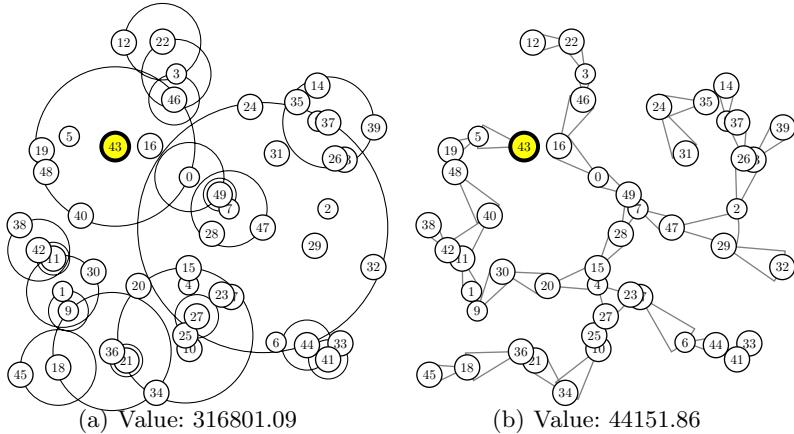


Fig. 1. Best solutions found for instance p50.03. (a) shows the solution in the case of omni-directional antennas, and (b) shows the solution for directional antennas.

4 Conclusions and Future Work

In this work we presented an ant colony optimization algorithm using a sophisticated local search procedure for the minimum energy broadcast problem in static wireless ad-hoc networks. In the case of omni-directional antennas our algorithm outperforms existing metaheuristics and heuristics in solution quality as well as in computation time. In the case of directional antennas we could show that our algorithm greatly improves over the results of a well-known heuristic enhanced by our VND algorithm. In the future we plan to adapt our algorithms to the case of multicasting. Moreover, we plan to develop distributed algorithms that are more practical in real applications.

References

1. Rapport, T.: *Wireless Communications: Principles and Practices*. Prentice Hall, Englewood Cliffs (1996)
2. Wieselthier, J.E., Nguyen, G.D., Ephremides, A.: Energy-aware wireless networking with directional antennas: the case of session-based broadcasting and multicasting. *IEEE Trans. on Mobile Computing* 1(3), 176–191 (2002)
3. Cagalj, M., Hubaux, J.P., Enz, C.: Minimum-energy broadcast in all-wireless networks: NP-completeness and distribution issues. In: Proc. of ACM MobiCom, pp. 172–182. ACM press, New York (2002)
4. Wieselthier, J.E., Nguyen, G.D., Ephremides, A.: On the construction of energy-efficient broadcast and multicast trees in wireless networks. Proc. of INFOCOM 2000 2, 585–594 (2000)
5. Wan, P.J., Calinescu, G., Li, X.Y., Frieder, O.: Minimum-energy broadcast routing in static ad hoc wireless networks. *ACM Wireless Networks* 8(6), 607–617 (2002)
6. Liang, W.: Constructing minimum-energy broadcast trees in wireless ad hoc networks. In: Proc. of ACM MobiHoc 2002, pp. 112–122. ACM press, New York (2002)

7. Li, F., Nikolaidis, I.: On minimum-energy broadcasting in all-wireless networks. In: Proc. of IEEE LCN, pp. 14–16. IEEE press, Los Alamitos (2001)
8. Guo, S., Yang, O.: A dynamic multicast tree reconstruction algorithm for minimum-energy multicasting in wireless ad hoc networks. In: Proc. of IEEE IPCCC, pp. 637–642. IEEE press, Los Alamitos (2004)
9. Das, A.K., Marks, R.J., El-Sharkawi, M., Arabshahi, P., Gray, A.: *r-shrink*: A heuristic for improving minimum power broadcast trees in wireless networks. In: Proc. of GLOBECOM 2003, pp. 523–527. IEEE press, Los Alamitos (2003)
10. Das, A.K., Marks, R.J., El-Sharkawi, M., Arabshahi, P., Gray, A.: The minimum power broadcast problem in wireless networks: an ant colony system approach. In: Proc. of the IEEE CAS Wshp. on Wireless Communications and Networking (2002)
11. Kang, I., Poovendran, R.: Iterated local optimization for minimum energy broadcast. In: Proc. of WiOpt 2005, pp. 332–341. IEEE press, Los Alamitos (2005)
12. Al-Shihabi, S., Merz, P., Wolf, S.: Nested partitioning for the minimum energy broadcast. In: Proc. of LION 2007. Springer, Berlin (2007)
13. Wolf, S., Merz, P.: Evolutionary local search for the minimum energy broadcast problem. In: van Hemert, J., Cotta, C. (eds.) EvoCOP 2008. LNCS, vol. 4972, pp. 61–72. Springer, Heidelberg (2008)
14. Cartigny, J., Simplot-Ryl, D., Stojmenovic, I.: An adaptive localized scheme for energy-efficient broadcasting in ad hoc networks with directional antennas. In: Niemegeers, I.G.M.M., de Groot, S.H. (eds.) PWC 2004. LNCS, vol. 3260, pp. 399–413. Springer, Heidelberg (2004)
15. Guo, S., Yang, O.: Improving energy efficiency for multicasting in ad-hoc networks with directional antennas. In: Proc. of IEEE WiMob 2005, pp. 344–351. IEEE press, Los Alamitos (2005)
16. Guo, S., Yang, O.: Minimum-energy multicast in wireless ad hoc networks with adaptive antennas: MILP formulations and heuristic algorithms. IEEE Trans. on Mobile Computing 5(4), 333–346 (2006)
17. Guo, S., Yang, O.W.W.: Energy-aware multicasting in wireless ad hoc networks: A survey and discussion. Computer Communications 30, 2129–2148 (2007)
18. Dorigo, M., Stützle, T.: Ant Colony Optimization. MIT Press, Cambridge (2004)
19. Hansen, P., Mladenović, N.: Variable neighborhood search: Principles and applications. European Journal of Operational Research 130, 449–467 (2001)
20. Blum, C., Dorigo, M.: The hyper-cube framework for ant colony optimization. IEEE Trans. on Systems, Man and Cybernetics – Part B 34(2), 1161–1172 (2004)
21. Hernández, H., Blum, C., Francès, G.: Ant colony optimization for energy-efficient broadcasting in ad-hoc networks. Technical Report LSI-08-13, LSI, Universitat Politècnica de Catalunya (2008)

Ant Colony Optimization for Genome-Wide Genetic Analysis

Casey S. Greene, Bill C. White, and Jason H. Moore

Dartmouth College, Lebanon, NH, USA

{Casey.S.Greene,Bill.C.White,Jason.H.Moore}@dartmouth.edu

Abstract. In human genetics it is now feasible to measure large numbers of DNA sequence variations across the human genome. Given current knowledge about biological networks and disease processes it seems likely that disease risk can best be modeled by interactions between biological components, which can be examined as interacting DNA sequence variations. The machine learning challenge is to effectively explore interactions in these datasets to identify combinations of variations which are predictive of common human diseases. Ant colony optimization (ACO) is a promising approach to this problem. The goal of this study is to examine the usefulness of ACO for problems in this domain and to develop a prototype of an expert knowledge guided probabilistic search wrapper. We show that an ACO approach is not successful in the absence of expert knowledge but is successful when expert knowledge is supplied through the pheromone updating rule.

1 Introduction

Researchers in the biological and biomedical sciences are now capable of generating enormous amounts of data. In human genetics it is now technically and economically feasible to measure more than one million DNA sequence variations from across the human genome. Here we focus on the single nucleotide polymorphism or SNP which is a single point in a DNA sequence that differs among people. It is anticipated that at least one SNP occurs approximately every 100 nucleotides across the 3×10^9 nucleotide human genome. An important goal in human genetics is the determination of which of the millions of SNPs are useful for predicting who is at risk for common diseases. This “genome-wide” approach is expected to revolutionize the genetic analysis of common human disease. The charge for computer science and bioinformatics is the development of algorithms for the detection and characterization of SNPs which are predictive of human health and disease. Success in this endeavor will be difficult due to nonlinearity in the genotype-to-phenotype mapping relationship that is due, in part, to epistasis or nonadditive gene-gene interactions. The implication of epistasis from a data mining point of view is that SNPs need to be considered jointly in learning algorithms rather than individually. The challenge of modeling attribute interactions has been previously described [1]. Due to the combinatorial magnitude of this problem, intelligent analysis strategies are needed.

1.1 Concept Difficulty

Combining the difficulty of modeling nonlinear attribute interactions with the challenge of attribute selection yields for this domain what Goldberg [2] calls a needle-in-a-haystack problem. That is, there may be a particular combination of SNPs that together with the right nonlinear function are a significant predictor of disease susceptibility. Considered individually they may not look any different than thousands of other SNPs not involved in the disease process. Under these models, the learning algorithm is truly looking for a genetic needle in a genomic haystack. These epistatic interactions are thought to be widespread, perhaps ubiquitous, among risk factors for these common human diseases [3]. A recent report from the International HapMap Consortium [4] suggests that approximately 300,000 carefully selected SNPs may be necessary to capture all of the relevant variation across the Caucasian human genome. Assuming this is true (it is probably a lower bound), we would need to scan 4.5×10^{10} pairwise combinations of SNPs to find a genetic needle. The number of higher order combinations is astronomical.

1.2 Ant Colony Optimization

Ant colony optimization (ACO) is a positive feedback approach to search modeled on the behavior of ants [5]. Ant colony optimization is attractive for the area of human genetics because it is a straightforward population based approach to search which is easily parallelizable. Ant colony systems have previously been applied to the mining of biological data. Parpinelli et al. [6] demonstrate their AntMiner system as a rule discovery method on biological data. Here we begin to develop a probabilistic search wrapper which can be integrated into the publicly available Multifactor Dimensionality Reduction (MDR) software. But is ant colony optimization suitable for a problem like this? Without expert knowledge the answer would seem to be no. There is no reason to expect that an ACO or any other wrapper method will perform better than a random attribute selector because there are no building blocks for this problem when accuracy is used as a metric of quality. The accuracy of any given classifier looks no better than any other with just one of the two correct SNPs in the model. Indeed, we have observed this in the field of genetic programming [7,8]. Subsequent work has shown that by integrating expert knowledge into a genetic programming scheme, it is possible to develop a wrapper that is able to perform better than a random attribute selector [9,10]. Fortunately the ACO metaheuristic is amenable to the inclusion of heuristic information. Work here examines whether or not it is possible to integrate expert knowledge, our heuristic information, into an ACO framework to develop a wrapper which performs well in this domain.

2 The Proposed Ant Colony Optimization Algorithm

Ant colony optimization is a particularly appropriate framework for this problem because of its simplicity and the ease with which expert knowledge can be included. For our work with genetic programming we developed specialized fitness

functions [7], recombination [8] and mutation operators [10]. With ACO it is conceptually much simpler to include expert knowledge. For this ACO metaheuristic we have elected to include expert knowledge as an additional component of the pheromone update rule. When the accuracy of the classifier is identical, ants that choose SNPs with better expert knowledge will contribute more pheromone to those paths than SNPs with lower expert knowledge scores.

2.1 Implementation

This ACO is implemented in C++. For the purposes of this power analysis solutions consist of pairs of attributes. The solution kept as the result is the pair of attributes with the highest balanced accuracy according to MDR (detailed in section 3). MDR analysis is performed through version 0.2.5 of the libmdr open source C library available from www.epistasis.org

2.2 Pheromone Updating with Expert Knowledge

We have discovered in our work with genetic programming that expert knowledge is critical for machine learning algorithms to be successful with this problem [7]. Here we apply principles discovered in our genetic programming work to the ACO arena and structured our pheromone updating rule such that expert knowledge is provided within the pheromone update rule. The pheromone is updated for each SNP, a , according to the following function after the i th update:

$$\tau_{a,i+1} = \tau_{a,i}\rho + \Delta\tau_{a,i} \quad (1)$$

$\Delta\tau_{a,i}$ is the additional pheromone contributed by ants during this update cycle and ρ is the evaporation factor. $\Delta\tau_{a,i}$ is obtained as a combination of the MDR accuracy and the expert knowledge information for each ant, k , of m total ants that contain attribute a :

$$\Delta\tau_{a,i} = \sum_{k=1}^m Q_{a,b}^\alpha E_a^\beta \quad (2)$$

where Q is the MDR accuracy of a model containing that attribute a and the other attribute, b , chosen by ant k . In the case that a and b are the same SNP the MDR accuracy is set to zero to push the metaheuristic away from SNPs that have a strong main effect without an epistatic effect. E is the expert knowledge information for attribute a , and α and β are coefficients that determine the relative weighting of Q and E . In this case Tuned ReliefF (TuRF) weights (see Section 4) are used as the expert knowledge [11]. This update rule is used through u total updates. This serves as an easy to understand pheromone updating rule which incorporates expert knowledge for ACO in the field of genetic analysis. This approach is similar to the use of heuristic information in other ACO approaches [5], except that here the heuristic alters the amount of pheromone deposited instead of modifying the likelihood of an ant selecting a path given pheromone information. This means that the initial search is very exploratory and that good SNPs by our heuristic information should be more heavily selected towards the end as pheromone information accumulates.

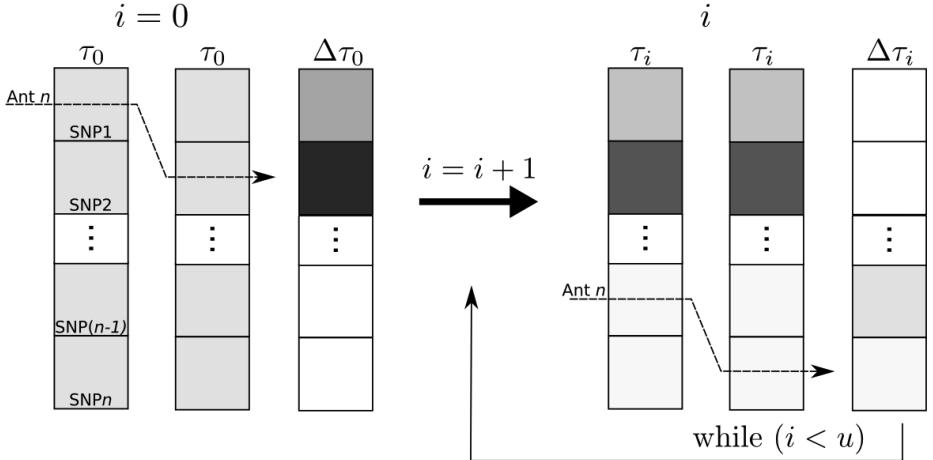


Fig. 1. In our ACO metaheuristic ants explore a pair of SNPs. From that pair of SNPs a $\Delta\tau_i$ is calculated as shown in equation 2 for each SNP. $\Delta\tau_i$ is a combination of the quality of the pair, Q , and the expert knowledge score for that SNP, E . Shading here is representative of the strength of pheromone, τ , which begins evenly distributed and changes with each update, (i).

2.3 Parameter Settings

We restrict the amount of the search space which can be explored to examine how well the algorithm performs given the ability to search a small number of the possible interactions. In each run 5000 total ants explore the search space. This means that at most approximately 1% of the total possible interactions (i.e. the search space) are examined. The power analysis (i.e. how often the correct answer is found) is performed with 250 ants per update for 20 updates, 500 ants per update for 10 updates and 1000 ants per update for 5 updates. The parameter α is fixed at 1 and β is tested at 0, 1, and 2. When β equals 0, the expert knowledge weighting factor becomes 1 and does not affect the updating of the pheromone, thus it is possible to examine the impact of expert knowledge on the ant colony optimization approach in this domain. The evaporation parameter ρ is held constant at 0.5.

3 Multifactor Dimensionality Reduction (MDR) for Attribute Construction

Multifactor dimensionality reduction (MDR) was developed as a nonparametric and genetic model-free data mining strategy for identifying combination of SNPs that are predictive of a discrete clinical endpoint [12,13,14,15]. The MDR method has been successfully applied to detecting gene-gene interactions for a variety of common human diseases including adverse drug reactions [16]. At the heart of the MDR approach is an attribute construction algorithm that creates a

new attribute by pooling genotypes from multiple SNPs. Constructive induction using the MDR kernel is accomplished in the following way. Given a threshold T , a multilocus genotype combination is considered high-risk if the ratio of cases (subjects with disease) to controls (healthy subjects) exceeds or equals T , otherwise it is considered low-risk. Genotype combinations considered to be high-risk are labeled $G1$ while those considered low-risk are labeled $G0$. This process constructs a new one-dimensional attribute with levels $G0$ and $G1$. It is this new single variable that is returned by the MDR function as the quality, Q , for the ACO metaheuristic. Moore et al. [14] describe the MDR method in more detail. Open-source MDR software is freely available from www.epistasis.org

4 Expert Knowledge from Tuned ReliefF (TuRF)

Our goal is to provide an external measure of attribute quality that can be used as expert knowledge for pheromone updating by the ACO metaheuristic. Here the external measure used is statistical, but it can just as easily be biological. There are many statistical and computational methods for determining the quality of attributes. Our goal is to use a method that is capable of identifying attributes that predict class primarily through dependencies or interactions with other attributes. Kira and Rendell [17] developed an algorithm called Relief that is capable of detecting attribute dependencies.

Relief estimates the quality of attributes through a nearest neighbor algorithm that selects neighbors (instances) from the same class and from the different class based on the vector of values across attributes. Weights (W) or quality estimates for each attribute (a) are estimated based on whether the nearest neighbor (nearest hit, H) of a randomly selected instance (R) from the same class and the nearest neighbor from the other class (nearest miss, M) have the same or different values. This process of adjusting weights is repeated for m instances. The algorithm produces weights for each attribute ranging from -1 (worst) to +1 (best). Kononenko [18] improved upon Relief by choosing n nearest neighbors instead of just one. This new ReliefF algorithm has been shown to be more robust to noisy attributes and missing data [19] and is widely used in data mining applications [19].

We developed a modified ReliefF algorithm for the domain of human genetics called Tuned ReliefF (TuRF). We have previously shown that TuRF is significantly better than ReliefF in this domain [11]. The TuRF algorithm systematically removes attributes that have low quality estimates so that the ReliefF values of the remaining attributes can be re-estimated. We apply TuRF as described by Moore and White [11] to each dataset. Here TuRF scores compose the expert knowledge component of the ACO metaheuristic, E .

5 Fisher's Exact Test

Fisher's exact test is a significance test appropriate for categorical count data [20]. The resulting p -value denotes the likelihood that an association of the observed magnitude is likely by chance alone. For our use we arrange the results in a 2x2 contingency table:

Table 1. 2x2 contingency table for power analysis

	Success	Failure
Parameter Set 1 (PS_1)	# Successful with PS_1	# Unsuccessful with PS_1
Parameter Set 2 (PS_2)	# Successful with PS_2	# Unsuccessful with PS_2

With this contingency table we can detect whether the association between success (power) at different parameter settings (PS_1 and PS_2) is likely due to chance alone. The resulting p -value for this test can be interpreted as the likelihood of seeing a difference among powers of the size observed without an association.

6 Data Simulation

The goal of the simulation study is to generate artificial datasets with high concept difficulty to evaluate the power of ACO in the domain of human genetics. We first develop 30 different penetrance functions (i.e. genetic models) that define a probabilistic relationship between genotype and phenotype where susceptibility to disease is dependent on genotypes from two SNPs in the absence of any independent effects. The 30 penetrance functions include groups of five with heritabilities of 0.025, 0.05, 0.1, 0.2, 0.3, or 0.4. These heritabilities range from a very small to a large genetic effect size. Each functional SNP has two alleles with frequencies of 0.4 and 0.6.

Table 2. Penetrance values for an example epistasis model

	AA (0.36)	Aa (0.48)	aa (0.16)
BB (0.36)	0.077	0.656	0.880
Bb (0.48)	0.892	0.235	0.312
bb (0.16)	0.174	0.842	0.106

Table 2 summarizes the penetrance values to three significant digits for one of the 30 models. The values in parentheses are the genotype frequencies. Each of the models is used to generate 100 replicate datasets with a sample size of 1600. Each dataset consists of an equal number of case (disease) and control (no disease) subjects. Each pair of functional SNPs is combined within a genome-wide set of 998 randomly generated SNPs for a total of 1000 attributes. A total of 3,000 datasets are generated and analyzed.

7 Experimental Design and Statistical Analysis

For each set of 100 datasets and for each set of parameters we count the number of times the correct two functional attributes are selected as the best model by our ACO implementation. This count, expressed as a percentage, is an estimate

of the power of the method. This percentage represents how often ACO meta-heuristic finds the answer that we know is present. We compare the significance of power estimates between the methods (e.g. $\beta = 0$, $\beta = 1$, and $\beta = 2$.) by performing fisher's exact test [20]. Results are considered statistically significant when $p \leq 0.05$.

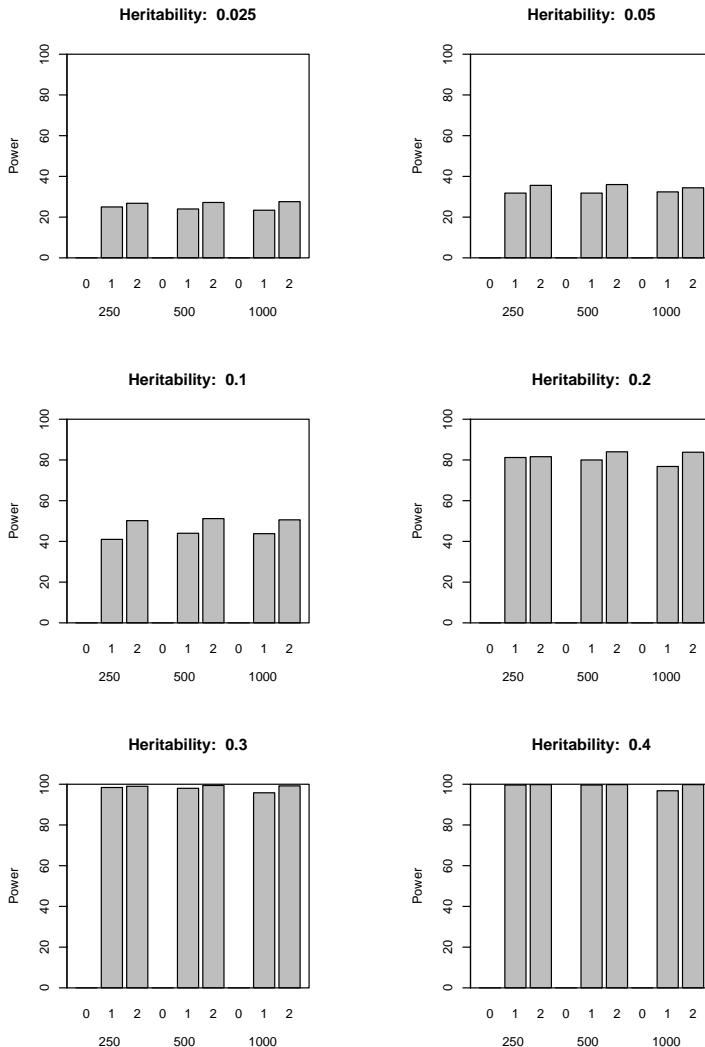


Fig. 2. The average power across heritabilities. Each group of three bars shows one combination of ants and updates (250 ants/20 updates, 500 ants/10 updates, 1000 ants/5 updates respectively). Within the ant and update groups the beta value is 0, 1, and 2 from left to right.

8 Experimental Results

Figure 2 summarizes the average power (% success) for each method. Each bar represents the power averaged over 500 datasets (5 models with 100 datasets each). Power represents the number of times out of 100 that the ACO finds the right two attributes. These results clearly show that the ACO approach is unable to successfully find the correct pair of attributes when expert knowledge is not used as the power is very low for all cases where the expert knowledge weighting parameter, β , is set to zero. These results also show that the ACO metaheuristic is frequently successful when β is one or two, showing the critical need for expert knowledge.

To assess the reliability and robustness of these results quantitatively we use Fisher's exact test (Section 5). As figure 3 shows, the difference between values of β of 0 and values of β of 1 and 2 is highly significant in all circumstances.

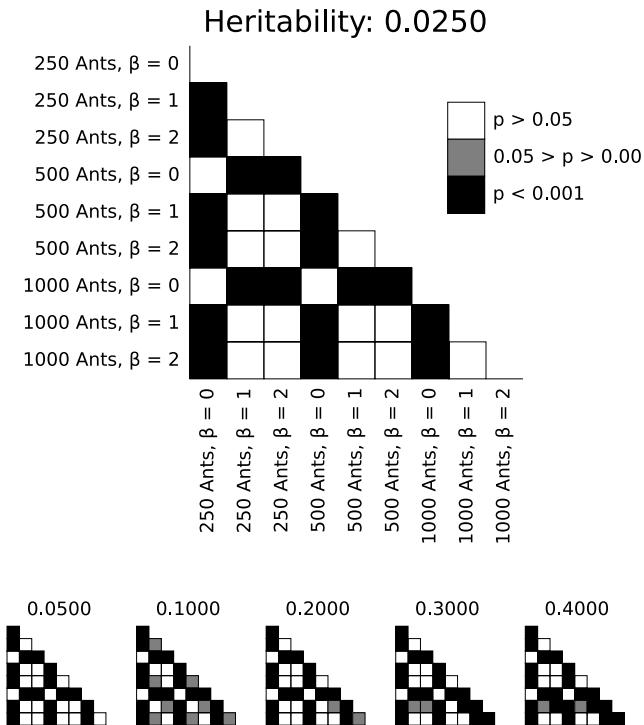


Fig. 3. Fisher's exact test p -values for assessing whether the differences among groups seen in the bar graph is significant. Values of p between 0.05 and 0.001 mean that between one time out of twenty and one time out of one-thousand, a difference of that magnitude is expected by chance alone. Values of p below 0.001 mean that less than one time out of one-thousand, a difference of that magnitude is expected by chance alone. Order of parameter settings is retained between the example heritability (0.0250) and the other heritabilities shown.

This quantitatively confirms that the expert knowledge factor, E is a crucial component in the success of the ACO metaheuristic in this domain. At a heritability of 0.100 where the largest performance difference occurs among different weightings of β , it is apparent that the difference between the powers for values of $\beta = 1$ and $\beta = 2$ is significant. This suggests that a higher weighting of β seems to be advantageous for this problem. In addition at higher heritabilities the difference between the 1000 ants/5 updates power with a β of 1 becomes significantly different from the power with other parameter settings, which also suggests that a high weighting of expert knowledge is more appropriate in these cases, especially when the number of ants is high and the number of updates low.

9 Discussion and Conclusion

Our results show that ACO is a viable approach to this problem when an expert knowledge is added in to the pheromone updating rule. This suggests that ACO may be an appropriate search strategy when exhaustive analysis is impossible. These results are also encouraging given the relative simplicity of this approach and the relatively high power given that only about 1% of the dataset can be explored by the metaheuristic. These results indicate a power somewhat greater than the previously used genetic programming approaches [7,9,10].

In this case the pheromone updating rule is a combination of the classifier accuracy and the expert knowledge from TuRF. Modifications such as a rank based ant system [21] or a *MIN – MAX* ant system [22,23] warrant investigation as these approaches may be better able to deal with this type of data. Also warranting more investigation is wide sweep of the β , expert knowledge weighting, parameter which leads to increased power at low (0.100) heritabilities. Merkle et al. show that dynamically altering the heuristic weighting factor, β , during the search can lead to greater success for a resource-constrained project scheduling problem [24]. Perhaps a similar approach is appropriate here to better balance exploration and exploitation.

Work now can focus on a number of areas within the ACO metaheuristic. What is the best way to initialize the pheromone? Are there more appropriate ant systems or updating rules for this problem? We have seen in the field of GP that by developing highly tuned operators it is possible to keep the power of the approach high while exploring a much smaller search space. Is it possible and advantageous to develop similar tuned approaches in the field of ant colony optimization while keeping parameters for the approach conceptually simple enough for users of the MDR software to understand?

In this work the building blocks of outside knowledge are obtained by pre-processing data with TuRF. For the realm of genetic studies, outside knowledge can also be obtained from the numerous public databases available to geneticists. Tools are being developed which integrate knowledge across these public databases and generate information about relationships between genes and disease in the context of protein interactions [25]. Future work will also focus on integrating multiple distinct expert knowledge types and sources. For the ACO

metaheuristic the question arises, is it better to include all types of outside knowledge in the same run in the same large pheromone updating rule, or is it better to use a reinitialization strategy once convergence occurs that takes advantage of different sources of expert knowledge in phases? Here we insert the heuristic information into the pheromone updating rule. We have found that given domain specific knowledge and an approach which takes advantage of this knowledge, it is possible for an ACO strategy to succeed, even for a needle-in-a-haystack problem. This indicates that ACO may be a useful wrapper for genome wide analysis of common human diseases with a complex genetic architecture.

Acknowledgements. This work was supported by NIH grants LM009012 and AI59694. The authors would like to thank Chantel Sloan, Anna Tyler and Ryan Urbanowicz for their careful reading of the manuscript.

References

1. Freitas, A.A.: Understanding the crucial role of attribute interaction in data mining. *Artif. Intell. Rev.* 16(3), 177–199 (2001)
2. Goldberg, D.E.: *The Design of Innovation: Lessons from and for Competent Genetic Algorithms*. Kluwer Academic Publishers, Norwell (2002)
3. Moore, J.H.: The ubiquitous nature of epistasis in determining susceptibility to common human diseases. *Human Heredity* 56, 73–82 (2003)
4. The International HapMap Consortium: A haplotype map of the human genome. *Nature* 437(7063), 1299–1320 (2005); 10.1038/nature04226
5. Dorigo, M., Maniezzo, V., Colorni, A.: Positive feedback as a search strategy. Technical report 91-016, Dipartimento di Elettronica e Informatica, Politecnico di Milano (1991)
6. Parpinelli, R., Lopes, H., Freitas, A.: An Ant Colony Based System for Data Mining: Applications to Medical Data. In: Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001), pp. 791–797 (2001)
7. Moore, J.H., White, B.C.: Genome-wide genetic analysis using genetic programming: The critical need for expert knowledge. *Genetic Programming Theory and Practice IV* (2007)
8. White, B.C., Gilbert, J.C., Reif, D.M., Moore, J.H.: A statistical comparison of grammatical evolution strategies in the domain of human genetics. In: Proceedings of the IEEE Congress on Evolutionary Computing, pp. 676–682 (2005)
9. Moore, J.H., White, B.C.: Exploiting expert knowledge in genetic programming for genome-wide genetic analysis. In: Runarsson, T.P., Beyer, H.-G., Burke, E.K., Merelo-Guervós, J.J., Whitley, L.D., Yao, X. (eds.) PPSN 2006. LNCS, vol. 4193, pp. 969–977. Springer, Heidelberg (2006)
10. Greene, C.S., White, B.C., Moore, J.H.: An expert knowledge-guided mutation operator for genome-wide genetic analysis using genetic programming. In: Rajapakse, J.C., Schmidt, B., Volkert, L.G. (eds.) PRIB 2007. LNCS (LNBI), vol. 4774, pp. 30–40. Springer, Heidelberg (2007)
11. Moore, J.H., White, B.C.: Tuning relieff for genome-wide genetic analysis. In: Marchiori, E., Moore, J.H., Rajapakse, J.C. (eds.) EvoBIO 2007. LNCS, vol. 4447, pp. 166–175. Springer, Heidelberg (2007)

12. Moore, J.H.: Computational analysis of gene-gene interactions using multifactor dimensionality reduction. *Expert Review of Molecular Diagnostics* 4(6), 795–803 (2004)
13. Moore, J.H.: Genome-wide analysis of epistasis using multifactor dimensionality reduction: feature selection and construction in the domain of human genetics. In: *Knowledge Discovery and Data Mining: Challenges and Realities with Real World Data*. IGI (2007)
14. Moore, J.H., Gilbert, J.C., Tsai, C.T., Chiang, F.T., Holden, T., Barney, N., White, B.C.: A flexible computational framework for detecting, characterizing, and interpreting statistical patterns of epistasis in genetic studies of human disease susceptibility. *Journal of Theoretical Biology* 241(2), 252–261 (2006)
15. Ritchie, M.D., Hahn, L.W., Roodi, N., Bailey, L.R., Dupont, W.D., Parl, F.F., Moore, J.H.: Multifactor dimensionality reduction reveals high-order interactions among estrogen metabolism genes in sporadic breast cancer. *American Journal of Human Genetics* 69, 138–147 (2001)
16. Wilke, R.A., Reif, D.M., Moore, J.H.: Combinatorial pharmacogenetics. *Nature Reviews Drug Discovery* 4, 911–918 (2005)
17. Kira, K., Rendell, L.A.: A practical approach to feature selection. In: *Machine Learning: Proceedings of the AAAI 1992* (1992)
18. Kononenko, I.: Estimating attributes: Analysis and extension of relief. In: *Machine Learning: ECML-1994*, vol. 94, pp. 171–182 (1994)
19. Robnik-Sikonja, M., Kononenko, I.: Theoretical and empirical analysis of relieff and reliefff. *Mach. Learn.* 53, 23–69 (2003)
20. Sokal, R.R., Rohlf, F.J.: *Biometry: the principles and practice of statistics in biological research*, 3rd edn. W. H. Freeman and Co., New York (1995)
21. Bullnheimer, B., Hartl, R., Strauss, C.: A new rank-based version of the ant system: a computational study. *Central European Journal for Operations Research and Economics* 7(1), 25–38 (1999)
22. Stützle, T., Hoos, H.: MAX-MIN Ant System and local search for the traveling salesman problem. *IEEE International Conference on Evolutionary Computation* 1997, 309–314 (1997)
23. Stützle, T., Hoos, H.H.: MAX-MIN Ant System. *Future Generation Computer Systems* 16(8), 889–914 (2000)
24. Merkle, D., Middendorf, M., Schmeck, H.: Ant colony optimization for resource-constrained project scheduling. *IEEE Transactions on Evolutionary Computation* 6(4), 333–346 (2002)
25. Gonzalez, G., Uribe, J.C., Tari, L., Brophy, C., Baral, C.: Mining gene-disease relationships from biomedical literature: Weighting protein-protein interactions and connectivity measures. In: *Pacific Symposium on Biocomputing*, vol. 12, pp. 28–39 (2007)

cAnt-Miner: An Ant Colony Classification Algorithm to Cope with Continuous Attributes

Fernando E.B. Otero, Alex A. Freitas, and Colin G. Johnson

Computing Laboratory, University of Kent, Canterbury, UK
`{febo2,A.A.Freitas,C.G.Johnson}@kent.ac.uk`

Abstract. This paper presents an extension to Ant-Miner, named *cAnt-Miner* (Ant-Miner coping with continuous attributes), which incorporates an entropy-based discretization method in order to cope with continuous attributes during the rule construction process. By having the ability to create discrete intervals for continuous attributes “on-the-fly”, *cAnt-Miner* does not require a discretization method in a preprocessing step, as Ant-Miner requires. *cAnt-Miner* has been compared against Ant-Miner in eight public domain datasets with respect to predictive accuracy and simplicity of the discovered rules. Empirical results show that creating discrete intervals during the rule construction process facilitates the discovery of more accurate and significantly simpler classification rules.

1 Introduction

Data mining is a multi-disciplinary field which aims to extract knowledge from databases [1]. The data mining task addressed in this paper is the classification task, where the goal is to predict the class of an example, given the values of a set of attributes for that example. In essence, the classification task consists of inducing a model from the data by observing relationships between predictor attributes and classes, which can be used later to classify new examples. The discovered knowledge is often represented in the form of *IF (conditions) THEN (class)* classification rules, which has the advantage of representing a comprehensible model to the user [2].

In the context of discovering classification rules in data mining, Ant Colony Optimization (ACO) [3] algorithms have been successfully applied to different classification problems [4]. Ant-Miner [5,6], the first implementation of an ACO algorithm for the classification task of data mining, has been shown to be competitive with the well-known C4.5 [7] and CN2 [8] classification-rule discovery algorithms.

Although real-world classification problems are often described by nominal (with a finite number of nominal or discrete values) and continuous (real-valued) attributes, Ant-Miner has the limitation of being able to cope only with nominal attributes in its rule construction process. In order to overcome this limitation, a commonly used approach is to discretize continuous attributes in a preprocessing step. In essence, a discretization method aims at converting continuous attributes into nominal (discrete) attributes by creating interval boundaries (e.g.

a continuous attribute *age* might be discretized into “0–14”, “15–24”, “25–64” and “65+” intervals). A potential disadvantage of this approach is that less information will be available to the classifier – since the discrete intervals have a coarser granularity – which can have a negative impact on the accuracy of the discovered knowledge.

This paper proposes an extension to Ant-Miner, named cAnt-Miner (Ant-Miner coping with continuous attributes), which incorporates an entropy-based discretization method in order to cope with continuous attributes during the rule construction process. cAnt-Miner has the ability to create discrete intervals for continuous attributes “on-the-fly”, taking advantage of all continuous attributes information, rather than requiring that a discretization method be used in a preprocessing step. Note that, although many Ant-Miner variations have been proposed – as reviewed in [4] – none of them can discretize attributes “on-the-fly” (during the rule construction process) as proposed in this paper.

The remainder of this paper is organized as follows. Section 2 presents an overview of Ant-Miner. Section 3 discusses some Ant-Miner variations proposed in the literature. In Section 4, the proposed cAnt-Miner algorithm is introduced. Section 5 presents the computational results evaluating cAnt-Miner. Finally, Section 6 presents the conclusion of the paper and future research directions.

2 Ant-Miner Overview

The goal of Ant-Miner is to extract *IF-THEN* classification rules of the form *IF* (*term*₁) *AND* (*term*₂) *AND* ... *AND* (*term*_{*n*}) *THEN* (*class*) from data. Each term in the rule is a triple (*attribute*, *operator*, *value*), where *operator* represents a relational operator and *value* represents a value of the domain of *attribute* (e.g. *Sex = male*). The *IF* part corresponds to the rule’s antecedent and the *THEN* part is the rule’s consequent, which represents the class to be predicted by the rule. An example that satisfies the rule’s antecedent will be assigned the class predicted by the rule. As the original Ant-Miner only works with nominal (categorical or discrete) attributes, the only valid relational operator is “=” (equality operator). Continuous attributes need to be discretized in a preprocessing step.

Algorithm 1 presents a high level pseudo-code of Ant-Miner [6]. In essence, Ant-Miner works as follows. It starts with an empty rule list and iteratively adds one rule at a time to that list while the number of uncovered training examples is greater than a user-specified maximum value (*while* loop). In order to construct a rule, a single ant starts with an empty rule (no terms in its antecedent) and adds one term at a time to the rule antecedent (*repeat-until* loop). It probabilistically chooses a term to be added to the current partial rule based on the values of the amount of pheromone (τ) and a problem-dependent heuristic information (η) associated with the term. A pheromone value and a heuristic value are associated with each possible term – i.e. each possible triple (*attribute*, *operator*, *value*). As usual in ACO, heuristic values are fixed (based on an information theoretical measure of the predictive power of the term), while pheromone values are iteratively updated based on the quality of the rules built

Algorithm 1. High level pseudo-code of Ant-Miner

```

begin Ant-Miner
    training_set  $\leftarrow$  all training examples;
    rule_list  $\leftarrow \emptyset$ ;
    while  $|training\_set| > max\_uncovered\_training\_examples$  do
         $\tau \leftarrow initialize\_\text{pheromones}$ ;
         $rule_{best} \leftarrow \emptyset$ ;
         $i \leftarrow 1$ ;
        repeat
             $rule_i \leftarrow CreateRule()$ ;
            ComputeConsequent( $rule_i$ );
            Prune( $rule_i$ );
            UpdatePheromones( $\tau, rule_i$ );
            if  $Q(rule_i) > Q(rule_{best})$  then
                 $rule_{best} \leftarrow rule_i$ ;
            end
             $i \leftarrow i + 1$ ;
        until  $i \geq max\_number\_rules$  OR convergence ;
        rule_list  $\leftarrow rule\_list \cup rule_{best}$ ;
        training_set  $\leftarrow training\_set \setminus CorrectlyCoveredExamples(rule_{best})$ ;
    end
end

```

by the ants. The ant keeps adding a term to the partial rule until any term added to the antecedent would make the rule cover less training examples than a user-specified threshold, which would make the rule too specific and unreliable, or all attributes have already been used by the ant. The latter rule construction stopping criterion is necessary because an attribute can only occur once in the antecedent of a rule, in order to avoid inconsistencies such as (*Sex = male*) AND (*Sex = female*). Once this process of rule construction has finished, first the rule constructed by the ant is pruned to remove irrelevant terms from the rule antecedent. Then, the consequent of the rule is chosen to be the class value most frequent among the set of training examples covered by the rule. Finally, pheromone trails are updated and another ant starts to construct a new rule. The process of constructing a rule is repeated until a user-specified number of rules has been reached, or the current ant has constructed a rule that is exactly the same as rules constructed by a predefined number of previous ants, which works as a rule convergence test. The best rule, based on a quality measure Q , found along this iterative process is added to the rule list and the correctly classified training examples are removed from the training set. An example is considered correctly classified if it satisfies the rule antecedent and has the class predicted by the rule consequent.

In [5,6], Ant-Miner was compared against the well-known C4.5 [7] and CN2 [8] rule induction algorithms. In terms of predictive accuracy, the results have shown that Ant-Miner is competitive with both C4.5 and CN2. The biggest difference found was related to the complexity of the discovered rules. Ant-Miner was able

to find significant simpler rules, both in terms of a smaller number of rules and a smaller number of terms (conditions) per rule, than C4.5 and CN2.

3 Related Work on Ant-Miner Variations

Following the introduction of Ant-Miner, several variations were proposed [4]. They involve different pruning and pheromone update procedures, new rule quality measures and heuristic functions, discovering fuzzy classification rules and discovering rules for multi-label classification problems.

Chan & Freitas [9] have proposed a new rule pruning procedure for Ant-Miner. They have observed that the original Ant-Miner's pruning procedure processing time increases significantly with a large increase in the number of attributes, which affects the scalability of the method. To overcome this limitation, it was proposed a new prune procedure that led to the discovery of simpler (shorter) rules and improved the computational time in datasets with a large number of attributes.

Martens et al. [10] have introduced a new classification algorithm, named AntMiner+, based on Ant-Miner. It differs from the original Ant-Miner implementation in several aspects. Firstly, it makes a distinction between nominal and ordinal attributes. Nominal attributes have unordered nominal values (e.g. gender has unordered values "male" and "female"). Ordinal attributes are those categorical or discrete attributes whose values are ordered (e.g. "0", "1", "2", "3" and "4 or more", which may be the domain of an attribute that represents the number of children in a family). Instead of creating a pair (*attribute* = *value*) for each value of an ordinal attribute, AntMiner+ creates two types of bounds that represent the interval of values to be chosen by the ants. The first type represents the lower bound of the interval and takes (*attribute* \geq *value_i*) form, and the second type represents the upper bound of the interval and takes (*attribute* \leq *value_j*) form (*value_i* and *value_j* are values from the attribute domain). Moreover, it employs different pheromone initialization and update procedures based on the $\mathcal{MAX} - \mathcal{MIN}$ ant system (\mathcal{MMAS}) [11]. For additional details refer to [10].

Galea & Chen [12] presented an ACO approach for the induction of fuzzy rules, named FRANTIC-SRL (Fuzzy Rules from ANT-Inspired Computation - Simultaneous Rule Learning). FRANTIC-SRL runs several ACO algorithm instances in parallel, where each instance generates rules for a particular class. By having separate ACO instances, separate pheromone matrices are maintained for each class.

Swaminathan [13] proposed an extension to Ant-Miner which enables interval conditions in the rules. While it still uses a discretization method to define discrete intervals for continuous attributes in a preprocessing step, the continuous values are not replaced in the dataset. For each discrete interval, a node (e.g. *humidity* \leq 75) is added to the construction graph and the pheromone value associated to the node is calculated using a mixed kernel probability density function (PDF).

Chan & Freitas [14] proposed a new ACO algorithm, named MuLAM (Multi-Label Ant-Miner) for the multi-label classification task. In a nutshell, MuLAM differs from the original Ant-Miner in three aspects. First, a classification rule can predict one or more class attributes, as in multi-label classification problems an example can belong to more than one class. Second, at each iteration, each ant constructs a set of rules instead of a single rule as in the original Ant-Miner. Third, it uses a pheromone matrix for each class attribute and pheromone updates only occur on the matrix of class attributes that occur in the rule's consequent.

Despite the Ant-Miner variations proposed in the literature, to the best of our knowledge, extending Ant-Miner to discretize continuous attributes “on-the-fly” (during the rule construction process) is a research topic that has not yet been explored. We believe that extending Ant-Miner to cope with continuous attributes “on-the-fly” would enhance its predictive accuracy given that the use of a discretization method in a preprocessing step can lead to loss of predictive power – since less information is available to the classification algorithm.

4 Handling Continuous Attributes in Ant-Miner

There are numerous discretization methods for handling continuous attributes available in the literature [15,16]. These methods can be grouped according to different discretization strategies. Methods that make use of the examples' class information are referred to as *supervised*, while *unsupervised* methods do not use the class information (*supervised* vs. *unsupervised*). *Global* methods use the entire example space to define discrete intervals while *local* methods use a subset of example space (*global* vs. *local*). One can also categorize discretization methods as *static*, if they are applied in a data preprocessing phase before the classification algorithm is run, or as *dynamic*, if they are applied while a classifier is being built (*static* vs. *dynamic*). For a more detailed overview of different kinds of discretization methods, see [15,16].

As mentioned in the previous section, the current version of Ant-Miner does not cope with continuous attributes directly. It requires continuous attributes to be discretized in a preprocessing step. In the experiments reported in [5,6], the discretization method C4.5-Disc [17] was applied prior to Ant-Miner in a data preprocessing phase. In essence, the C4.5-Disc discretization method consists in using the well-known C4.5 [7] decision tree induction algorithm to create discrete intervals for each continuous attribute separately. For each continuous attribute, C4.5 is applied to a reduced dataset which only contains the attribute to be discretized and the class attribute. After the decision tree which contains binary splits referring only to the single attribute being discretized is built, each path of the tree from a leaf node to the root node corresponds to a discrete interval. For further details, refer to [17]. The C4.5-Disc discretization method would be categorized as *supervised*, *global* and *static* based on the criteria described above.

In this paper, we propose a *dynamic* discretization method which is incorporated into Ant-Miner's rule construction process and consequently avoids the

need for running a discretization method in a preprocessing step. First of all, we have extended the original Ant-Miner to support continuous attributes in the rule antecedent taking the form of $(attribute_c < value)$ or $(attribute_c \geq value)$, where $value$ is a value belonging to the domain of the continuous attribute $attribute_c$. Furthermore, we incorporated an entropy-based discretization method into Ant-Miner's rule construction process to dynamically create thresholds on continuous attributes domain values. The entropy measure, which is derived from information theory and often used in data mining, quantifies the impurity of a collection of examples and it is the same measure used as heuristic function in Ant-Miner. Details of the proposed Ant-Miner extension, named cAnt-Miner, are provided in the next sub-sections.

4.1 Construction Graph

The original Ant-Miner's construction graph consists of a fully connected graph in which for each nominal attribute a_i and value v_{ij} (where a_i is the i -th attribute and v_{ij} is the j -th value belonging to the domain of a_i), a node $(a_i = v_{ij})$ is added to the graph representing the $term_{ij}$ used to create a classification rule.

We have extended the construction graph to cope with continuous attributes as follows. For each continuous attribute a_i , we add a node (a_i) to the graph representing the $term_i$. Then, the node (a_i) is connected to all previous nodes of the construction graph. It should be noted that at this point the continuous values have not been discretized. The discretization occurs when an ant selects a node that represents a continuous attribute to be added to its current partial rule, as described in sub-section 4.3.

4.2 Heuristic Problem-Dependent Information

The heuristic value associated with each $term_{ij}$ in Ant-Miner involves a measure of entropy. In the case of nominal attributes, where every $term_{ij}$ has the form $(a_i = v_{ij})$, the entropy for the attribute-value pair is computed as in equation (1) – used in the original Ant-Miner:

$$entropy(a_i = v_{ij}) \equiv \sum_{c=1}^k -p(c | a_i = v_{ij}) \cdot \log_2 p(c | a_i = v_{ij}) \quad (1)$$

where $p(c | a_i = v_{ij})$ is the empirical probability of observing class c conditional on having observed $a_i = v_{ij}$ and k is the number of classes. Note that the entropy is a measure of the impurity in a collection of examples, hence higher entropy values correspond to more uniformly distributed classes and smaller predictive power for the term in question.

However, the equation (1) cannot be straightforwardly applied to compute the entropy of nodes representing continuous attributes ($term_i$) since these nodes

do not represent an attribute-value pair. In order to compute the entropy of $term_i$, we need to select a threshold value v to dynamically partition the continuous attribute a_i into two intervals: $a_i < v$ and $a_i \geq v$. The best threshold value is the value v that minimizes the entropy of the partition, given by:

$$ep_v(a_i) \equiv \frac{|S_{a_i < v}|}{|S|} \cdot entropy(a_i < v) + \frac{|S_{a_i \geq v}|}{|S|} \cdot entropy(a_i \geq v) \quad (2)$$

where $|S_{a_i < v}|$ is the total number of examples in the partition $a_i < v$ (partition of training examples where the attribute a_i has a value less than v), $|S_{a_i \geq v}|$ is the total number of examples in the partition $a_i \geq v$ (partition of training examples where the attribute a_i has a value greater or equal to v) and $|S|$ is the total number of training examples. After the selection of the threshold v_{best} , the entropy of the $term_i$ corresponds to the minimum entropy value of the two partitions and it is defined as:

$$entropy(term_i) \equiv min (entropy(a_i < v_{best}), entropy(a_i \geq v_{best})) \quad (3)$$

We select the lowest entropy value since it corresponds to the value associated with the “purest” partition (the partition with more examples belonging to the same class) and it represents the expected predictive power (quality) of the $term_i$ (when $term_i$ is added to the rule). It should be noted that the entropy of every $term_i$ – i.e. every term having a continuous attribute – is always the same as the entropy value of every $term_{ij}$ – every term representing an attribute-value pair of a nominal attribute. Therefore, the entropy of all $term_i$ and $term_{ij}$ are computed as a preprocessing step to save computational time.

Concerning the computational complexity, the process of finding a threshold value can be divided into two steps. First, the continuous attribute values have to be sorted in order to facilitate the computation of the number of examples belonging to each candidate interval. The time complexity of this step is $O(n \log n)$, where n is the number of examples under consideration. Second, the evaluation of candidate threshold values has the complexity of $O(n)$, where in this case n represents the number of candidate values to be evaluated. It should be noted that not all candidate threshold values are evaluated, only those that form boundaries between classes, as proposed by [18]. Therefore, the efficiency of the evaluation is increased since less candidate threshold values need to be checked.

4.3 Rule Construction

As every term in the antecedent of a rule must be a triple (*attribute*, *operator*, *value*), when an ant chooses a node that represents a continuous attribute a_i to add to its current partial rule, a relational operator and a threshold value are selected as follows. First, the best threshold value for attribute a_i is selected as described in sub-section 4.2, subject to one restriction: only the examples covered by the current partial rule are considered in the evaluation of threshold values. Therefore, the selection of a threshold value is influenced by the terms occurring in the current partial rule. This is what makes the proposed discretization method a *dynamic* one, so that the choice of a threshold value is tailored

to the current candidate rule. The only exception to this restriction is when the current partial rule is empty. In those cases, all training examples are used in the evaluation of threshold values, as given by equation (2).

Then, after selecting the threshold value v_{best} , a relational operator op is selected based on the entropy values of the two partitions generated. If the partition $a_i < v_{best}$ has a lower entropy, then the operator “ $<$ ” (less-than operator) is selected. If the partition $a_i \geq v_{best}$ has a lower entropy, then the operator “ \geq ” (greater-equal operator) is selected. The operator selection has a bias of selecting the more “pure” partition, given that lower entropy values are favored over higher entropy values.

Once the threshold value v_{best} and the operator op are selected, a term in the form of a triple (a_i, op, v_{best}) is added to the ant’s current partial rule (e.g $age \geq 25$) and the rule continues to undergo the Ant-Miner’s rule construction process.

4.4 Pheromone Updating

In the original Ant-Miner, every $term_{ij}$ has an associated pheromone value which undergoes the pheromone updating (increasing and decreasing) process. In summary, the pheromone updating process works as follows. The pheromone associated with each $term_{ij}$ occurring in the rule created by an ant is increased in proportion to the quality of the rule in question. The pheromone associated with each $term_{ij}$ that does not occur in the rule is decreased, simulating the pheromone evaporation effect observed in real ant colonies.

We have extended the original Ant-Miner’s pheromone updating process to cope with $term_i$ (a term that represents a continuous attribute a_i) as follows. Since the pheromone value is associated with a continuous attribute a_i , and not the triple (a_i, op, v_{best}) that is added to the current partial rule (see sub-section 4.3), the operator op and the threshold value v_{best} are discarded in the updating process. In other words, there is a single entry for each continuous attribute a_i in the pheromone matrix, in contrast to multiple entries for nominal attributes, which have an entry for every (a_i, v_{ij}) pair (where a_i is the i -th nominal attribute and v_{ij} is the j -th value belonging to the domain of a_i).

In the proposed cAnt-Miner, pheromone is still used to indicate the quality of continuous attributes, but the actual choice of the threshold for each continuous attribute is dynamically customized to each rule being constructed by the ants. This effectively incorporates task-dependent knowledge into the algorithm, which tends to increase its effectiveness.

5 Computational Results and Discussion

In order to evaluate the proposed cAnt-Miner algorithm, we have selected eight datasets from the UCI Irvine machine learning repository [19] which had at least one continuous attribute. Table 1 shows a summary of the selected datasets. All experiments were conducted running a well-known 10-fold cross-validation

procedure [2]. Since the original version of Ant-Miner requires the data to be discretized in a preprocessing step, for each cross-validation fold we separately discretized (using the C4.5-Disc discretization method [17]) the training set and the created discrete intervals were used to discretize the test set. This was necessary because, if we had discretized the entire dataset before creating the cross-validation folds, the discretization method would have access to the test data, which would have compromised the reliability of the experiments. Also, we removed the duplicated instances (instances with the same values for all attributes) from the resulting discrete dataset to avoid the possibility that a test set contains an example that is the same as a training example.

We have compared the performance of *cAnt*-Miner against Ant-Miner, with respect to predictive accuracy and simplicity of the discovered rule lists. In all experiments, the user-defined parameters of *cAnt*-Miner and Ant-Miner were set to: *No_of_ants* = 3000, *min_cases_per_rule* = 5, *max_uncovered_cases* = 10 and *No_rules_converg* = 10 (detailed explanation of these parameters can be found in [6]). We have made no attempt to optimize these parameters for the datasets used in the experiments. It should be noted that for *cAnt*-Miner, the original – with nominal (or discrete) and continuous attributes – datasets were used, and for Ant-Miner, the discrete – with only nominal (or discrete) attributes – datasets were used. To make the comparison as fair as possible, the same cross-validation folds were used in both datasets, with the exception that in the discrete datasets we removed the duplicated examples.

Table 2 summarizes the results comparing the predictive accuracy of Ant-Miner and *cAnt*-Miner. Each entry in the table shows the average value of the accuracy obtained via the cross-validation procedure followed by the standard deviation. An entry in the *cAnt*-Miner column is shown in bold if, for the

Table 1. Summary of the datasets used in the experiments. The first column gives the dataset name, the second and third columns give the number of nominal and continuous attributes respectively, the forth column gives the number of classes, the fifth column gives the number of instances in the original dataset and the sixth column gives the number of instances in the discrete dataset (after the removal of duplicated examples).

Dataset	Attributes		Classes	Size	
	Nominal	Continuous		Original	Discrete
wdbc	0	30	2	569	366
crx	9	6	2	690	639
hepatitis	13	6	2	155	116
glass	0	9	7	213	119
ionosphere	0	34	2	350	292
wine	0	13	3	178	126
australian	8	6	2	690	637
heart	6	7	2	270	232

Table 2. Predictive accuracy (*mean \pm standard deviation*) of Ant-Miner and cAnt-Miner after the 10-fold cross-validation procedure. An entry in the cAnt-Miner column is shown in bold if, for the corresponding dataset, the accuracy achieved with cAnt-Miner was significantly greater than the accuracy achieved with Ant-Miner for that dataset – according to a two-tailed Student’s t-test with significance level $\alpha = 5\%$.

Dataset	Ant-Miner	cAnt-Miner
wdbc	93.27 ± 1.44	95.57 ± 0.55
crx	85.32 ± 1.26	85.56 ± 1.16
hepatitis	74.61 ± 2.80	84.89 ± 2.57
glass	51.48 ± 4.84	65.69 ± 2.59
ionosphere	90.68 ± 2.32	90.00 ± 1.49
wine	94.58 ± 2.51	95.14 ± 2.01
australian	85.52 ± 1.60	86.60 ± 1.46
heart	77.62 ± 3.27	79.27 ± 2.74

corresponding dataset, the accuracy achieved with cAnt-Miner was significantly greater than the accuracy achieved with Ant-Miner for that dataset – according to a two-tailed Student’s t-test with significance level $\alpha = 5\%$. In two datasets, namely hepatitis and glass, cAnt-Miner was significantly more accurate than Ant-Miner. Both Ant-Miner and cAnt-Miner achieved similar (with no significant difference) accuracies in the remaining six datasets.

Table 3 summarizes the results concerning the simplicity of the discovered rule lists, measured by the total number of terms (conditions) in all discovered rules. Each entry in the table shows the average rule list size obtained via cross-validation procedure followed by the standard deviation. An entry in the cAnt-Miner column is shown in bold if, for the corresponding dataset, the rule list discovered by cAnt-Miner was significantly simpler than the rule list discovered by Ant-Miner for that dataset – according to a two-tailed Student’s t-test with significance level $\alpha = 5\%$. Concerning the simplicity of the discovered rule lists, cAnt-Miner discovered significantly simpler rule lists than Ant-Miner in six out of eight datasets. In two datasets, namely crx and australian, both cAnt-Miner and Ant-Miner discovered rule lists with similar simplicity.

The results obtained in the experiments can be summarized as follows. With respect to predictive accuracy, cAnt-Miner was significantly more accurate than Ant-Miner in the hepatitis and glass dataset. In the remaining six datasets, both cAnt-Miner and Ant-Miner achieved similar accuracies. Hence, overall cAnt-Miner was the most accurate algorithm for this set of eight datasets. Regarding the simplicity of the discovered rule lists, in six out of eight datasets, cAnt-Miner discovered rules significantly simpler than Ant-Miner’s rules. These results empirically show that the proposed *dynamic*-discretization cAnt-Miner facilitates the discovery of more accurate and significantly simpler classification rules when compared to the *static*-discretization Ant-Miner.

Table 3. Simplicity of the discovered rule list (*mean \pm standard deviation*), measured as total number of terms in all rules, of Ant-Miner and *c*Ant-Miner after the 10-fold cross-validation procedure. An entry in the *c*Ant-Miner column is shown in bold if, for the corresponding dataset, the rule list discovered by *c*Ant-Miner was significantly simpler than the rule list discovered by Ant-Miner for that dataset – according to a two-tailed Student’s t-test with significance level $\alpha = 5\%$.

Dataset	Ant-Miner	<i>c</i> Ant-Miner
wdbc	54.60 ± 5.16	9.40 ± 0.70
crx	25.60 ± 1.86	22.80 ± 2.32
hepatitis	26.10 ± 2.51	16.00 ± 0.95
glass	38.70 ± 1.52	33.30 ± 1.87
ionosphere	21.90 ± 3.08	10.20 ± 1.00
wine	8.80 ± 0.79	6.30 ± 0.45
australian	18.30 ± 1.16	21.40 ± 1.45
heart	20.40 ± 1.53	16.40 ± 0.76

An important remark is that there was no significant increase in the computational time of *c*Ant-Miner by comparison with Ant-Miner’s time, since the number of examples that need to be considered when a continuous attribute is added to the rule decreases proportionally to the number of terms present in the current partial rule (see sub-sections 4.2 and 4.3).

6 Conclusion and Future Work

This paper has presented an extension to Ant-Miner, named *c*Ant-Miner, which copes with continuous attributes during the rule construction process. By having the ability to create discrete intervals for continuous attributes “on-the-fly”, *c*Ant-Miner does not require a discretization method in a preprocessing step.

*c*Ant-Miner has been compared against Ant-Miner with respect to predictive accuracy and simplicity of the discovered rule lists in eight public domain datasets. Regarding predictive accuracy, *c*Ant-Miner significantly outperformed Ant-Miner in two datasets. Regarding simplicity of the discovered rule lists, *c*Ant-Miner found significantly simpler rule lists than Ant-Miner in six out of eight datasets. Therefore, the results obtained by *c*Ant-Miner are promising.

As future research direction, it would be interesting to extend the entropy-based discretization method used in the rule construction process to allow the creation of intervals with both lower and upper bound values in the form $v_{lower} \leq attribute \leq v_{upper}$.

References

1. Fayyad, U., Piatetsky-Shapiro, G., Smith, P.: From data mining to knowledge discovery: an overview. In: Fayyad, U., Piatetsky-Shapiro, G., Smith, P., Uthurusamy, R. (eds.) *Advances in Knowledge Discovery & Data Mining*, pp. 1–34. MIT Press, Cambridge (1996)
2. Witten, H., Frank, E.: *Data Mining: Practical Machine Learning Tools and Techniques*, 2nd edn. Morgan Kaufmann, San Francisco (2005)
3. Dorigo, M., Stützle, T.: *Ant Colony Optimization*. MIT Press, Cambridge (2004)
4. Freitas, A., Parpinelli, R., Lopes, H.: Ant colony algorithms for data mining. In: *Encyclopedia of Info. Sci. & Tech.*, 2nd edn. (to appear, 2008)
5. Parpinelli, R., Lopes, H., Freitas, A.: An ant colony algorithm for classification rule discovery. In: Abbass, H., Sarker, R., Newton, C. (eds.) *Data Mining: a Heuristic Approach*, pp. 191–208. Idea Group Publishing (2002)
6. Parpinelli, R., Lopes, H., Freitas, A.: Data mining with an ant colony optimization algorithm. *IEEE Transactions on Evolutionary Computation* 6(4), 321–332 (2002)
7. Quinlan, J.: *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Francisco (1993)
8. Clark, P., Niblett, T.: The CN2 rule induction algorithm. *Machine Learning* 3(4), 261–283 (1989)
9. Chan, A., Freitas, A.: A new classification-rule pruning procedure for an ant colony algorithm. In: Talbi, E.-G., Liardet, P., Collet, P., Lutton, E., Schoenauer, M. (eds.) *EA 2005. LNCS*, vol. 3871, pp. 25–36. Springer, Heidelberg (2006)
10. Martens, D., Backer, M.D., Haesen, R., Vanthienen, J., Snoeck, M., Baesens, B.: Classification with ant colony optimization. *IEEE Transactions on Evolutionary Computation* 11(5), 651–665 (2007)
11. Stützle, T., Hoos, H.: MAX-MIN ant system. *Future Generation Computer Systems* 16(8), 889–914 (2000)
12. Galea, M., Shen, Q.: Simultaneous ant colony optimization algorithms for learning linguistic fuzzy rules. In: Graham, A., Grosan, C., Ramos, V. (eds.) *Swarm Intelligence in Data Mining*, pp. 75–99. Springer, Heidelberg (2006)
13. Swaminathan, S.: Rule induction using ant colony optimization for mixed variable attributes. Master's thesis, Texas Tech University (2006)
14. Chan, A., Freitas, A.: A new ant colony algorithm for multi-label classification with applications in bioinformatics. In: Proc. Genetic and Evolutionary Computation Conference (GECCO-2006), pp. 27–34 (2006)
15. Dougherty, J., Kohavi, R., Sahami, M.: Supervised and unsupervised discretization of continuous features. In: *Machine Learning: Proceedings of the Twelfth Int. Conference on Artificial Intelligence*, pp. 194–202. Morgan Kauffmann, San Francisco (1995)
16. Liu, H., Hussain, F., Tan, C., Dash, M.: Discretization: An enabling technique. *Data Mining and Knowledge Discovery* 6, 393–423 (2002)
17. Kohavi, R., Sahami, M.: Error-based and entropy-based discretization of continuous features. In: *Proceedings of the 2nd International Conference Knowledge Discovery and Data Mining*, pp. 114–119. AAAI Press, Menlo Park (1996)
18. Fayyad, U., Irani, K.: Multi-interval discretization of continuous-valued attributes for classification learning. In: *Thirteenth International Joint Conference on Artificial Intelligence*, pp. 1022–1027. Morgan Kaufmann, San Francisco (1993)
19. Asuncion, A., Newman, D.: UCI machine learning repository (2007),
<http://www.ics.uci.edu/~mlearn/MLRepository.html>

Finding Minimum Spanning/Distances Trees by Using River Formation Dynamics*

Pablo Rabanal, Ismael Rodríguez, and Fernando Rubio

Dept. Sistemas Informáticos y Computación, Facultad de Informática
Universidad Complutense de Madrid, Madrid, Spain
prabanal@fdi.ucm.es, {isrodrig,fernando}@sip.ucm.es

Abstract. *River Formation Dynamics* (RFD) is an heuristic method similar to *Ant Colony Optimization* (ACO). In fact, RFD can be seen as a gradient version of ACO, based on copying how water forms rivers by eroding the ground and depositing sediments. As water transforms the environment, altitudes of places are dynamically modified, and decreasing gradients are constructed. The gradients are followed by subsequent drops to create new gradients, reinforcing the best ones. By doing so, good *solutions* are given in the form of *decreasing* altitudes. We apply this method to solve two NP-complete problems, namely the problems of finding a *minimum distances tree* and finding a *minimum spanning tree* in a variable-cost graph. We show that the gradient orientation of RFD makes it specially suitable for solving these problems, and we compare our results with those given by ACO.

1 Introduction

Ant Colony Optimization (ACO) [1,2,3] approaches provide algorithms based on copying how (natural) ants find the shortest path from the colony to the nest. In ACO a pheromone value is attached to each place, and ants probabilistically tend to choose those edges where the ratio ‘pheromone trail at destination’/‘edge cost’ is the highest. Alternatively, let us consider that this decision were based on the *gradient* of trail values instead of on the trail values themselves. In particular, let us suppose that ants probabilistically tend to choose the movement providing the highest ratio ‘difference of trails between the new place and the current place’/‘edge cost’ (for instance, the more decrease, the higher probability). Leaving aside by now the important question of how ants could iteratively create paths of *decreasing* pheromone trails (which will be addressed later), what are the differences between this gradient approach and the standard approach?

First, in the standard approach ants can be led by pheromone trails in such a way that, after some movements, it is impossible not to repeat a node, i.e. a local cycle is followed. When an ant finds that it cannot avoid to repeat a node, it is either *killed* or reinserted at the origin node. In both cases, the computational

* Research partially supported by projects TIN2006-15578-C02-01, PAC06-0008-6995, and MRTN-CT-2003-505121/TAROT.

effort required to move it was useless. However, following a cycle is impossible in the gradient approach because it would require an *ever decreasing* cycle, which is contradictory. Second, let us note that in the standard approach, when an ant finds a shorter path, it needs a lot of movements to *convince* other ants following older well-reinforced paths to join the new path. Technically speaking, reinforcing the new path until pheromone trails are higher than in older paths requires a lot of subsequent steps. On the other hand, if the difference of trails is considered then, when a shorter path is discovered, from this precise moment on its edges are preferable in the overall to the edges of older paths. This is because the difference of pheromone trails between the final destination and the origin is the same in these paths (the origin and the destination are the same indeed), but the cost is lower in the shorter path. So, the ratio '*total difference of trails*'/'*total cost*' is higher in the shorter path. On the contrary, when a shorter path is found in the standard approach, the edges of this path are not preferable yet (not even when considered as a whole) because the amount of pheromones in its edges is still negligible.

Though adopting this alternative approach provides some advantages, it immediately leads to the following question: If formed paths are not sequences of high pheromone trails but sequences of *decreasing* pheromone trails, how pheromone values must be changed after an ant moves? We can easily find an answer to this question by giving the ant metaphor up and getting some inspiration from another nature-based phenomenon: The *river formation dynamics*. Let us consider that a water mass is unleashed at some high point. Gravity will make it to follow a path down until it cannot go down anymore. In Geology terms, when it rains in a mountain, water tries to find its own way down to the sea. Along the way, water erodes the ground and transforms the landscape, which eventually creates a riverbed. When a strong down slope is traversed by the water, it extracts soil from the ground in the way. This soil is deposited later when the slope is lower. Rivers affect the environment by reducing (i.e. eroding) or increasing (i.e. depositing) the altitude of the ground. Let us note that if water is unleashed at all points of the landscape (e.g., it rains) then the river form tends to optimize the task of collecting all the water and take it to the sea, which does not imply to take the shortest path from a *given* origin point to the sea. Let us remark that there are *a lot* of origin points to consider (one for each point where a drop fell). In fact, a kind of *combined* grouped shortest path is created in this case. However, if water flows from a *single* point and no other water source is considered, then the water tends to form an efficient way to reduce the altitude (i.e., it tends to find the shortest path).

In [4] an algorithm based on these ideas called RFD (*River Formation Dynamics*) was presented. In order to apply the previous scheme, ants are substituted by *drops* and pheromone trails are replaced by *altitudes*. Drops tend to take down slopes and they modify altitudes in the process. A classical benchmark NP-complete problem, the *Traveling Salesman Problem* (TSP) [5,6], was considered, which required to adapt the general scheme to this particular problem. For instance, since the general framework implicitly avoids that drops follow cycles, a change was introduced to allow cycles involving *all* nodes. The experimental

results were compared with those obtained by ACO for the same input graphs. It was observed that the time required by RFD to find good solutions is in general longer than the time required by ACO to find equivalent solutions, though solutions provided by RFD outperform those given by ACO after some additional time passes. The reasons for these differences lie in the fact that RFD develops a *deeper* exploration of the graph (see [4] for details).

In this paper we adapt RFD to deal with two related problems. Given a cost-evaluated graph, we consider (a) finding the minimum spanning tree, and (b) finding the minimum distances tree, that is, a tree such that the addition of distances from each node to a given *exit node* is minimal. The standard forms of both problems do not require using heuristic methods because they can be polynomially solved (e.g., by using Kruskal and Dijkstra algorithms, respectively). However, some generalizations of both problems are NP-complete indeed. In this paper we introduce the following generalization: Let us consider that the cost of taking an edge e depends on the path followed so far. That is, if we traverse e after following a path σ then the cost of adding e to the path is $c_{e,\sigma}$; in general, we have $c_{e,\sigma} \neq c_{e,\sigma'}$ for any other path σ' . As far as we known, this variant of both problems has not been studied in the literature. Thus, in order to properly consider both problems under this assumption, a proof of their NP-completeness has been done (in [7] we polynomially reduce 3-SAT to each of them). Their applicability to computational problems will be discussed in subsequent sections.

Since both proposed problems consist in finding short paths, the characteristics of RFD commented before (that is, the avoidance of local cycles and the fast reinforcement of shorter paths) make it a suitable choice. Moreover, the Geology metaphor provides another characteristic that is important in this regard. Note that the *erosion* process provides a method to *punish* inefficient paths as well as to avoid blocked paths: If a path leads to a node that is lower than any adjacent node (i.e., it is a blind alley) then the drop will deposit its sediment, which will increase the altitude of the node. Eventually, this node will match the altitude of its neighbors, which will avoid that other drops fall in this node. If the ground reaches this level, other drops will be allowed to cross this node from one adjacent node to another. Thus, paths will not be interrupted at this point.

We apply both RFD and ACO to solve the presented problems. In both problems, we observe that solutions given by ACO in the short term are better than those given by RFD. However, after some additional time passes, the quality of the solutions given by RFD surpasses the quality of the solutions given by ACO. In fact, the general tendency of RFD to enable a deeper exploration of the analyzed graph remains in these cases. Let us remark that only some improvements, out a big set of choices, have already been applied to the basic RFD scheme. Thus, we think that these results are not only interesting but also promising.

The rest of the paper is structured as follows. Next we describe the behavior of RFD. In Section 3 we formally define the problems we have considered in this paper to analyze the performance of RFD. Next, in Section 4 we apply RFD and ACO to solve both problems and we report some results. Finally, in Section 5 we present our conclusions and some lines of future work.

2 River Formation Dynamics Method

In this section we introduce the basic structure of our method based on river formation dynamics. The method works as follows. Instead of associating pheromone values to edges, we associate *altitude* values to nodes. *Drops* erode the ground (they reduce the altitude of nodes) or deposit the sediment (increase it) as they move. The probability of the drop to take a given edge instead of others is proportional to the gradient of the down slope in the edge, which in turn depends on the difference of altitudes between both nodes and the distance (i.e. the *cost* of the edge). At the beginning, a flat environment is provided, that is, all nodes have the same altitude. The exception is the destination node, which is a *hole*. Drops are unleashed at the origin node, which spread around the flat environment until some of them fall in the destination node. This erodes adjacent nodes, which creates new down slopes, and in this way the erosion process is propagated. New drops are inserted in the origin node to transform paths and reinforce the erosion of promising paths. After some steps, good paths from the origin to the destination are found. These paths are given in the form of sequences of decreasing edges from the origin to the destination.

Let us consider the applicability of RFD to the problems proposed in this paper, previously sketched in the introduction (they are formally defined in the next section). Both problems consist in finding a kind of *combination* of short paths, in particular a tree. After executing RFD for some time, for each node we take the edge with the highest gradient, and we discard the rest of edges. Due to the avoidance of local cycles, the resulting graph must be a tree. As discussed before, natural rivers do not tend to form solutions where each drop goes to the sea through its shortest path, but they tend to form grouped solutions. This allows RFD to implicitly deal with *path conflicts*, i.e. situations where, at a given node, two drops coming from different origins have different preferences regarding which edge should be taken next (because costs are different for each of them; recall that we are considering that costs depend on previously followed paths). In these situations, the tendency of RFD to form grouped solutions implicitly leads to forming paths with a suitable cost tradeoff between available choices: After some steps, the erosion will reinforce more strongly the slopes providing the lowest overall cost. In addition, in the *minimum spanning tree* problem, the tendency of drops to join each other is very appropriate: If drops tend to join the main *flow*, instead of following their respective individual shortest paths, then less edges are added to the tree and the tree cost is reduced.

The tendency of ACO methods to form grouped solutions is well known, so similar arguments can be given in the case of RFD. In particular, ACO allows to form short paths from a single node to a single destination. However, combining some short paths departing from different points in such a way that a *tree* is formed is not a natural task for ACO. Let us suppose that two paths coming from different origins join at a given node and then continue together.¹ Ants coming from a departure node can be confused by pheromone trails and go on

¹ That is, the converge area reminds the form of a ‘Y’ letter.

to the *other* departure node, instead of following to the destination node. Solving this problem requires to use some artificial methods (e.g., using different types of pheromones, using *directed* pheromones, associating ants to specific areas, etc). On the contrary, edge gradients formed by RFD are *intrinsically* directed, and their direction naturally leads to the destination node. This eases the task of constructing trees in RFD. Interestingly, we can adapt RFD to the *minimum distances tree* problem just by changing a parameter: If we reduce the erosion caused by *high flows*, then the incentive of drops to join each other is partially reduced, and thus each drop tends to follow its own shortest path. For instance, we can achieve this effect by changing the erosion rules in such a way that, if n drops traverse an edge, then they make the effect of e.g. a single drop. In this case, grouped paths are promoted by the method only when they are required to solve path conflicts. Moreover, by considering *intermediate* erosion effects, we can construct trees partially fitting into the objectives of both problems (i.e., a combination of minimum spanning tree and minimum distances tree). This may be a suitable choice for several optimization problems.²

2.1 Basic Algorithm

The basic scheme of the RFD algorithm follows:

```

initializeDrops()
initializeNodes()
while (not allDropsFollowTheSamePath()) and (not otherEndingCondition())
    moveDrops()
    erodePaths()
    depositSediments()
    analyzePaths()
end while

```

The scheme shows the main ideas of the proposed algorithm. First, drops are initialized (**initializeDrops()**), i.e., all drops are put in the initial node(s). Next, all nodes of the graph are initialized (**initializeNodes()**). This consists of two operations. On the one hand, the altitude of the destination node is fixed to 0. In terms of the river formation dynamics analogy, this node represents the *sea*, that is, the final goal of all drops. On the other hand, the altitude of the remaining nodes is set to some equal value.

The **while** loop of the algorithm is executed until either all drops find the same solution (**allDropsFollowTheSamePath()**), that is, all drops departing from the same initial nodes traverse the same sequences of nodes, or another alternative finishing condition is satisfied (**otherEndingCondition()**). This condition may be used, for example, for limiting the number of iterations or the execution time. Another choice is to finish the loop if the best solution found so far is not surpassed during the last n iterations.

² For instance, we design a subway network to carry citizens from different areas to the downtown in such a way that (a) the time spent by citizens to arrive to the downtown is minimized (i.e., we need a minimum distances tree), and (b) the expenses required to build tunnels are minimized (i.e., we need a minimum spanning tree).

The first step of the loop body consists in moving the drops across the nodes of the graph (`moveDrops()`) in a partially random way. The following *transition rule* defines the probability that a drop k at a node i chooses the node j to move next:

$$P_k(i, j) = \begin{cases} \frac{\text{decreasingGradient}(i, j)}{\sum_{l \in V_k(i)} \text{decreasingGradient}(i, l)} & \text{if } j \in V_k(i) \\ 0 & \text{if } j \notin V_k(i) \end{cases} \quad (1)$$

where $V_k(i)$ is the set of nodes that are *neighbors* of node i that can be visited by the drop k and $\text{decreasingGradient}(i, j)$ represents the negative gradient between nodes i and j , which is defined as follows:

$$\text{decreasingGradient}(i, j) = \frac{\text{altitude}(j) - \text{altitude}(i)}{\text{distance}(i, j)} \quad (2)$$

where $\text{altitude}(x)$ is the altitude of the node x and $\text{distance}(i, j)$ is the length of the edge connecting node i and node j . Note that, at the beginning of the algorithm, the altitude of all nodes is the same, so $\sum_{l \in V_k(i)} \text{decreasingGradient}(i, l)$ is 0. In order to give a special treatment to flat gradients, we modify this scheme as follows: The probability that a drop moves through an edge with 0 gradient is set to some (non null) value. This enables drops to spread around a flat environment, which is mandatory, in particular, at the beginning of the algorithm.

In fact, going one step further, we also introduce this improvement: We let drops climb *increasing* slopes with a low probability. This probability will be inverse proportional to the increasing gradient, and it will be reduced during the execution of the algorithm by using a similar method to the one used in *Simulated Annealing* (see [8,9]). This new feature improves the search of good paths. Let us note that allowing climbing up gradients does not invalidates the argument that local cycles are avoided in practice in our method: After following a sequence of down gradients, completing a cycle requires to climb up all the altitude lost so far, and the probability of climbing a big up gradient is negligible.

In the next phase (`erodePaths()`) paths are eroded according to the movements of drops in the previous phase. In particular, if a drop moves from node A to node B then we erode A. That is, the altitude of this node is reduced depending on the current gradient between A and B. In particular, the erosion is higher if the down slope between A and B is high. If the edge is flat or increasing then a small erosion is performed. The altitude of the final node (i.e., the *sea*) is never modified and it remains equal to 0 during all the execution.

Once the erosion process finishes, the altitude of all nodes of the graph is slightly increased (`depositSediments()`). The objective is to avoid that, after some iterations, the erosion process leads to a situation where all altitudes are close to 0, which would make gradients negligible and would ruin all formed paths. In fact, we also enable drops to *deposit* sediment in nodes. This happens when all movements available for a drop imply to climb an increasing slope and the drop fails to climb any edge (according to the probability assigned to it). In this case, the drop is blocked and it deposits the sediments it transports. This increases the altitude of the current node. The increment is proportional to the amount of cumulated sediment.

Finally, the last step (`analyzePaths()`) studies all solutions found by drops and stores the best solution found so far.

3 Formal Problem Definition

In this section we formally define the problems that will be addressed in Section 4 by means of RFD and ACO. We assume that the cost of a path of edges e_1, \dots, e_n from a given origin node o to a given destination node d depends on the evolution of a *variable* through the path. Initially, a value v_o is assigned to this variable at node o . Then, the cost added to the path due to the inclusion of edge e_1 is an amount depending on v_o . After traversing e_1 , the value of the variable is updated to a new value v_1 . Next, the cost of adding e_2 to the path depends on v_1 . After taking e_2 , the value of the variable is updated again, and the process continues so on until we obtain the whole cost of the path e_1, \dots, e_n .

We can define a variable-cost graph by attaching some information to a standard graph. Let us consider a set of *origin nodes* (in particular, this set could include *all* nodes of the graph). Then, (1) we assign an *initial value* to each origin node; (2) we assign a *cost function* to each edge. Depending on the value of the variable just before traversing the edge, taking the edge adds a different cost; and (3) we assign a *transformation function* to each edge. Given the value of the variable before traversing the edge, it returns the new value after taking it.

Let us suppose that a variable-cost graph defined in these terms is given. On the one hand, a *minimum distances tree* is a tree connecting each origin node with the destination node in such a way that the addition of distances from each origin node to the destination is minimal. Since the returned solution is a tree, paths departing from different origin nodes could share some edges (in particular, different sequences of edges could share some suffixes). Let us note that, in general, the cost of a shared edge is different for each path because the value of the variable when the edge is reached may be different for each path. On the other hand, a *minimum spanning tree* is a tree connecting all origin nodes with the destination node in such a way that the addition of costs of edges is minimal. In this case, the cost of an edge e in a tree t is computed as follows. Let us consider all the paths of t connecting an origin node with the destination node and including edge e . The cost of e in t is the *average* of the cost of e for all of these paths. Let us note that, in both problems, trees are not required to include all nodes from the original graph, but only those actually used to connect origin nodes to the destination node. In particular, if all nodes are considered origin nodes then the resulting tree must include all nodes indeed.

Definition 1. A *variable-cost graph* is a tuple $G = (N, O, d, V, A, E)$ where:

- N is the *set of nodes*,
- $O \subseteq N$ is the *set of origin nodes*,
- d is the *destination node*,
- $V = \{v_1, \dots, v_n\}$ is a finite set of *values*,
- $A : O \longrightarrow V$ is the *initial value function*, that is, a function assigning an initial value to each origin node.
- E is the *set of edges*. Each edge $e \in E$ is a tuple (n_1, n_2, C, T) where $n_1, n_2 \in N$ are the *origin* and *destination* nodes, respectively, and

- $C : V \rightarrow \mathbb{N}$ is the *cost function* of e . Given a value in V denoting the current value of the variable, it returns the cost of traversing e .
- $T : V \rightarrow V$ is the *transformation function* of e . Given the current value of the variable, it returns the new value assigned to the variable if e is traversed.

Paths are sequences of edges departing at an origin node and arriving to the destination node. Formally, a path of G is a sequence of edges $\sigma = (e_1, \dots, e_k)$ with $e_i = (n_i, n'_i, C_i, T_i) \in E$ for all $1 \leq i \leq k$ such that $n_1 \in O$, $n'_k = d$, and for all $1 \leq i \leq k - 1$ we have $n'_i = n_{i+1}$. The *cost* of σ , denoted by $c(\sigma)$, is equal to

$$C_1(A(n_1)) + C_2(T_1(A(n_1))) + C_3(T_2(T_1(A(n_1)))) + \dots + C_k(T_{k-1}(\dots(T_2(T_1(A(n_1))))\dots))$$

The term denoting the cost of traversing e_i in the previous expression, that is $C_i(T_{i-1}(\dots(T_2(T_1(A(n_1))))\dots))$, will be denoted by $c_{e_i}(\sigma)$. In a notation abuse, we will write $e \in \sigma$ if $e = e_i$ for some $1 \leq i \leq k$.

We say that $G' = (N', O, d, V, E', A)$ with $N' \subseteq N$ and $E' \subseteq E$ is a *tree* of G if for all $o \in O$ there exists a single path $\sigma = (e_1, \dots, e_k)$ of G' departing from o , that is, such that $e_1 = (o, n, C, T)$ for some n, C, T . For each $o \in O$, we denote by σ_o the unique path of G' departing from o .

The *distances cost* of G' , denoted by $dc(G')$, is equal to $\sum_{o \in O} c(\sigma_o)$. The *spanning cost* of G' , denoted by $sc(G')$, is equal to $\sum_{e' \in E'} \frac{\{c_{e'}(\sigma_o) | o \in O, e' \in \sigma_o\}}{\{c_{e'}(\sigma_o) | o \in O, e' \in \sigma_o\}}$. \square

Now we are provided with all the needed machinery to formally define the problems considered in this paper.

Definition 2. The problem of the *minimum distances tree for a variable-cost graph*, denoted by **MDV**, is stated as follows: Given a variable-cost graph G and a natural number $K \in \mathbb{N}$, is there any tree G' of G such that $dc(G') \leq K$?

The problem of the *minimum spanning tree for a variable-cost graph*, denoted by **MSV**, is stated as follows: Given a variable-cost graph G and a natural number $K \in \mathbb{N}$, is there any tree G' of G such that $sc(G') \leq K$? \square

The previous problems generalize the classical *minimum spanning tree* and the *minimum distances tree* problems to the case where the cost of traversing each edge depends on the path traversed before taking the edge. The past path is abstracted by the *value* of the variable, which particularizes the cost of each edge for each path. Let us note that, in formal terms, we do not need to consider *several* variables in the problem definition because the dependence on past paths can be denoted by a using a single variable. Though several variants of the minimum spanning tree and the minimum distances tree problems have been studied in the literature, as far as we are concerned the variant problems proposed in this paper have not been considered. Hence, their properties must be analyzed. A proof of the NP-completeness of both problems is presented in [7].

As a matter of fact, the proposed generalization of both problems increases their applicability to new interesting scenarios. In fact, we recently came across these problems because we were constructing some *testing derivation algorithms* (for an introduction to *Formal Testing Techniques*, see e.g. [10]). Typically, the

goal of a testing methodology is to interact with the analyzed system so that all system states are reached *at least once*. If previous system configurations can be *restored*, then we can explore a part of the system, then go back to a previously traversed point, and next go on through a different way. Thus, the problem of reaching all states consists in creating a tree embracing all states. Since the time required to go from state s to state s' depends on the previous activities of the system (available resources, values of variables, etc), composing the optimal tree reaching all states at least once requires taking past activities into account. We can use a variable-cost graph to denote how the execution time of each activity depends on the current values of variables. Thus, if we assume that previous configurations can be *restored* then finding a tree allowing to reach all states in minimum time is similar to solving **MSV** for this graph. In fact, there exists several related testing problems whose basic structure is the same.

Next we consider an applicability example of **MDV**. Let us consider that a local area network (LAN) is constructed on top of a given existing networking infrastructure. The transmission cost of a given connection (i.e. *edge*) depends on the kind of information being transmitted (e.g., low connections are unacceptable for a real-time video stream, but may be suitable for low priority packets). Besides, the kind of information being transmitted depends on the kind of sender machine. Thus, a variable-cost graph can be used to define communication costs in the existing infrastructure. Let us suppose that we want to design a networking tree allowing to communicate all nodes with a central dispatcher in such a way that average communication costs are minimized. Finding this tree consists in solving **MDV**. Other applicability scenarios of **MSV** and **MDV** may be considered as well (for instance, the *subway design problem* we briefly sketched before).

4 Applying RFD and ACO to MDV and MSV

In this section we describe the application of our approach to solve **MDV** and **MSV** and we report some experimental results. We compare the results found by using ACO methods and the solutions found by using our method. All the experiments were performed in an Intel Core Duo T7250 processor with 2.00 GHz.

We present the results obtained when **MSV** is solved by both methods. In the case of RFD, we have directly applied the method presented in Section 2, while in the case of ACO we have used an implementation inspired on [11]. Three randomly generated variable-cost graphs with 100, 200, and 300 nodes were considered. In these graphs, each node is connected to approximately 40% of the rest of nodes. Variables can take up to 10 possible values. Cost functions and transformation functions attached to edges are randomly generated. In particular, features such as monotonicity or injectivity are not required in these functions. Figure 1 shows the results of an experiment where the input of both algorithms was the graph with 100 nodes. The graphic shows the cost of the solution found by each algorithm for each execution time. Analogously, Figure 2 contains the results obtained using the graph with 200 nodes. In both cases, figures show the evolution of the algorithms in a *single* execution. The basic shape shown in both

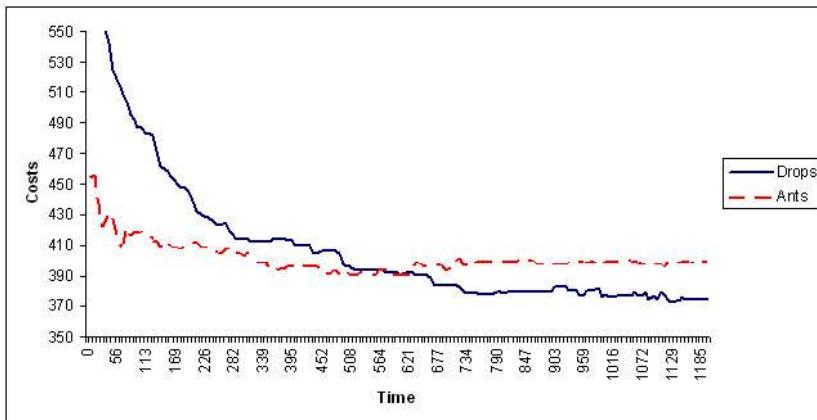


Fig. 1. Results for a randomly generated variable-cost graph with 100 nodes

figures is also obtained with the 300 nodes graph. In order to report solutions that are not biased by a single execution, each algorithm was executed ten times for each of the graphs. The following table summarizes the average and variance of the solutions found by each method.

Graph size	Average RFD	Average ACO	Variance RFD	Variance ACO
100 nodes	372.76	390.56	6.5824	4.9344
200 nodes	812.38	881.03	7.3416	9.9101
300 nodes	1221.65	1263.83	41.6905	112.7221

The results presented in the previous table show that average solutions found by the RFD method are 4%-8% better than the solutions found by ACO. Moreover,

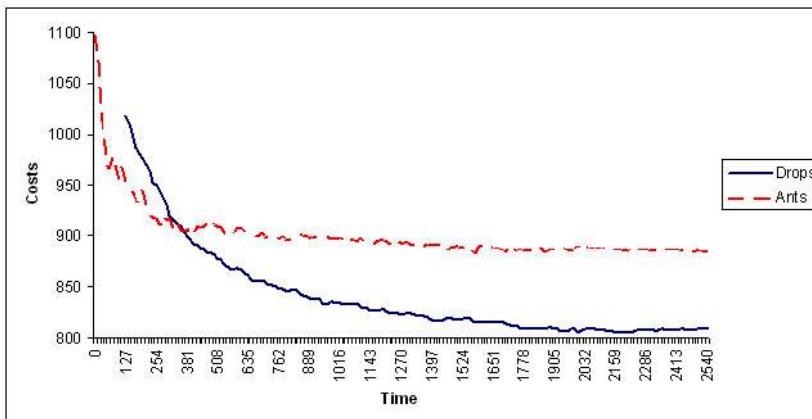


Fig. 2. Results for a randomly generated variable-cost graph with 200 nodes

the variance is also lower in our algorithm than in the ACO algorithm, with the only exception of the smallest graph.

The comparison results obtained for the MDV problem are basically the same as in the case of MSV (they are not depicted due to lack of space). That is, RFD obtains again better solutions, but solutions provided by ACO in short times are better. Times required by RFD to surpass ACO solutions are similar as well.

We extract the following conclusions from the experimental results obtained for both considered problems. We observe that ACO provides good solutions earlier than RFD. However, after some additional time passes, the solutions provided by RFD surpass the quality of those given by ACO. These features are a consequence of the fact that the exploration of the graph is *deeper* in RFD than in ACO, which in turn is due to the differences between both methods. Let us note that drops are not endowed with memory of past movements but ants are. The implicit avoidance of cycles allows us not to give drops any memory, but ants do need it because they can fall in cycles indeed. In RFD, cycles are avoided by formed gradients, but gradients are still weak during the first steps of the algorithm. Thus, drops *follow* some cycles during these early steps. In the long term, when gradients are stronger, drops avoid cycles without maintaining any memory structure, which boosts their performance with respect to ACO.

5 Conclusions and Future Work

We have applied the River Formation Dynamics approach to the problems of finding a minimum distances tree and finding a minimum spanning tree in a variable-cost graph. Let us note that RFD is conceptually related to both ACO methods and other gradient-oriented Evolutionary Computation (EC) approaches. On the one hand RFD is, in a rough sense, a gradient-driven variant of ACO. On the other hand, the gradient orientation of RFD reminds the way methods like e.g. Hill Climbing (HC) or Genetic Algorithms (GA) traverse a space of solutions by seeking for solutions with higher fitness. However, there is a big difference between RFD and these methods. RFD modifies the points of a given structure (*a graph*) by iteratively traversing and transforming these points. In this way, the structure is iteratively transformed as well, and finally the formed structure constitutes the returned solution. In HC and GA, the traversed structure is the *space of solutions* itself, which is of exponential size in general. Hence, only a small proportion of these points can be traversed. In these cases, aiming at modifying this space is unfeasible.

There are other features of RFD that make it different from other EC approaches, specially ACO. The main ones are the mechanism of focalized punishment of inefficient paths, the avoidance of local cycles, and the fast reinforcement of shorter paths. These characteristics are a consequence of the natural tendency of RFD to form paths that are intrinsically *directed* towards a given final destination. As commented in previous sections, these features are suitable for dealing with the MDV and MSV problems. Interestingly, a simple parameter change allows RFD to solve one of these problems or the other one.

Since RFD is a young method, there is still a big free room for introducing both general improvements and specific purpose variants. Out of a long list of choices, we are specially interested in simulating the *speed* of the drops. Intuitively, when a drop falls through a strong down slope, its speed is increased. This gives the drop some *kinetic energy* allowing it climb up slopes later. Thus, the speed is a kind of drop *credit*. In this way, a second derivative mechanism would be introduced in RFD.

References

1. Dorigo, M., Maniezzo, V., Colorni, A.: Ant system: optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man and Cybernetics, Part B* 26(1), 29–41 (1996)
2. Dorigo, M.: *Ant Colony Optimization*. MIT Press, Cambridge (2004)
3. Dorigo, M., Gambardella, L.: Ant colonies for the traveling salesman problem. *BioSystems* 43(2), 73–81 (1997)
4. Rabanal, P., Rodríguez, I., Rubio, F.: Using river formation dynamics to design heuristic algorithms. In: Akl, S.G., Calude, C.S., Dinneen, M.J., Rozenberg, G., Wareham, H.T. (eds.) UC 2007. LNCS, vol. 4618, pp. 163–177. Springer, Heidelberg (2007)
5. Gutin, G., Punnen, A.: *The Traveling Salesman Problem and Its Variations*. Kluwer, Dordrecht (2002)
6. Applegate, D.L., Bixby, R.E., Chvátal, V., Cook, W.J.: *The Traveling Salesman Problem: A Computational Study*. Princeton University Press, Princeton (2006)
7. Rabanal, P., Rodríguez, I., Rubio, F.: MDV and MSV NP-completeness proof (2008), <http://kimba.mat.ucm.es/~{}fernando/mdvmsv.pdf>
8. Kirkpatrick Jr., S., Gelatt, C.D., Vecchi, M.: Optimization by Simulated Annealing. *Science* 220(4598), 671 (1983)
9. Fleischer, M.: Simulated annealing: past, present, and future. In: Proceedings of the 27th conference on Winter simulation, pp. 155–161 (1995)
10. Lee, D., Yannakakis, M.: Principles and methods of testing finite state machines: A survey. *Proceedings of the IEEE* 84(8), 1090–1123 (1996)
11. Bui, T., Zrncic, C.: An ant-based algorithm for finding degree-constrained minimum spanning tree. In: GECCO, pp. 11–18. ACM Press, New York (2006)

Gathering Multiple Robotic Agents with Crude Distance Sensing Capabilities

Noam Gordon, Yotam Elor, and Alfred M. Bruckstein

Center for Intelligent Systems, CS Department
Technion — Israel Institute of Technology, Haifa, Israel
`{ngordon,yotame,freddy}@cs.technion.ac.il`

Abstract. In this follow-up to an ANTS 2004 paper we continue to investigate the problem of gathering a swarm of multiple robotic agents on the plane using very limited local sensing capabilities. In our previous work, we assumed that the agents cannot measure their distance to neighboring agents at all. In this paper, we consider a crude range-limited sensing capability that can only tell if neighboring agents are either *near* or *far*. We introduce two new variants of our previously proposed algorithm that utilize this capability. We prove the correctness of our algorithms, and show that the newly added capability can improve the performance of the algorithm significantly.

1 Introduction

The *gathering problem* is roughly defined as the problem of gathering multiple agents on the plane into a point or a small region, within finite or finite expected time. In some variants, it is sometimes also referred to as the problem of *point formation*, *convergence* or *rendezvous*. In the emerging field of theoretical swarm-robotic research, the gathering problem has been given increasing attention in recent years. This follows a general increase in interest in swarm robotics, and, in particular, what we feel as an increasing urge to attain more substantial theoretical backing to this field which has been initially mostly experimental. Being such a fundamental problem, a basis to many formation and consensus problems, it is an ideal setting for theoretical exploration of swarm robotics.

Several theoretical works on this subject exist. Current approaches include agreement on a meeting point with some unique geometrical property, assuming unlimited visibility [1,2,3,4]; using a common compass [5,6]; cyclic pursuit [7,8,9]; and others [10,11,12,13,14]. Sugihara et al. suggested a simple way to fill a convex shape, which is also useful for gathering [15].

These methods rely on strong assumptions about the agents: Some rely on labeling (e.g., pursuit), some on common orientation, and many on infinite-range visibility. Nearly all works rely on the agents' ability to measure their mutual distances. We focus on the problem under the *ant-robotic* paradigm, which assumes anonymous, homogeneous, memoryless agents lacking common knowledge and communication capabilities, and having only limited local sensing. In previous works [16,17] we proposed gathering algorithms which do not

utilize distance sensing at all. To our best knowledge, the gathering problem under the ant-robotic model and without sensing distances has not been considered elsewhere.

The initial inspiration and motivation for our work came from experiments with real robots in our lab [18], made from LEGO parts and very simple sensors, which are range-limited and do not provide usable distance measurements.

The algorithm proposed in [16] was validated in simulations which showed that the agents always converge into a small dense cluster. The key property which ensured the convergence, was that the algorithm maintains mutual visibility. However, since the agents were not aware of their mutual distances, their movement was quite conservative, in order to maintain visibility. As a result, and as evidenced by our simulations, the convergence rate was slow. A formal correctness proof of this algorithm currently remains an open problem. In [17] we proposed a randomized variant of that algorithm, which we were able to prove, yet its behavior in simulations was similar with regard to the convergence rate.

In this work we investigate whether adding a crude distance sensing capability to the agents can help them gather more efficiently. We feel that this added ability, done minimally, does not violate the ant-robotic paradigm. It is plausible that robots will be able to tell *near* from *far* at the least. That's exactly what we give them, and the answer to our question is clearly affirmative.

In Sect. 2 we provide the basic definitions and the system model. In Sect. 3 we present and prove the termination of the main proposed algorithm. In Sect. 4 we present a variant of the model and the algorithm, which adds the capability of collapsing into nearby agents. We present and discuss simulations of the algorithms in Sect. 5, and conclude in Sect. 6. Most proofs were omitted due to space constraints, and will be published in a forthcoming paper.

2 The System Model

2.1 Basic Definitions

The *world* consists of the infinite plane \mathbb{R}^2 and n point *agents* living in it. We adapt Suzuki et al.'s convenient way of modeling a system of asynchronous agents [4], sometimes referred to as the *semi-synchronous* model: *Time* is a discrete series of *time steps* $t = 0, 1, \dots$. In each time step, each agent may be either *active* or *inactive*, having no control over the random scheduling of its activity times. An active agent atomically senses its environment, performs calculations, and optionally moves instantly to another point within a distance σ (the *maximum step length*).

An agent is able to see other agents within distance V (the *visibility radius* or *range*). However, it cannot measure its exact *distance* from them. Rather, it can only tell if a visible agent is at either less or more than the *near-visibility* distance r . We assume that $3r < V$. There are no collisions. Several agents may occupy the same point. All agents are *memoryless*, *anonymous* (indistinguishable in their appearance) and *homogenous* (they lack any individuality or identity, and perform the same algorithm).

In what follows, we use the following definitions and notations:

- Denote a closed disc of radius R centered at a point p by $B_p(R)$;
- Denote the number of agents in the system by n .
- For an agent a , we also denote its position by a . The agent's position after moving (as described in the context) is denoted by a' ;
- Agents at a distance r or less are *nearby*. Otherwise, they are *far*;
- We term visibility between nearby agents *near-visibility*, and visibility between far agents *far-visibility*;
- Let b be an agent far-visible by agent a . The point on the line segment ab at a distance r from a is the *image* of b with regard to a . It is denoted by \tilde{b} .
- Denote by F the subset of the agents which have far-visible neighbors. Denote by $CH(F)$ the convex hull of the set F .
- For ease of notation we shall use F also to denote the *far-visibility graph*, whose nodes are the agents in F , and its edges are all *far-visibility edges*: $(a, b) \in F \iff r < \|a - b\| \leq V$;

2.2 Strong Asynchronicity

In Suzuki et al.'s original model, there are no assumptions regarding the activity schedule of the agents (except that no agent sleeps forever, i.e., each agent is active an infinite number of times). In our opinion, this proved to be too weak in the context of formation problems, as they could mostly achieve impossibility results in their works. In [4], they proved that there are no algorithms guaranteed to form shapes which are not purely symmetrical (e.g., perfect polygons). The reason was that the agents might not be able to break symmetries given certain schedules (e.g., if they happen to be synchronous). Perncipe et al. achieved similar impossibility results using similar reasoning under their more elaborate *totally asynchronous* model (See [19,20]), in which the agent activity cycles are neither atomic nor instantaneous. These results are very interesting in the context of computational swarm-robotic research, yet we feel that, in the research of real autonomous swarm-robotic systems, this weakness is somewhat artificial. Therefore, we revise the model by adding an assumption on the agent scheduling, striving as we can to make it as minimal and generic as possible, and reflect the natural asynchrony between autonomous robots. It is as follows:

Definition 2.1 (Strong Asynchronicity assumption). *There exists a constant $\varepsilon > 0$, such that for any subset S of the agents and in each time step t , the probability that S will be the set of active agents is at least ε .*

This assumption guarantees that any synchrony between robots will break in finite expected time, since there is always a probability of at least ε that it will break. More generally, the strong asynchronicity assumption makes us immune to adversarial schedules and allows us to prove the termination of our algorithms by construction, that is, if we show that there always exists a schedule that brings us to a goal configuration, then it is guaranteed that the algorithm will terminate in finite time. Let us formalize and generalize this idea in the following theorem.

Denote the state (or configuration) space by \mathcal{C} , and the subset of *goal configurations* by $G \subset \mathcal{C}$. The system is defined by \mathcal{C} , an initial configuration $c_0 \in \mathcal{C}$, and a Markovian transition function $\tau : \mathcal{C} \times \mathcal{C} \rightarrow [0, 1]$, which defines the transition probabilities between the states. Define a *reachable configuration* as one that it is possible to reach (directly or indirectly) from c_0 . Denote the set of reachable configurations by $\mathcal{C}' \subseteq \mathcal{C}$.

Theorem 2.2. *A system will reach a goal configuration in finite expected time, if there exist constants M and $\varepsilon > 0$, such that for each reachable configuration $c \in \mathcal{C}'$, there exists a path in τ from c to some configuration in G , with at most M transitions, each having a probability of at least ε .*

Armed with this theorem, we can use a *constructive* approach in our proofs — we need to show that one can always construct a path from any configuration to the goal. The strong asynchronicity assumption provides the required lower bound ε on all transition probabilities.

3 The Algorithm

3.1 Definition

We mentioned above that the proposed algorithm is a variant of an algorithm presented in [16]. Let us begin with the original algorithm in Algorithm 1, which assumes no distance sensing ability. It is presented in first person, being performed by each agent from its own point of view.

Algorithm 1. Gathering with no distance sensing

- 1: if all of the visible agents lie within a wedge which spans less than half of my visibility disc **then**
 - 2: move a step of length $\min(V \cos(\psi/2), V/2, \sigma)$ along the wedge's bisector, where ψ is the angle of the wedge.
 - 3: **else**
 - 4: do not move.
-

Beside the physical limitation σ , the step length is set so that visibility between the agents is maintained after they move, no matter what their mutual distances are. consequently, the algorithm is somewhat inefficient, as agents which are very close move (or do not move at all) much more conservatively than needed to maintain visibility. A central motivation in this work is to examine whether some crude knowledge of distance can improve the performance of the algorithm, hence the added capability to sense whether a visible agent is near or far. The new algorithm, Algorithm 2, is identical to the original, except that the agent ignores nearby agents, and the step length is different. For simplicity, we fix $\sigma = r$ in the remainder of this paper.

Algorithm 2. Gathering with crude near/far distance sensing

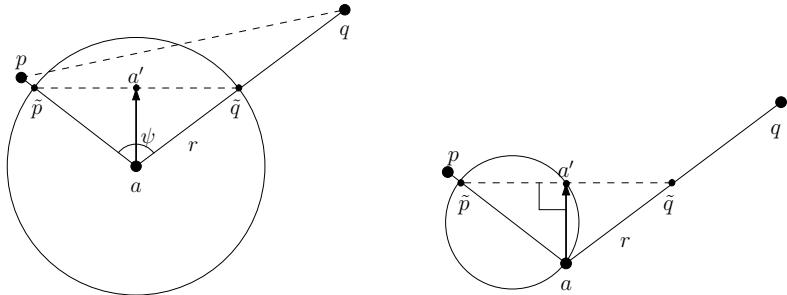
Let N be the set of all my neighbors at a distance between r and V .

- 1: if $N \neq \emptyset$ and all of the agents in N lie within a wedge which spans less than half of my visibility disc **then**
- 2: move a step of length $r \cos(\psi/2)$ along the wedge's bisector, where ψ is the angle of the wedge.
- 3: **else**
- 4: do not move.

In what follows, we call the agents that reside on the edges of the said wedge the *pulling* agents, as the agent seems to be “pulled” by these agents. We make the following simple yet important geometrical observations:

Remark 3.1. A moving agent’s destination is the midpoint between the images of its pulling agents (See Fig. 1(a)).

Remark 3.2. Given the locations of agent a and the image of one of its pulling agents \tilde{p} , the set of all possible locations of its destination a' is a circle whose diameter is the line segment $a\tilde{p}$. This is due to Thales’s classic theorem and the fact that a , a' , and \tilde{p} always form a right angle (See Fig. 1(b)). Take note of the two extreme cases: The case $a' = \tilde{p}$ corresponds to $\psi = 0$, where a is pulled only by \tilde{p} . The case $a' = a$ corresponds to $\psi = \pi$, where a doesn’t move at all.



(a) a' is the midpoint between \tilde{p} and \tilde{q} and it’s always inside the triangle apq .

(b) a' is always on the circle whose diameter is $a\tilde{p}$

Fig. 1. Geometric properties of agent movement

3.2 Proof

We now prove that Algorithm 2 indeed gathers all agents within an area of diameter r , in finite expected time. We use the constructive strategy suggested in Sect. 2.2, by showing that there always exists a bounded-time schedule that will either make the perimeter of $CH(F)$ shrink or new visibility links will be created. Chaining enough such sequences will generate a bounded-time schedule that shrinks $CH(F)$ into nothing. Then we apply this to Theorem 3.8.

Lemma 3.3. *The algorithm maintains visibility.*

Lemma 3.4. *Consider the following scenario:*

1. *An agent a moves (Others are not active);*
2. *Consequently, one or more nearby agents $b_i \notin F$ become far from a ($i = 1, \dots, k$, $k > 0$);*
3. *Next, these agents become active and move.*

For each i ,

- a. b'_i must be near a' ;
- b. b'_i must see at least one of the pulling agents of agent a ;
- c. For each j , b'_i must be near b'_j ;
- d. If, as a consequence of b_i 's movement, some agent c near b_i becomes far from it, then c must be in F .

Lemma 3.5. *A moving agent cannot move outside $CH(F)$.*

Let a_0 be the agent at a corner of the boundary of $CH(F)$, and let φ be the angle at that corner. Denote by A the isosceles triangle formed by the two boundary edges touching a_0 and a third line, such that the length of each of the triangle's edges touching a is $r \cos \frac{\varphi}{2}$.

Lemma 3.6. *For any agent in triangle A , if it becomes active and moves, it will move outside A .*

Lemmas 3.4, 3.5, and 3.6 are used to prove the following lemma, by constructing a schedule that empties the triangular corner A of F -agents, making $CH(F)$ shrink.

Lemma 3.7. *There exist constants $s^* > 0$ and $t^* \in \mathbb{N}$, such that, as long as $F \neq \emptyset$, there always exists an activity schedule that will decrease the perimeter of $CH(F)$ by at least s^* in at most t^* time steps, assuming that no new visibility links are created during that time.*

Theorem 3.8. *Given an initial configuration with a connected visibility graph, when performing Algorithm 2, the system will reach a static configuration of diameter r or less, in finite expected time.*

Proof. The initial visibility graph is globally connected and always remains so, due to Lemma 3.3, so the perimeter of $CH(F)$ is bounded. Lemma 3.7 always holds. Thus, we can always chain schedules constructed in Lemma 3.7 enough times so that the perimeter of $CH(F)$ becomes so small that it implies that its diameter is less than r . This, in turn, implies that $F = \emptyset$ by definition. The required number of these cycles is bounded, since each cycle decreases the perimeter by at least a minimal amount s^* , except possibly a bounded number of cycles during which new visibility links are created (since visibility is maintained, and there are $n(n - 1)/2$ possible links in total). Each cycle's time is bounded

by t^* , so the resulting chained schedule is of bounded length (Let's denote the bound by M). We can construct such a schedule for any given configuration. The strong asynchronicity assumption gives a minimum probability ε for each transition. Therefore, Theorem 2.2 holds, and the system will reach a state where $F = \emptyset$ in finite expected time. It is straightforward to show that, together with the fact that the visibility graph is globally connected, this implies that all agents are mutually nearby, and so the diameter of the configuration is at most r . By definition of the algorithm, this configuration is static. \square

4 A Variant — Gathering and Collapsing

When considering real robots, it is plausible to assume that in very close range $r \ll V$ the robots are able to sense each other better and perhaps communicate. Some works on formation (e.g., [21]) suggest that flocks of robots can move together as a whole, using some control hierarchy, where, for instance, a slave robot follows the movements of a master robot in the flock. In self-assembly and aggregation contexts, it is assumed that close robots can join in some rigid physical link, and effectively function and move as a single unit. In this section, we present a slightly modified model that abstracts these ideas, and a modified gathering algorithm for it.

4.1 Definitions

We use the same model and definitions as in Sect. 3.1, with the following modifications.

- We do *not* assume strong asynchronicity.
- An agent can move to the *exact* location of another *nearby* agent. Once it does so, it is permanently assimilated or *collapsed* into the other agent. From our point of view, the agent effectively disappears from the system.
- We regard the unification of nearby agents as a “low-level” action. Thus, we assume that in each time step, all collapses take place before movements. The order of collapses is arbitrary.

Algorithm 3 is quite similar to Algorithm 2, with the differences being a shorter step length (needed for our proof), and the collapsing into a nearby agent instead of just standing, when no far neighbors are visible.

4.2 Proof

The proof idea is somewhat similar to that of Algorithm 2, in the sense that we show that the convex hull (of *all* agents here, not just those in F) must shrink in finite time. Specifically, we show that the convex hull cannot expand and that some triangular corner of it must become empty once all agents wake up. We use the following definitions and notations:

Algorithm 3. Gathering with crude distance sensing and collapsing

Let N be the set of all my neighbors at a distance between r and V .

- 1: **if** $N \neq \emptyset$ and all of the agents in N lie within a wedge which spans less than half of my visibility disc **then**
 - 2: move a step of length $\frac{1}{2}r \cos(\psi/2)$ along the wedge's bisector, where ψ is the angle of the wedge.
 - 3: **else**
 - 4: Collapse into my closest neighbor.
-

- Denote the convex hull of *all* agent locations by C ;
- Let b be an agent visible by agent a . The point on the line segment ab at a distance $r/2$ from a is the *close image* of b with regard to a (Note the difference from the definition of the image of b in Sect. 2.1!).

Analogously to Remark 3.1 on Algorithm 2, we have the following remark:

Remark 4.1. A moving agent's destination is the midpoint between the close images of its pulling agents. This follows directly from the algorithm definition.

Remark 4.2. If an agent crosses a line as it moves, then at least one of its pulling agents must be across that line. This follows straightforwardly from the previous remark.

Lemma 4.3. *The algorithm maintains visibility.*

Lemma 4.4. *A moving agent cannot move outside C*

Let a_0 be the agent at a corner of the boundary of C , and let φ be the angle at that corner. Denote by A the isosceles triangle formed by the two boundary edges touching a_0 and a third line, such that the length of each of the triangle's edges touching a_0 is $\frac{r}{2} \cos \frac{\varphi}{2}$. Note the differences from the definition of A in Sect. 3.2 — It is the corner of C (not $CH(F)$) and its dimensions are halved.

Lemma 4.5. *For any agent in triangle A , if it becomes active, it will either move outside A or collapse into another agent.*

Lemma 4.6. *An agent outside triangle A cannot enter it.*

Theorem 4.7. *Given an initial configuration with a connected visibility graph, when performing Algorithm 3, the system will converge to a point, in finite time.*

Proof. According to Lemmas 4.4, 4.5, and 4.6, active agents in $C \setminus A$ remain there (or otherwise collapse), while agents in A must necessarily move to $C \setminus A$ as well (or otherwise collapse). Thus, once all agents become active, C will shrink into an area within $C \setminus A$. This will happen in finite time according to the model, and it is true for any of the convex hull's corners. In particular, this is true for the most acute corner. Thus, it can be easily shown that the perimeter of C will decrease by at least a minimum amount $s^* = r \cos \frac{\varphi^*}{2} \left(1 - \sin \frac{\varphi^*}{2}\right) > 0$, where

$\varphi^* = \pi \left(1 - \frac{2}{n}\right) < \pi$. Since the initial perimeter is bounded (because the initial configuration is globally connected), it will take a finite number of these finite-time cycles until the perimeter becomes small enough that it will imply that all agents are mutually nearby. At that point, by definition of the algorithm, all agents will collapse into each other once they become active. \square

5 Experiments

The formal proofs above guarantee that our algorithms indeed work, yet they do not provide practical bounds on their performance. To gain more insight on their behavior and compare their performance, we performed extensive simulations of the three algorithms. We tried various combinations of values of n , r , and V . We fixed $\sigma = r$ for simplicity as well as fairness, in the sense that the maximum step length became equal in both Algorithms 1 and 2. In all simulations, the initial positions were randomly selected uniformly in a large square area, while ensuring that the visibility graph is connected. During the simulation runs, each agent became active independently with probability 1/2.

The most obvious difference between the three algorithms is in their final or steady state. In Algorithm 1, the agents never stop moving. They contract into a small cluster whose diameter is around the order of r , and keep leaping one over the other ad infinitum. This is a direct result of the agents' inability to measure distances — they have no idea that they all became so close. The cluster is not tied in place and it slowly drifts in the plane. In Algorithm 2, as expected, the agents stop once they are all within an area of diameter r . Finally, in Algorithm 3, the agents collapse into a point.

Another interesting aspect is the behavior of the swarm during the convergence process. In all of our simulations of Algorithm 3, agent collapses were quite rare during the process, with most of them occurring in the very end of the run, once all agents became nearby. This is not surprising given the way we initially positioned the agents. In order for a collapse to occur before the end, there must exist an agent with no far-visible neighbors adjacent to another agent which does see a far agent, which is a somewhat special configuration (Even with the largest setting $r = V/3$, more than 96% of the collapses occurred in the end). As a result, the behavior of Algorithms 2 and 3 was very similar, aside from the slower convergence rate of Algorithm 3 due to its halved step size.

Figure 2 shows a typical run of Algorithm 2. It can be seen that the most significant movement is, as expected, in the outskirts of the swarm, where there are the most agents which are not surrounded by far neighbors (i.e., their wedge angle ψ is less than π). Initially, the swarm assumes a shape with several “tentacles” or “lobes”. These tentacles are formed in areas where there was initially a somewhat denser “mass” of agents, which pull more agents from the sparser areas. Thus, we observe what we believe is a reinforcing process of the denser areas, due to the larger number of inactive agents there (in absolute terms). All the while, the agents keep converging in a steady pace until all tentacles contract into an oval body which ultimately contracts into the final configuration. Interestingly,

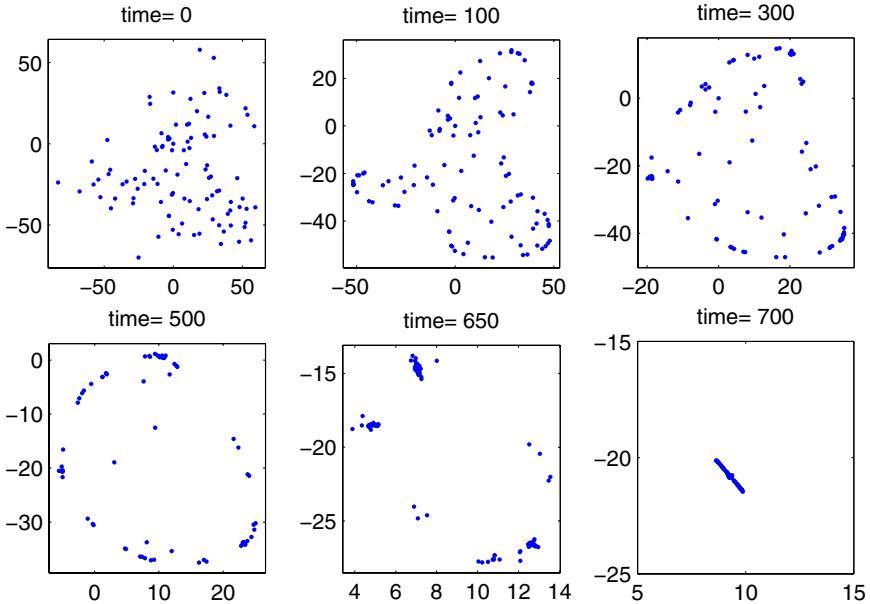
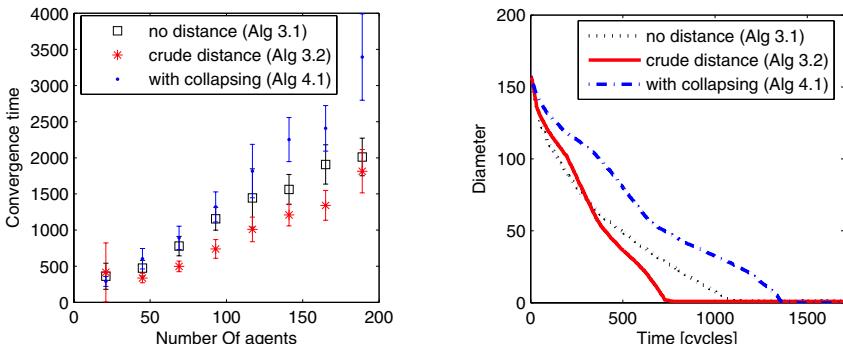


Fig. 2. A typical run of Algorithm 2. Here $n = 100$, $V = 20$ and $r = 1$. Notice that the scale is different between frames.

the swarm typically assumes a different shape with Algorithm 1 (See [16]). Here, the denser areas along the swarm perimeter soon become dense clusters which contract inwards very slowly. Viewed at large scale, the swarm perimeter assumes an almost polygonal form with sharp dense corners and straight sparse edges. The reason that these dense clusters move more slowly is that most of the agents inside are surrounded by their mates and are therefore immobile.



(a) Convergence time statistics for the three algorithms. 12 simulation runs were performed for each value of n .

(b) Diameter vs. time in typical runs of the algorithms. Here $n = 100$, $V = 20$ and $\sigma = r = 1$.

Fig. 3.

Figure 3(a) shows the statistics of our simulations. Algorithm 3 is the slowest, as expected, due to the smaller step size. It is clear that Algorithm 2 provides a significant improvement in convergence time over Algorithm 1, especially for larger swarms. Interestingly, with smaller swarms, there is no clear advantage. Figure 3(b) sheds more light on this issue. It shows the contraction of the swarm’s diameter over time in a typical run of each algorithm. Initially, Algorithm 1 is somewhat faster. This is due to the larger step size in Algorithm 1 for most wedge angles ψ (It is $\sigma = r$ versus $r \cos \psi / 2$ in Algorithm 2). However, the convergence rate continuously deteriorates whereas the convergence in Algorithm 2 remains more or less constant. This is due to the accumulation of “mass” in the corners of the swarm, which has a more adverse effect on their mobility, as explained above.

6 Conclusion

In this paper we continued to explore the gathering problem under severe distance sensing limitations. Previously, we showed how a swarm of robotic agents can gather without sensing distances at all, but only into a drifting cluster and with suboptimal performance. Now, we have shown that even the crudest form of near/far distance sensing can improve the situation a lot. With the new capability, the swarm was able to contract more swiftly and steadily into a small cluster and stop in place. Finally, introducing the capability of collapsing into nearby agents, the swarm was able to contract into a point.

Future work includes investigation of robustness to noise, failures, and per-agent variations to parameters such as r and V , and development of other algorithms under the crude distance sensing model, such as formation and flocking.

An important part of our work is the contribution to what we feel is an improved model of swarms, through the addition of the strong asynchronicity assumption to the classical semi-synchronous model. Not only does it dismiss the somewhat artificial problem of symmetry breaking (in the context of practical robotics), but it also enables a simple yet powerful approach of proving the termination of an algorithm by showing (by construction) the existence of paths to goal configurations. Strong asynchronicity can be applied to other models, such as the totally-asynchronous model of [19], and possibly to other types of distributed multi-agent systems as well.

References

1. Cieliebak, M., Flocchini, P., Prencipe, G., Santoro, N.: Solving the robots gathering problem. In: Proc. of ICALP 2003 (2003)
2. Gordon, N., Wagner, I.A., Bruckstein, A.M.: Discrete bee dance algorithms for pattern formation on a grid. In: Proc. of IEEE Intl. Conf. on Intelligent Agent Technology (IAT 2003), pp. 545–549 (2003)
3. Schlude, K.: From robotics to facility location: Contraction functions, weber point, convex core. Technical Report 403, CS, ETHZ (2003)

4. Suzuki, I., Yamashita, M.: Distributed anonymous mobile robots: Formation of geometric patterns. *SIAM Journal on Computing* 28(4), 1347–1363 (1999)
5. Flocchini, P., Prencipe, G., Santoro, N., Widmayer, P.: Gathering of autonomous mobile robots with limited visibility. In: Ferreira, A., Reichel, H. (eds.) STACS 2001. LNCS, vol. 2010. Springer, Heidelberg (2001)
6. Souissi, S., Défago, X., Yamashita, M.: Gathering asynchronous mobile robots with inaccurate compasses. In: Shvartsman, M.M.A.A. (ed.) OPODIS 2006. LNCS, vol. 4305, pp. 333–349. Springer, Heidelberg (2006)
7. Bruckstein, A.M., Cohen, N., Efrat, A.: Ants, crickets and frogs in cyclic pursuit. Technical Report CIS-9105, Technion – IIT (1991)
8. Bruckstein, A.M., Mallows, C.L., Wagner, I.A.: Probabilistic pursuits on the grid. *American Mathematical Monthly* 104(4), 323–343 (1997)
9. Marshall, J.A., Broucke, M.E., Francis, B.A.: A pursuit strategy for wheeled-vehicle formations. In: Proc. of CDC 2003, pp. 2555–2560 (2003)
10. Ando, H., Oasa, Y., Suzuki, I., Yamashita, M.: A distributed memoryless point convergence algorithm for mobile robots with limited visibility. *IEEE Trans. on Robotics and Automation* 15(5), 818–828 (1999)
11. Cohen, R., Peleg, D.: Robot convergence via center-of-gravity algorithms. In: Kralovic, R., Sýkora, O. (eds.) SIROCCO 2004. LNCS, vol. 3104, pp. 79–88. Springer, Heidelberg (2004)
12. Lin, Z., Broucke, M.E., Francis, B.A.: Local control strategies for groups of mobile autonomous agents. *IEEE Trans. on Automatic Control* 49(4), 622–629 (2004)
13. Melhuish, C.R., Holland, O., Hoddell, S.: Convoying: using chorusing to form travelling groups of minimal agents. *Robotics and Autonomous Systems* 28, 207–216 (1999)
14. Cortes, J., Martinez, S., Bullo, F.: Robust rendezvous for mobile autonomous agents via proximity graphs in arbitrary dimensions. *IEEE Trans. on Automatic Control* 51(8), 1289–1298 (2006)
15. Sugihara, K., Suzuki, I.: Distributed algorithms for formation of geometric patterns with many mobile robots. *Journal of Robotic Systems* 13(3), 127–139 (1996)
16. Gordon, N., Wagner, I.A., Bruckstein, A.M.: Gathering multiple robotic a(ge)nts with limited sensing capabilities. In: Dorigo, M., Birattari, M., Blum, C., Gambardella, L.M., Mondada, F., Stützle, T. (eds.) ANTS 2004. LNCS, vol. 3172, pp. 142–153. Springer, Heidelberg (2004)
17. Gordon, N., Wagner, I.A., Bruckstein, A.M.: A randomized gathering algorithm for multiple robots with limited sensing capabilities. In: Proc. of MARS 2005 workshop at ICINCO 2005, INSTICC (2005)
18. The Center of Intelligent Systems, Technion IIT web site,
<http://www.cs.technion.ac.il/Labs/Is1/index.html>
19. Prencipe, G.: On the feasibility of gathering by autonomous mobile robots. In: Pelc, A., Raynal, M. (eds.) SIROCCO 2005. LNCS, vol. 3499, pp. 246–261. Springer, Heidelberg (2005)
20. Efrima, A., Peleg, D.: Distributed models and algorithms for mobile robot systems. In: van Leeuwen, J., Italiano, G.F., van der Hoek, W., Meinel, C., Sack, H., Plášil, F. (eds.) SOFSEM 2007. LNCS, vol. 4362, pp. 70–87. Springer, Heidelberg (2007)
21. Fredslund, J., Mataric, M.J.: Robot formations using only local sensing and control. In: Proc. of the Intl. Symposium on Computational Intelligence in Robotics and Automation (IEEE CIRA 2001), Banff, Alberta, Canada, pp. 308–313 (2001)

Integration of ACO in a Constraint Programming Language

Madjid Khichane^{1,2}, Patrick Albert¹, and Christine Solnon²

¹ ILOG SA, Gentilly, France

{mkhichane,palbert}@ilog.fr

² LIRIS CNRS UMR 5205, University of Lyon I, Villeurbanne, France

christine.solnon@liris.cnrs.fr

Abstract. We propose to integrate ACO in a Constraint Programming (CP) language. Basically, we use the CP language to describe the problem to solve by means of constraints and we use the CP propagation engine to reduce the search space and check constraint satisfaction; however, the classical backtrack search of CP is replaced by an ACO search. We report first experimental results on the car sequencing problem and compare different pheromone strategies for this problem.

1 Introduction

Our motivations mainly come from the two following observations:

- Ant Colony Optimization (ACO) has been successfully applied to a wide range of combinatorial optimization problems [1]; however most works have focused on designing efficient ACO algorithms for solving specific problems, but not on integrating these algorithms within declarative languages so that solving a new problem with this approach usually implies a lot of procedural programming;
- Constraint Programming (CP) languages provide high level features to declaratively model problems by means of constraints; however, most CP solvers are based on a systematic “Branch and Propagate” exploration of the search space, and fail to solve some hard problems within a reasonable time limit.

Hence, we investigate the integration of ACO within a CP language. Our research is based upon ILOG Solver [2], and we use its modeling language to describe the problem to solve by means of constraints and its propagation engine to reduce the search space and check constraint satisfaction; however, the search of solutions is guided by ACO. This approach has the benefit of reusing all the work done by ILOG at the modeling level as well as the code dedicated to constraint propagation and verification. We can as well test different variations of our ideas on a large benchmark library. Note that this work could be easily extended to other CP languages, such as, e.g., CHOCO or GECODE.

It is worth reporting that an hybridization of ACO and CP has already been proposed in [3] to solve a timetabling problem which contains hard constraints,

that must be satisfied, and has an objective function to optimize. In this approach, constraint propagation is used to build feasible solutions that satisfy all hard constraints while ACO is used to find high quality solutions with respect to the objective function. Our approach in this paper is rather different as ACO is used to guide a search procedure aiming at satisfying all the constraints.

The paper is organized as follows. In the next section, we briefly recall some definitions and terminology about CP. Section 3 describes the basic *Ant-CP* algorithm for solving constraint satisfaction problems. Section 4 shows how this algorithm may be used to solve the car sequencing problem and introduces different pheromone strategies and different heuristics for this problem. Section 5 experimentally compares these different variants of *Ant-CP*.

2 Background

A Constraint Satisfaction Problem (CSP) [4] is defined by a triple (X, D, C) such that X is a finite set of variables, D is a function that maps every variable $x_i \in X$ to its domain $D(x_i)$, that is, the finite set of values that can be assigned to x_i , and C is a set of constraints, that is, relations between some variables which restrict the set of values that can be assigned simultaneously to these variables.

Solving a CSP involves assigning values to variables so that constraints are satisfied. More formally, an *assignment* is a set of variable-value couples, noted $\langle x_i, v \rangle$ and corresponding to the assignment of a value $v \in D(x_i)$ to a variable x_i . The variables assigned in an assignment \mathcal{A} are denoted by $var(\mathcal{A})$. An assignment \mathcal{A} is *partial* if some variables are not assigned in \mathcal{A} , i.e., $var(\mathcal{A}) \subset X$; it is *complete* if all variables are assigned, i.e., $var(\mathcal{A}) = X$. An assignment \mathcal{A} is *consistent* if it does not violate any constraint. A *solution of a CSP* (X, D, C) is a complete and consistent assignment.

CSPs are solved in a generic way by constraint solvers which are embedded within CP languages. These constraint solvers are usually based on a systematic exploration of the search space: starting from an empty assignment, they incrementally extend a partial consistent assignment by choosing a non assigned variable and a consistent value for it until either the current assignment is complete (a solution has been found) or the current assignment cannot be extended without violating constraints (the search must backtrack to a previous choice point and try another extension). To reduce the search space, this exhaustive exploration of the search space is combined with constraint propagation techniques: each time a variable is assigned to a value, constraints are propagated to filter the domains of the variables that are not yet assigned, i.e., to remove values that are not consistent with respect to the current assignment. If constraint propagation detects an inconsistency or if it removes all values from a domain, the search must backtrack. Different levels of consistency may be considered (e.g., node consistency or arc consistency); some consistencies are stronger than others, removing more values from the domains, but have also higher time complexities.

Algorithm 1. *Ant-CP* procedure

Input: A CSP (X, D, C) , a pheromone strategy Φ , a heuristic factor η
Output: A consistent (partial or complete) assignment for (X, D, C)

```

1 Initialize all pheromone trails of  $\Phi$  to  $\tau_{max}$ 
2 repeat
3   foreach  $k$  in  $1..nbAnts$  do
4     /* Construction of a consistent assignment  $\mathcal{A}_k$  */
5      $\mathcal{A}_k \leftarrow \emptyset$ 
6     repeat
7       Select a variable  $x_i \in X$  so that  $x_i \notin var(\mathcal{A}_k)$ 
8       Choose a value  $v \in D(x_i)$ 
9       Add  $\langle x_i, v \rangle$  to  $\mathcal{A}_k$ 
10      Propagate constraints to filter domains of  $D$ 
11    until  $var(\mathcal{A}_k) = X$  or Failure ;
12  Update pheromone trails of  $\Phi$  using  $\{\mathcal{A}_1, \dots, \mathcal{A}_{nbAnts}\}$ 
13 until  $var(\mathcal{A}_i) = X$  for some  $i \in \{1..nbAnts\}$  or max cycles reached ;
14 return the largest constructed assignment

```

3 Description of *Ant-CP*

Some ACO algorithms have been previously proposed for solving CSPs [5,6,7,8]. In these algorithms, ants iteratively build complete assignments (that assign a value to every variable) that may violate constraints, and their goal is to minimize the number of constraint violations; a solution is found when the number of constraint violations is null.

In this paper, we investigate a new ACO framework for solving CSPs: ants iteratively build partial assignments (such that some variables may not be assigned to a value) that do not violate constraints, and their goal is to maximize the number of assigned variables; a solution is found when all variables are assigned. This new ACO framework may be combined with the propagation engine of ILOG Solver in a very straightforward way.

More precisely, the proposed algorithm for solving CSPs, called *Ant-CP*, is sketched in Algorithm 1. First, pheromone trails are initialized to some given value τ_{max} . Then, at each cycle (lines 2-12), each ant k constructs a consistent assignment \mathcal{A}_k (lines 4-10): starting from an empty assignment, the ant iteratively chooses a variable which is not yet assigned and a value to assign to this variable; this variable assignment is added to \mathcal{A}_k , and constraints are triggered which might in turn narrow the domains of non assigned variables, trigger new assignments, or detect a failure; this process is iterated until either all variables have been assigned (i.e., a solution has been found) or the propagation step detects a failure. Once every ant has constructed an assignment, pheromone trails are updated. The algorithm stops iterating either when an ant has found a solution, or when a maximum number of cycles has been performed.

In the next paragraphs, we define the pheromone strategy Φ used to bias the search, and we describe the variable selection, value selection, propagation and pheromone updating steps.

Pheromone Strategy. The pheromone strategy, denoted by Φ , is a parameter of *Ant-CP* and is defined by a triple $\Phi = (S, \tau, comp)$ such that:

- S is the set of components on which ants lay pheromone;
- τ is a function which defines how pheromone trails of S are used to bias the search. More precisely, given a partial assignment \mathcal{A} , a variable $x_i \notin var(\mathcal{A})$, and a value $v \in D(x_i)$, the function $\tau(\mathcal{A}, x_i, v)$ returns the value of the pheromone factor which evaluates the learnt desirability of adding $\langle x_i, v \rangle$ to the partial assignment \mathcal{A} ;
- $comp$ is a function which defines the set of components on which pheromone is laid when rewarding an assignment \mathcal{A} , i.e., the function $comp(\mathcal{A})$ returns the set of components associated with \mathcal{A} .

The goal of the pheromone strategy is to learn from previous constructions which decisions have allowed ants to build good assignments, and to use this information to bias further constructions. The default pheromone strategy, denoted by $\Phi_{default}$, is defined as follows:

- ants lay pheromone on variable-value couples, i.e.,

$$S = \{\tau_{\langle x_i, v \rangle} | x_i \in X, v \in D(x_i)\}$$

so that each pheromone trail $\tau_{\langle x_i, v \rangle}$ represents the learnt desirability of assigning value v to x_i ;

- the pheromone factor is defined by $\tau(\mathcal{A}, x_i, v) = \tau_{\langle x_i, v \rangle}$;
- the set of components associated with an assignment is

$$comp(\mathcal{A}) = \{\tau_{\langle x_i, v \rangle} | \langle x_i, v \rangle \in \mathcal{A}\}$$

For specific problems, the user may design other pheromone strategies. In this case, he must define the triple $(S, \tau, comp)$. We shall propose and compare two other pheromone strategies for the car sequencing problem in the next section.

Selection of a Variable. When constructing an assignment, the order in which the variables are assigned is rather important and variable ordering heuristics have been studied widely in the context of backtrack search [4,9]. These heuristics can be used as well in our context of greedy construction of assignments. For example, we can use the different variable ordering heuristics that are predefined in ILOG solver such as, e.g., the `IloChooseMinSizeInt` heuristic which selects the variable that has the smallest domain.

Choice of a Value. Once a variable x_i has been selected, ants have to choose a value v in the domain $D(x_i)$ of x_i . Note that this domain may have been reduced by constraint propagation and may not contain all values of the initial domain of x_i . The main contribution of ACO for solving CSPs is to provide a generic heuristic for choosing values. The value v to be assigned to a variable x_i is randomly chosen within $D(x_i)$ with respect to a probability $p(x_i, v)$ which depends on a pheromone factor $\tau(\mathcal{A}, x_i, v)$ —which reflects the past experience

of the colony regarding the addition of $\langle x_i, v \rangle$ to the partial assignment \mathcal{A} — and a heuristic factor $\eta(\mathcal{A}, x_i, v)$ —which is problem-dependent, i.e.,

$$p(x_i, v) = \frac{[\tau(\mathcal{A}, x_i, v)]^\alpha [\eta(\mathcal{A}, x_i, v)]^\beta}{\sum_{w \in D(x_i)} [\tau(\mathcal{A}, x_i, w)]^\alpha [\eta(\mathcal{A}, x_i, w)]^\beta} \quad (1)$$

where α and β are two parameters that determine the relative weights of pheromone and heuristic information.

Constraint Propagation. Each time a variable is assigned to a value, a propagation algorithm is called. This algorithm filters the domains of the non assigned variables: it removes the values that are inconsistent with respect to some partial consistencies such as, e.g., node-consistency, arc-consistency or path-consistency [4]. If the domain of a variable becomes a singleton, then the partial assignment \mathcal{A}_k is completed by the assignment of this variable and the propagation process is continued. If the domain of a variable becomes empty, then *Failure* is returned.

One may consider different propagation algorithms, that ensure different partial consistencies. In our premininary experiments, we have used the default propagation algorithm integrated to ILOG solver.

Pheromone Updating Step. Once every ant has constructed an assignment, pheromone trails are updated according to ACO: first, they are decreased by multiplying them by $(1 - \rho)$ (where $\rho \in [0; 1]$ is the evaporation rate); then, they are rewarded with respect to their contribution to the construction of good assignments. More precisely, let *BestOfCycle* be the set of all the best assignments constructed during the cycle, that is,

$$\text{BestOfCycle} = \{\mathcal{A}_i \in \{\mathcal{A}_1, \dots, \mathcal{A}_{nbAnts}\} \mid \#\text{var}(\mathcal{A}_i) \text{ is maximal}\}$$

For each assignment $\mathcal{A}_k \in \text{BestOfCycle}$, pheromone trails associated with pheromone components of \mathcal{A}_k are increased by a quantity δ_τ which is proportionally inverse to the gap of sizes between \mathcal{A}_k and the largest assignment \mathcal{A}_{best} built since the beginning of the search (including the current cycle), i.e.,

$$\delta_\tau = 1 / (1 + \#\mathcal{A}_{best} - \#\mathcal{A}_k)$$

The set of pheromone components associated with a given assignment depends on the considered pheromone strategy and is defined by *comp*.

Note that *Ant-CP* follows the MAX-MIN Ant System scheme [10] so that pheromone trails are bounded between two given bounds τ_{min} and τ_{max} .

4 Using *Ant-CP* to Solve the Car Sequencing Problem

The car sequencing problem involves scheduling cars along an assembly line in order to install options (e.g., sun-roof or air-conditioning) on them. Each option is installed by a different station, designed to handle at most a certain percentage of the cars passing along the assembly line, and the cars requiring this option

must be spaced so that the capacity of the station is never exceeded. More precisely, a car sequencing problem is defined by a tuple (C, O, p, q, r) , where C is the set of cars to be produced and O is the set of different options. The two functions $p:O \rightarrow \mathbb{N}$ and $q:O \rightarrow \mathbb{N}$ define the capacity constraint associated with each option $o_i \in O$, i.e., for any sequence of $q(o_i)$ consecutive cars on the line, at most $p(o_i)$ of them may require o_i . The function $r:C \times O \rightarrow \{0, 1\}$ defines option requirements, i.e., for each car $c_i \in C$ and for each option $o_j \in O$, $r(c_i, o_j)$ returns 1 if o_j must be installed on c_i , and 0 otherwise.

Solving a car sequencing problem involves finding an arrangement of the cars in a sequence, defining the order in which they will pass along the assembly line, such that the capacity constraints are met. This problem is NP-hard [11]. A more general problem –which introduces paint batching constraints and priority levels for capacity constraints– has been proposed by Renault for the ROADEF challenge in 2005 [12].

4.1 CP Model

The car sequencing problem has been first introduced in the CP community in 1988 [13]. Since then, it has been very often used to evaluate CP solvers and it is the first problem of the CSP library *CSplib* [14]. To evaluate *Ant-CP*, we have considered a classical CP model for the car sequencing problem which basically corresponds to the first model described in the user’s manual of ILOG solver. In order to reduce the search space, this model introduces the concept of car classes: all cars requiring a same set of options are grouped into a same car class.

There are two different kinds of variables:

- A slot variable x_i is associated with each position i in the sequence of cars. This variable corresponds to the class of the i^{th} car in the sequence and its domain is the set of all car classes.
- An option variable y_i^j is associated with each position i in the sequence and each option j . This variable is assigned to 1 if option j has to be installed on the i^{th} car of the sequence, and 0 otherwise, so that its domain is $\{0, 1\}$.

There are three different kinds of constraints:

- Link constraints specify the link between slot and option variables, i.e., $y_i^j = 1$ iff option j has to be installed on x_i .
- Capacity constraints specify that station capacities must not be exceeded, i.e., for each option j and each subsequence of q_j cars, a linear inequality specifies that the sum of the corresponding option variables must be smaller or equal to p_j .
- Demand constraints specify, for each car class, the number of cars of this class that must be sequenced.

4.2 Variable Ordering Heuristic

In experiments reported in Section 5 we have used a classical sequential variable ordering heuristic, which consists in assigning slot variables in the order defined

by the sequence of cars, i.e., x_1, x_2, x_3, \dots . Note that option variables y_i^j are assigned by propagation when the corresponding slot variable x_i is assigned.

4.3 Pheromone Strategies

As pheromone is at the core of the efficiency of any ACO implementation, we explore the impact of its structure: besides the default pheromone strategy Φ_{default} (which associates a trail with every couple (x_i, j) such that x_i is the variable associated with position i and j is a car class), we consider two other strategies which will be experimentally compared in the next section.

Pheromone Strategy Φ_{classes} . This pheromone strategy has been introduced in [15]. The set S associates a trail $\tau_{(v,w)}$ with every couple of car classes (v, w) . This pheromone trail represents the learnt desirability of sequencing a car of class w just after a car of class v or, in other words, of assigning the value w to a variable x_i when x_{i-1} has just been assigned to v .

For this pheromone strategy, the pheromone factor is equal to the pheromone trail between the last assigned car class and the candidate car class (this factor is equal to one when assigning the first variable), i.e., if $i > 1$ and $\langle x_{i-1}, w \rangle \in \mathcal{A}$, then $\tau(\mathcal{A}, x_i, v) = \tau_{(w,v)}$, otherwise $\tau(\mathcal{A}, x_i, v) = 1$.

When updating pheromone trails, pheromone is laid on couples of consecutively assigned values, i.e., $\text{comp}(\mathcal{A}) = \{\tau_{(v,w)} | \{\langle x_i, v \rangle, \langle x_{i+1}, w \rangle\} \subseteq \mathcal{A}\}$.

Pheromone Strategy Φ_{cars} . This pheromone strategy has been introduced in [8]. The set S associates a trail $\tau_{(v,j,w,k)}$ with each couple of car classes (v, w) and each $j \in [1; \#v]$ and $k \in [1; \#w]$ where $\#v$ and $\#w$ are the number of cars within the classes v and w respectively. This trail represents the learnt desirability of sequencing the k^{th} car of class w just after the j^{th} car of class v .

In this case, the pheromone factor $\tau(\mathcal{A}, x_i, v)$ is equal to 1 for the first car, i.e., $\tau(\mathcal{A}, x_i, v) = 1$ if $i = 1$. Otherwise $\tau(\mathcal{A}, x_i, v) = \tau_{(w,j,v,k+1)}$ where w is the value assigned to x_{i-1} in \mathcal{A} , j is the number of variables assigned to w in \mathcal{A} , and k is the number of variables assigned to v in \mathcal{A} .

When updating pheromone trails, pheromone is laid on couples of consecutively sequenced cars, i.e.,

$$\text{comp}(\mathcal{A}) = \{\tau_{(v,j,w,k+1)} | \{\langle x_l, v \rangle, \langle x_{l+1}, w \rangle\} \subseteq \mathcal{A} \text{ and } j = \#\{\langle x_m, v \rangle | m \leq l\} \text{ and } k = \#\{\langle x_m, w \rangle | m \leq l\}\}.$$

4.4 Heuristic Factors for the Car Sequencing Problem

In the transition probability defined by eq. (1), the pheromone factor $\tau(\mathcal{A}, x_i, v)$ — which represents the past experience of the colony — is combined with a heuristic factor $\eta(\mathcal{A}, x_i, v)$ — which is problem-dependent. We now introduce two different problem-dependent heuristics for the car sequencing problem; these heuristics will be compared in Section 5.

Dynamic Sum of Utilisation Rates (DSU). Smith [9] has introduced value ordering heuristics based on option utilization rates: the utilization rate of an option i is the ratio of the number of cars requiring i with respect to the maximum number of cars in the sequence which could have i while satisfying its capacity constraint. Gotlieb et al. [16] have compared different value ordering heuristics based on option utilization rates, and have shown that one of the best performing heuristics is the so-called Dynamic Sum of Utilization rates (DSU), i.e.,

$$\eta(\mathcal{A}, x_i, v) = \sum_{o_j \in \text{reqOptions}(v)} \frac{\text{reqSlots}(o_j, n_j)}{N}$$

where $\text{reqOptions}(v)$ is the set of options that are required by cars of class v , N is the number of cars that have not yet been sequenced in \mathcal{A} , n_j is the number of cars that require option o_j and have not yet been sequenced in \mathcal{A} , and $\text{reqSlots}(o_j, n_j)$ is a lower bound of the number of slots needed to sequence n_j cars that require option o_j without violating capacity constraints. For defining $\text{reqSlots}(o_j, n_j)$, we have considered the formula introduced in [17], i.e., if $n_j \% p_j = 0$ then $\text{reqSlots}(o_j, n_j) = q_j * n_j / p_j - (q_j - p_j)$, otherwise $\text{reqSlots}(o_j, n_j) = q_j * (n_j - n_j \% p_j) / p_j + n_j \% p_j$.

Dynamic Sum of Utilisation Rates with Propagation (DSU+P). We can exploit utilization rates to detect inconsistencies and filter variable domains:

- when the utilization rate of an option becomes greater than 1 (i.e., when $\text{reqSlots}(o_j, n_j)$ becomes greater than N) one can conclude that it is not possible to complete the sequence without violating constraints so that one can stop the current assignment construction on a failure;
- when the utilization rates of one or more options become equal to 1, we can remove from the domain of the next variable every car class which does not require all options that have an utilization rate equal to 1.

5 Experimental Results

Test Suite. Satisfiable instances of the library *CSplib* [14] with 100 cars (4 instances described in [18]) and 200 cars (70 instances described in [19]) are all easily solved by *Ant-CP*. Hence, we consider a harder test suite which is described in [20]. All instances have 8 options and 20 car classes; capacity constraints are randomly generated in such a way that $\forall o_i \in O, 1 \leq p(o_i) \leq 3$ and $p(o_i) < q(o_i) \leq p(o_i) + 2$. This test suite contains 32 instances with 100 cars, 21 instances with 300 cars and 29 instances with 500 cars. All these instances are satisfiable.

Considered *Ant-CP* Instanciations. We compare the three pheromone strategies Φ_{default} , Φ_{classes} , and Φ_{cars} . To evaluate the influence of the pheromone on the solution process, we also consider a strategy without pheromone, denoted by Φ_\emptyset : in this case, the set S is the empty set and the pheromone factor $\tau(\mathcal{A}, x_k, v)$ is set to 1 so that *Ant-CP* behaves like a greedy randomized approach which chooses values with respect to the heuristic factor only.

We also compare the two heuristic factors DSU and DSU+P. We note $\text{Ant-CP}(\Phi, h)$ the *Ant-CP* instantiation obtained with the pheromone strategy $\Phi \in \{\Phi_\emptyset, \Phi_{\text{classes}}, \Phi_{\text{cars}}, \Phi_{\text{default}}\}$ and the heuristic $h \in \{\text{DSU}, \text{DSU+P}\}$.

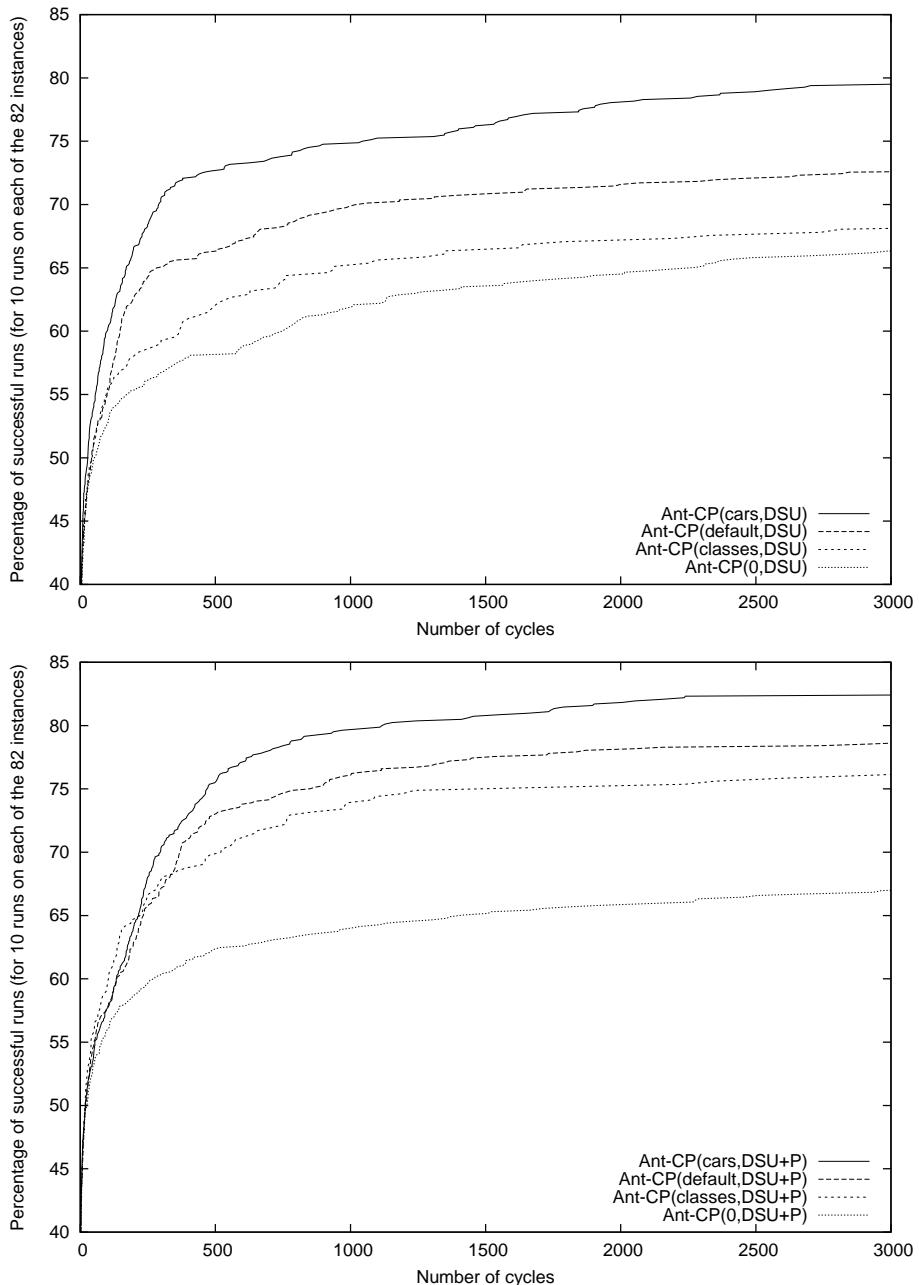


Fig. 1. Comparison of different instantiations of $\text{Ant-CP}(\Phi, h)$, with $\Phi \in \{\Phi_{\text{default}}, \Phi_{\text{classes}}, \Phi_{\text{cars}}, \Phi_{\emptyset}\}$ and $h \in \{\text{DSU}, \text{DSU} + P\}$: each curve plots the evolution of the percentage of runs that have found a solution with respect to the number of cycles (for 10 runs for each of the 82 instances)

Parameter Setting. Parameters have been set as follows: $\alpha = 1$, $\beta = 6$, $\rho = 0.02$, $nbAnts=30$, $\tau_{min} = 0.01$, and $\tau_{max} = 4$.

Comparison of Ant-CP Instanciations. Fig. 1 compares the four different pheromone strategies when using the DSU heuristic (upper curves) and then when using the DSU+P heuristic (lower curves). One notes that after 3000 cycles, using pheromone increases the success rate from 66.32% for $Ant-CP(\Phi_\emptyset, DSU)$ to 79.39% for $Ant-CP(\Phi_{cars}, DSU)$, 72.56% for $Ant-CP(\Phi_{default}, DSU)$ and 68.29% for $Ant-CP(\Phi_{classes}, DSU)$. Hence, on these instances, the best pheromone strategy is Φ_{cars} ; the default pheromone strategy is worse than Φ_{cars} , but better than $\Phi_{classes}$.

The DSU+P heuristic usually obtains better results than DSU. However, the improvement depends on the considered pheromone strategy: it is not significant when pheromone is ignored (the success rate of Φ_\emptyset is increased from 66.32% to 66.97%); it is rather important for the pheromone strategies $\Phi_{classes}$ and $\Phi_{default}$ (success rates are increased from 68.29% to 74.39% for $\Phi_{classes}$ and from 72.56% to 78.54% for $\Phi_{default}$); it is not as important for Φ_{cars} (the success rate is increased from 79.39% to 82.32%).

Let us finally note that, given a heuristic, the four variants spend nearly the same CPU time to perform one cycle, as most of the time is spent by propagation procedures. However, cycles are performed quicker with the DSU+P heuristic than with the DSU heuristic. Indeed, DSU+P filters variable domains and detects earlier some inconsistencies. Hence, for the instances with 100 cars, 3000 cycles are performed in 5 minutes with the DSU heuristic whereas they are performed in 3 minutes with the DSU+P heuristic (on a 2GHz Intel Core Duo).

6 Conclusion

These first experiments on the Car Sequencing problem show that integrating an ACO search might be designed as a simple extension of a modular CP language such as ILOG Solver. It is worth noting that thanks to the modular nature of ILOG Solver that separates the modeling of the problem from the computation of its solution, the CP model used to describe the sequencing problem does not depend on the search technique: one can try or combine other search methods without changing the problem description.

First experimental results are very promising. Indeed, classical CP solvers based on exhaustive tree search approaches are still not able to solve within a reasonable amount of time all instances of *CSPLib* with 100 and 200 cars, even when using dedicated filtering algorithms such as, e.g., those proposed in [18,21,22]. All these instances are easily solved by *Ant-CP* (whatever the pheromone strategy is): the 70 instances with 200 cars (resp. 4 instances with 100 cars) are solved in less than a second (resp. less than a minute). As a comparison, filterings introduced in [22] for the “sequence” constraint can solve less than half of these instances in less than 100 seconds when they are combined with the default tree search of ILOG Solver on a Pentium 4 sequenced at 3.2 Ghz.

Of course, these experiments should be pursued on other problems to further validate our approach.

These first results might be further enhanced by adapting the propagation algorithms to the specificities of ACO. Indeed, propagation algorithms integrated in most CP solvers have been designed to fit a procedure that enables backtracking on the choice points. Such algorithms have thus to maintain data structures enabling the restitution at each backtrack of the context of the previous choice point. The allocation and management of these data structures bring a cost both in terms of memory and Cpu time which is necessary in the context of a backtrack-based search procedure but not in the context of our ACO inspired search method which never backtracks.

A similar remark might be done with respect to a language such as COMET that does not rely on a backtracking tree search but on local search. Indeed, Van Hentenryck and Michel have shown in [23] that the COMET language may be used to implement an ACO algorithm in a very declarative way. However, COMET is dedicated to local search which explores the search space by iteratively applying elementary moves to a current configuration. In order to efficiently select the best move to be applied, COMET maintains a set of data structures that support the incremental evaluation of invariant properties after each elementary move. These data structures can be used to support an ACO search (they support the incremental evaluation of the heuristic factor at each step of the construction) but again, they maintain more information than necessary because choices made in ACO during the greedy construction of the solution are never revised.

Still, we did not explore in both cases (tree-search based solver such as ILOG Solver, or local-search based solver such as COMET) how well the cost of extra data structures could be compensated by an effective combination of the ACO search procedure and the default search of the host solver. For example one could do a limited tree-search based exploration of the remaining search space when a constraint triggers a failure in ILOG Solver, or perform some local search at the end of the exploration of each or some of the artificial ants.

References

1. Dorigo, M., Stützle, T.: *Ant Colony Optimization*. MIT Press, Cambridge (2004)
2. ILOG: Ilog solver user's manual. Technical report, ILOG (1998)
3. Meyer, B., Ernst, A.: Integrating aco and constraint propagation. In: Dorigo, M., Birattari, M., Blum, C., Gambardella, L.M., Mondada, F., Stützle, T. (eds.) *ANTS 2004*. LNCS, vol. 3172, pp. 166–177. Springer, Heidelberg (2004)
4. Tsang, E.: *Foundations of Constraint Satisfaction*. Academic Press, London (1993)
5. Schoofs, L., Naudts, B.: Solving csp's with ant colonies. In: *ANTS* (2000)
6. Roli, A., Blum, C., Dorigo, M.: ACO for maximal constraint satisfaction problems. In: *Meta-heuristics International Conference (MIC)* (2001)
7. Solnon, C.: Ants can solve constraint satisfaction problems. *IEEE Transactions on Evolutionary Computation* 6(4), 347–357 (2002)
8. Solnon, C.: Combining two pheromone structures for solving the car sequencing problem with Ant Colony Optimization. In: *EJOR* (to appear, 2008)

9. Smith, B.: Succeed-first or fail-first: A case study in variable and value ordering heuristics. In: PaCT 1997, pp. 321–330 (1996)
10. Stützle, T., Hoos, H.: MAX-MIN Ant System. Future Generation Computer Systems, special issue on Ant Algorithms 16, 889–914 (2000)
11. Kis, T.: On the complexity of the car sequencing problem. Operations Research Letters 32, 331–335 (2004)
12. Solnon, C., Cung, V., Nguyen, A., Artigues, C.: The car sequencing problem: overview of state-of-the-art methods and industrial case-study of the ROADEF 2005 challenge problem. In: EJOR (to appear, 2008)
13. Dincbas, M., Simonis, H., van Hentenryck, P.: Solving the car-sequencing problem in constraint logic programming. In: Kodratoff, Y. (ed.) Proceedings of ECAI-1988, pp. 290–295 (1988)
14. Gent, I., Walsh, T.: CSPLib: A benchmark library for constraints. Technical report (1999), <http://csplib.cs.strath.ac.uk/>
15. Gravel, M., Gagné, C., Price, W.: Review and comparison of three methods for the solution of the car-sequencing problem. In: JORS (2004)
16. Gottlieb, J., Puchta, M., Solnon, C.: A study of greedy, local search and ACO approaches for car sequencing problems. In: Raidl, G.R., Cagnoni, S., Cardalda, J.J.R., Corne, D.W., Gottlieb, J., Guillot, A., Hart, E., Johnson, C.G., Marchiori, E., Meyer, J.-A., Middendorf, M. (eds.) EvoIASP 2003, EvoWorkshops 2003, EvoSTIM 2003, EvoROB/EvoRobot 2003, EvoCOP 2003, EvoBIO 2003, and Evo-MUSART 2003. LNCS, vol. 2611. Springer, Heidelberg (2003)
17. Boysen, N., Fliedner, M.: Comments on solving real car sequencing problems with ant colony optimization. EJOR 182(1), 466–468 (2007)
18. Regin, J.C., Puget, J.F.: A filtering algorithm for global sequencing constraints. In: Smolka, G. (ed.) CP 1997. LNCS, vol. 1330, pp. 32–46. Springer, Heidelberg (1997)
19. Lee, J., Leung, H., Won, H.: Performance of a comprehensive and efficient constraint library using local search. In: 11th Australian JCAI. LNCS (LNAI). Springer, Heidelberg (1998)
20. Perron, L., Shaw, P.: Combining forces to solve the car sequencing problem. In: Régin, J.-C., Rueher, M. (eds.) CPAIOR 2004. LNCS, vol. 3011, pp. 225–239. Springer, Heidelberg (2004)
21. van Hoeve, W.J., Pesant, G., Rousseau, L.M., Sabharwal, A.: Revisiting the sequence constraint. In: Benhamou, F. (ed.) CP 2006. LNCS, vol. 4204, pp. 620–634. Springer, Heidelberg (2006)
22. Brand, S., Narodytska, N., Quimper, C.-G., Stuckey, P.J., Walsh, T.: Encodings of the sequence constraint. In: Bessière, C. (ed.) CP 2007. LNCS, vol. 4741, pp. 210–224. Springer, Heidelberg (2007)
23. Hentenryck, P.V., Michel, L.: Constraint-based local search. MIT Press, Cambridge (2005)

Learning from House-Hunting Ants: Collective Decision-Making in Organic Computing Systems*

Arne Brutschy¹, Alexander Scheidler², Daniel Merkle³,
and Martin Middendorf²

¹ IRIDIA, CoDE, Université Libre de Bruxelles, Brussels, Belgium
arne.brutschy@ulb.ac.be

² Parallel Computing and Complex Systems Group, Computer Science Department
University of Leipzig, Leipzig, Germany
{scheidler,middendorf}@uni-leipzig.de

³ Department of Mathematics and Computer Science
University of Southern Denmark, Odense, Denmark
daniel@imada.sdu.dk

Abstract. This paper proposes ant-inspired strategies for self-organized and decentralized collective decision-making in computing systems which employ reconfigurable units. The particular principles used for the design of these strategies are inspired by the house-hunting of the ant *Temnothorax albipennis*. The considered computing system consists of two types of units: so-called worker units that are able to execute jobs that come into the system, and scout units that are additionally responsible for the reconfiguration process of all units. The ant-inspired strategies are analyzed experimentally and are compared to a non-adaptive reference strategy. It is shown that the ant-inspired strategies lead to a collective decentralized decision process through which the units are able to find good configurations that lead to a high system throughput even in complex configuration spaces.

1 Introduction

With the increasing complexity of modern computing systems, new design paradigms become important with regards to solving the resulting management and reliability issues. Computing systems that follow the paradigms of Autonomic Computing (e.g., [1]) or Organic Computing (e.g., [2,3]) should ideally be able to address certain critical tasks autonomously, follow the principle of self-organization, and possess so-called self-x properties. Examples of self-x properties are self-management, self-reconfiguration, self-optimization, self-servicing and of course self-organization itself.

Closely related to the property of self-reconfiguration is the field of dynamically reconfigurable hardware (e.g., [4]). A central problem in systems that use dynamically reconfigurable hardware is to find a good reconfiguration strategy

* Supplementary online material: <http://iridia.ulb.ac.be/supp/IridiaSupp2008-006/>

for deciding when to carry out reconfiguration operations and which configurations to use. In these systems, a trade-off exists between increasing reconfiguration costs (i.e. when reconfiguration operations are carried out more frequently) and the possible speed gain resulting from adapted configurations.

Because designing computing systems that work autonomously and are able to make collective decisions is complicated, principles of self-organized decision-making in social insects (see, e.g., [5,6]) have become an important source of inspiration for system designers. This approach has been successfully applied to systems in different technical domains, e.g., scheduling [7], task-allocation in networks [8] and robotics [9].

In this paper, we propose a nature-inspired reconfiguration strategy for systems that consist of reconfigurable units. The proposed strategy is inspired by principles that are used by the ant *Temnothorax albipennis* for house-hunting. To our best knowledge, these principles have been exploited so far only once for the design of a technical system [10]. It is shown that a cluster of decentralized and self-organized reconfigurable units which uses the proposed strategy can adapt itself to a dynamic environment by making an efficient compromise between accuracy and speed of the decision-making process.

The paper is organized as follows. A brief overview on the biological system that inspired the proposed strategy is given in Section 2. Section 3 describes the model of the computing system with reconfigurable units. Some details about the reconfiguration strategies are explained in Section 4. Experiments and results are presented in Section 5. Conclusions are given in Section 6.

2 House-Hunting in *Temnothorax Albipennis*

Temnothorax albipennis is an ant species that has small workers and lives in small colonies. As it tends to build the nests in structurally unstable places such as in crevices and under rocks, the ant has to emigrate frequently [11]. Due to its peculiar properties, and because having to deal with only a small number of individuals is an advantage, the emigration behaviour of *T. albipennis* has been studied thoroughly [11,12,13]. The details of the emigration process are described in the following.

When emigration is necessary, the scout ants (which form approximately 20–30% of the colony [14]) start to search the surrounding environment for a new nest site. Upon finding a candidate site, a scout starts to assess the site according to several criteria (e.g., size and darkness). If the scout considers the site to be superior to the current nest, it tries to get a “second opinion” by guiding another scout to the candidate nest site. The guidance is accomplished by *tandem-running*, which means that one ant teaches another ant the route by leading it while keeping close physical contact to allow bidirectional feedback. This makes this technique slow and therefore costly [15]. As soon as the other scout reaches the candidate nest site, it assesses it and, if it considers it to be good, starts recruiting another scout as well. Scouts delay recruitment to a candidate nest site by a time that is inversely proportional to the perceived quality

of the site. This behaviour ensures that better nests will attract scouts faster, thus making the emigration an autocatalytic process.

As soon as a certain number of ants prefer a certain candidate nest site, the scouts will switch from tandem-running to a transportation behaviour known as *social-carrying*. In social-carrying, a scout picks up a passive ant or brood and carries it to the new nest site. Social carrying is three times faster than tandem-running, but has the disadvantage that a carried ant does not learn the route between the old and the new nest site. The switch from tandem-running to social-carrying can be seen as the start of the actual emigration process towards the new nest.

The number of scouts that are required for the behaviour to switch to social-carrying (and thus for making the decision for the nest site) is called *quorum threshold*. By adapting the quorum threshold to the colony's needs, *T. albipennis* is able to make a compromise between the accuracy and speed of its decisions. A low quorum threshold leads to fast but error-prone decisions. This might be acceptable when a fast emigration is required and the quality of the new nest is of minor importance, for example, when the current nest has been destroyed by a predator. On the other hand, if the ants have more time for emigration (e.g. when the current nest is too small to support the colony), the ants can take enough time to select a superior site by requiring a high quorum threshold. As the behaviour of the ants makes them capable of decentralized and self-organized decisions, we think this strategy has potential to be applied successfully in the technical domain.

3 Model of the Organic Computing System

The model for an organic computing system that is used for our study is described in this section. The system consists of a set of computing units. These units are connected by a simple interconnection network that only allows one-to-one communication; broadcast operations are not possible. Units can communicate with any other single unit at any time. As no complicated communication is employed, we do not associate any costs with communication. The purpose of the system is to maximize the total throughput of jobs. Throughput is measured as the number of jobs processed by the system per time step. In the current model it is assumed that there are always j different job types in the system. Additionally, it is assumed that the system is saturated with jobs, meaning that jobs have to be executed at every time step and the units are never idle. All units work in parallel and each unit can only work on a single job at one time step.

Each computing unit consists of s slices that can be reconfigured independently (see also [16]). Jobs are executed by the unit using computational logic configured in these slices. The more slices are configured for a certain type of job, the bigger the computational logic can be. The degree of specialization of a unit for a certain type of job depends on the number of slices that are configured for this job type. Hence, the set of slices of a unit is partitioned with respect to the different types of jobs. This partition is called the configuration of the unit

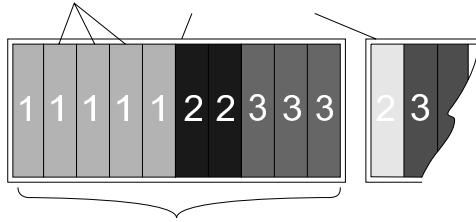


Fig. 1. Example partition of a unit's configuration resources in 10 slices. Each slice is configured to one of three possible job types. Numbers denote the job type the respective slice is configured to.

(see Figure 1 for an example). A configuration can be described as a vector $c(i)$ with $\sum c(i) = s$ and $1 \leq i \leq j$. Each unit has to be able to service every type of job, i.e. at least one slice has to be configured for every type of job. Hence, $1 \leq c(i) \leq m$ with $m = s - j + 1$ is the maximum number of slices available for a single job. As the purpose of the system is to maximize the throughput of jobs, configurations which result in higher throughput are considered to be superior to others.

It is assumed that the computing system consists of two types of units: units that are only able to execute jobs, and units that can execute jobs and are additionally able to reconfigure themselves or other units. The former units are referred to as *worker units*, whereas the latter units are called *scout units*. The computing system consists of n_w worker units and n_s scout units. Let $n = n_w + n_s$. The ratio between scouts and workers is assumed to be fixed. A worker unit can only change configuration with the help of a scout unit. One motivation for this is that a reconfigurable system needs to know, or must be able to compute reconfiguration data that is required to define the new configuration. In our case, only the scouts are able to compute this data. It is the task of the scout units to find and evaluate new configurations. In order to reconfigure another unit, the corresponding scout unit requires knowledge of the reconfiguration data for the new configuration. When this knowledge is additionally transmitted during a scout unit's reconfiguration, the newly reconfigured scout unit is able to reconfigure other units to the corresponding configuration as well. In this case, reconfiguration takes r_t time steps and is called a full reconfiguration. When a reconfigured scout did not receive the knowledge, it is not able to reconfigure other units to its new configuration. The cost r_c for such a reconfiguration (i.e., without transfer of the special knowledge) is lower than for a full reconfiguration ($r_c < r_t$). Worker units can only be reconfigured using the second reconfiguration type.

Each job has a certain run time that depends on the number of slices a unit has configured for it. Typically, one would assume that more slices result in shorter execution time. In reality, processing speed does not increase linearly with processing resources (e.g. because of caches or verification mechanisms). Hence, we cannot necessarily assume that more slices result in shorter execution time. The function which defines the run time for each job type and each possible

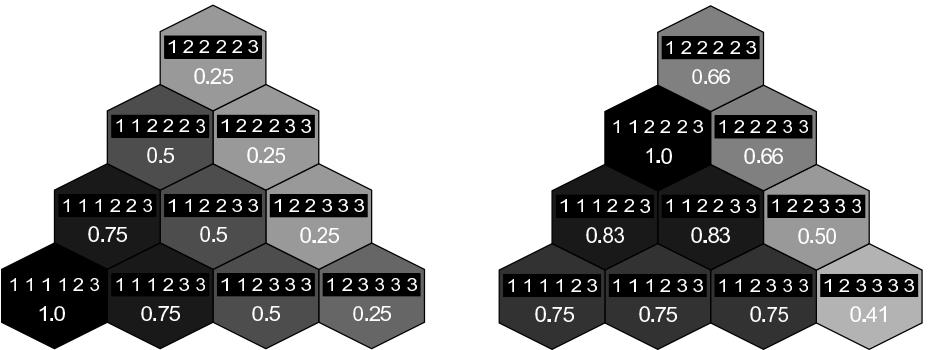


Fig. 2. Example configuration spaces with $s = 6$ and $j = 3$. Each cell denotes a configuration with configured slices per job type (in boxes) and its relative quality (below boxes). Colors illustrate the relative quality value. (a) The underlying run time function is linear, thus the configurations are ordered by their quality. In the displayed case the distribution of job type 1 is 100%, which makes the configuration in the lower left corner the optimum. (b) Non-linear (exponential) run time function. Job type 1 cannot utilize more than two slices, whereas job type 2 can. Thus, the optimal configuration gives more slices to job type 2 than the job distribution would indicate.

slice partition is referred to as the run time function. It is assumed that neither the global job distribution nor the underlying runtime function is known to the units. Hence, the units have to sample from the population of jobs.

The *quality* of a configuration o_c is defined as the average number of jobs which can be serviced per time step. Hence, $o_{\max} = 1$. The quality depends on the distribution of the different types of jobs that are in the system. Because the job types and their distribution can change in a dynamic environment, it is possible that the quality of a configuration changes over time. The ratio of the actual quality of a configuration to the quality of the optimal reconfiguration $o'_c = \frac{o_c}{o_{\max}}$ is referred to as *relative quality*.

The set of all possible configurations is called configuration space R_c . The size of the configuration space is $\binom{s-1}{j-1}$ and depends on the number of slices and number of job types. In the worst case, the size of the configuration space increases factorially with linear increasing parameters (see Figure 2 for two example configuration spaces). The configuration space is strongly characterized by the run times of the underlying job types. Job types which have a linear run time function result in spaces where configurations are ordered linearly by their quality. Hence, this type of configuration space is referred to as a linear configuration space. In order to study the impact of different underlying run time functions on the system, configuration spaces were classified by the behaviour of

Table 1. Classification of underlying run time functions

Name	Behaviour with increasing number of slices	Example
constant	constant	
linear	linearly decreasing	
monotone	monotonically decreasing	
polygonal	decreasing with local maxima	
exponential	exponentially decreasing	
increasing	increasing	
random	all functions not classified otherwise	

their run time functions.¹ All classes of run time functions that have been used in this paper are given in Table 1.

4 Search and Reconfiguration Strategies

The units of the computing system need to adapt constantly their configuration in order to deliver good performance in a dynamic environment. In order to accomplish this, the scout units need to search the configuration space and decide which configuration should actually be used. Several search and reconfiguration strategies have been developed which are described in the following.

4.1 Reference Model

As a point of reference, we developed a model that tries to solve the reconfiguration problem employing classical methods. It is therefore called the *reference model*. In this model, each scout unit has a fixed set of worker units assigned to it. A scout unit starts to search for a new configuration when it detects a change in the current job distribution; when no change is detected scout units act as workers and process jobs. New configurations are found by employing a simple heuristic. First, the scout unit generates a partition which reflects the actual job type distribution in the system. For example, if 80% of the jobs in the system are of type 1 and 20% of type 2, the heuristic will assign 80% of the computation resources to job type 1 and the rest to job type 2. After assessing the generated configuration by trial, the scout unit tries other configurations by making a random walk in the neighbourhood of this configuration (two configuration are neighboured when the assignment of the slices differs only for one slice), assessing a total of 4 other configurations. Thereafter, the scout unit starts to reconfigure the worker units that are assigned to it to the best configuration it has found. In order to make the model comparative it uses the same scout-worker ratio as the other models.

¹ Formal details on the generation of a run time model by using these classes are given in the supplementary online material.

4.2 Ant-Inspired Models

Two models have been developed which utilize the emigration strategy of *T. albipennis* for solving the reconfiguration problem. In one of the models the system adapts to the environment by making a compromise between accuracy and speed of the decision-making process, whereas the other remains static. As both models share the basic principle, we first describe the common characteristics followed by the model-specific behaviour.²

Analogously to the case of the real ants, the decision-making process and the following reconfiguration of the units are referred to as *emigration*. Scout units start to search for a better configuration with the probability p_s per time step. In this study, new configurations are tried by generating a random partition c_n (if specific characteristics of the run time functions are known, specialized strategies might be of advantage). The scout unit then assesses the new configuration for 100 time steps by executing tasks. The perceived quality o_n is then compared with the quality of the currently preferred configuration, o_p . The preferred configuration c_p might be the old configuration if the scout unit just started its search, or another configuration which was assessed by the scout unit before. If the new configuration is considered to be better than the current preferred configuration, the new configuration becomes the preferred configuration. If not, the scout unit returns to its previous state. Upon finding a preferable configuration, the scout unit starts recruiting other scout units with a probability $p_r = \frac{o_p}{o_{\max}}$ per time step, which is proportional to the perceived quality of the configuration. If the quorum threshold has not been reached for this configuration (i.e., not enough scout units prefer this configuration), the scout unit tries to gather more opinions on its preferred configuration by reconfiguring other scout units to it. Analogously to the case of the real ants, this reconfiguration is referred to as *tandem-running*. As soon as the required quorum has been reached, the scout units start to reconfigure the remaining units to the new configuration. This is called *social-carrying*, because, as with the real ants, the scout units do not transmit the knowledge required for reconfiguring other units to the new configuration. In general, units are contacted randomly, regardless of their state.

When scout units are not participating actively in the emigration, they act as worker units and service jobs in order to increase the cluster's performance. When a scout unit tries to recruit another scout unit via tandem-running, it contacts a random scout unit of its colony. If the contacted scout unit is in working state, the scout stops working with the probability p_a per time step and joins the tandem-run. If the contacted scout unit is already searching for another configuration, it switches preference with the probability p_p per time step. Scout units switch back to working state as soon as an emigration has been completed. As scout units are only asked to leave working state when there is an active emigration and possible better configuration, the number of scout units taking an active part in the emigration varies with the cluster's needs. In order to detect changes in the current configuration's quality, scout units reevaluate it periodically.

² A finite state diagram of the scout units' behaviour is available in the supplementary online material.

The following subsections describe variants in the handling of the quorum threshold.

Fixed Quorum Threshold Model. The first model is the so-called *fixed quorum threshold model* (fixed model). In this model, the computing system is not able to adapt its quorum threshold to the environment. This means that there is always the same number of scout units required to make a decision regardless of the situation. Therefore, the decision speed and accuracy is fixed and does not change. This model is a simplification of the behaviour of the real ants.

Adaptive Quorum Threshold Model. In the second model the computing system is able to adapt its quorum threshold to the environment. As for the real ants, the units are able to sense the current environment and adjust the compromise between speed and accuracy of the decision-making process accordingly. Thus, the model is referred to as the *adaptive quorum threshold model* (adaptive model). The adaptation is accomplished by scaling the quorum threshold with the relative throughput of the system. It is assumed that a system yielding a low performance should reconfigure as fast as possible. Hence, a low system throughput results in a low quorum threshold. On the other hand, when the system is delivering a high throughput, it uses a high quorum threshold. Two limits define the throughput values between which the quorum threshold is scaled linearly. Below the lower limit l_l the minimal threshold $T_{\min} = 1$ is used, whereas above the upper limit l_u the maximal quorum threshold $T_{\max} = n_s$ applies. The knowledge of the overall and optimal throughput of the cluster is required for the scaling mechanism. In this study it was assumed that the units possess this knowledge. Experiments have also been made concerning the distribution of this knowledge via gossiping protocols, but this cannot be described within the limited space of this paper.

5 Experiments

Experiments have been conducted in a standard environment with the following characteristics. In order to simulate a dynamic environment, one of the job types was replaced every 50000 time steps. Thus, always j type of jobs where present in the system. Jobs were generated randomly with a uniform distribution. The distribution of job types in the system was changed every 5000 time steps. The configuration space was generated using a polygonal run time function (see listing of run time functions in Table 1 for reference). The standard metric used in the experiments is a *fitness* metric which is defined as the ratio of average system throughput to optimal system throughput $f = \frac{d_a}{d_o}$. An optimal system exhibits a fitness of $f = 1$. The model was simulated with the Repast Agent Simulation Toolkit³.

Each parameter set was evaluated by 20 simulation runs. If not stated otherwise, the following parameter values were used. The total number of units in

³ <http://repast.sourceforge.net/>

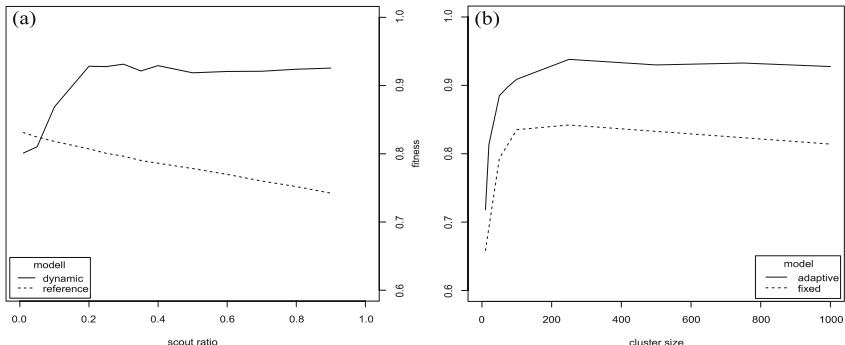


Fig. 3. (a) Fitness of the adaptive model and the reference model for scout ratios from 0.1 to 0.9, increased in steps of 0.05. (b) Fitness of the adaptive and the fixed model for increasing cluster sizes ($n \in \{10, 50, 100, 250, 1000\}$).

the system was $n = 250$, with a scout ratio of $\frac{n_s}{n} = 0.3$, $s = 10$ slices and $j = 3$ job types. Default probabilities were $p_s = 0.005$, $p_a = 0.001$ and $p_p = 0.7$. The quorum threshold for the fixed threshold model $T_{\text{fix}} = 0.3 \cdot n_s$, which proved to be optimal when used in conjunction with the default parameter settings.⁴ The quorum threshold scaling limits of the adaptive models l_l and l_u were set to 0.5 and 1.0 respectively. For the reconfiguration times, it was assumed that a scout reconfiguration takes three times as long as the reconfiguration of a worker unit ($r_t = 150$, $r_c = 50$). This is similar to the time difference found between tandem-running and social-carrying in colonies of *T. albipennis*.

5.1 Results and Discussion

One of the most characteristic features of the model is the distinction between scout and worker units. Therefore, we first studied the impact of the scout-worker ratio on the system fitness. Figure 3(a) shows the fitness of the system for different ratios. As it can be seen clearly, the adaptive model performs badly when used with a low number of scouts. If less than 20% of the units act as scouts, the fitness drops rapidly. On the other hand, with more than 20% of scouts the fitness remains nearly constant. This can be explained with the working-state mechanism: only those scouts that are actually required take part in an emigration, the other units continue to act as workers. Thus, increasing the percentage of scout units beyond the cost-optimal value of 20% does not increase the fitness of the system. This behaviour is analogous to the real ants, where the percentage of scouts has been found to be 20-30% [14]. The lower fitness in the reference model can be explained by the increased reconfiguration overhead when using more scouts.

The influence of the cluster size has been studied as well. The results are shown in Figure 3(b). In this experiment, the fixed model shows the same general

⁴ See the supplementary online material for an experimental validation of this choice.

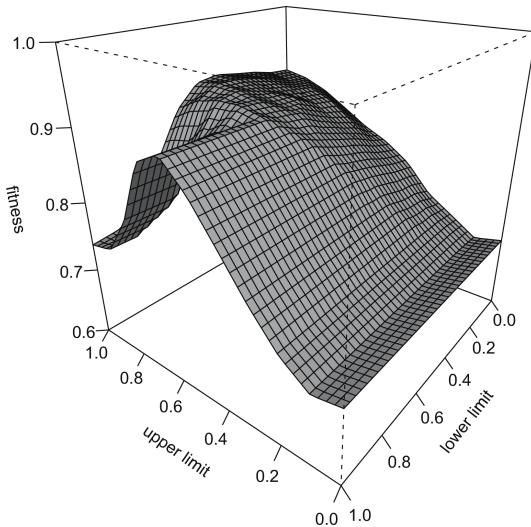


Fig. 4. Fitness of the adaptive model for different quorum threshold scaling limits l_l and l_u . Limits from 0.0 to 1.0 increased by 0.1.

behaviour as the adaptive model but exhibits a much lower fitness. This loss in fitness is attributed to the fixed quorum threshold, which enforces the same percentage of units on each decision, regardless of the situation. The decrease of fitness when using less than 100 units can be explained with the aforementioned minimal number of scout units required for the emigration to work. Apart from this minor restriction, the models scale well with an increasing number of units. As the fixed model is less effective than the adaptive model, we omitted it in the description of the following experiments.

In order to study the impact of the quorum threshold scaling on the adaptive model, we varied the scaling limits l_l and l_u as shown in Figure 4. As the results show, threshold scaling strongly affects the fitness of the system. An upper limit of less than 60% of the total number of units yields too many bad decisions even in situations where this is not acceptable, thus reducing the fitness substantially. The system does not react as sensitively to changes of the lower limit as it does to changes of the upper limit, although the results show that a lower limit over 0.5 results in a quorum threshold that requires too many units to decide unanimously. Having a lower limit higher than the upper limit results in a dysfunctional quorum scaling. Hence, a strong decrease of fitness can be observed in left corner of Figure 4.

The final experiment studies the impact of the different configuration spaces on the fitness of the various models. Figure 5 shows a comparison of the fitness on configuration spaces that are generated by the aforementioned different classes of run time functions (see Table 1). The ant-inspired models perform nearly constantly on all run time function classes, with the previously observed advantage of the adaptive model. The reference model's fitness drops with an

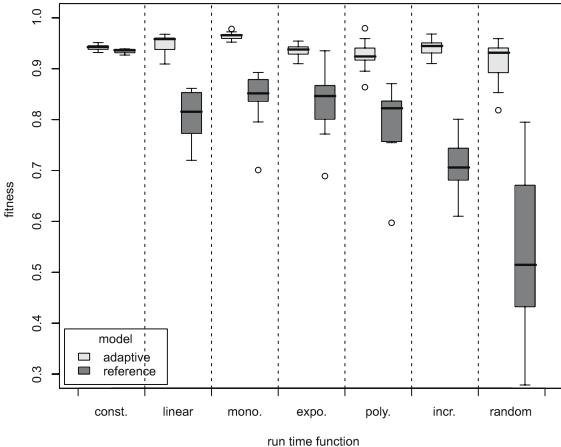


Fig. 5. Boxplot of the fitness on different configuration spaces. Whiskers indicate standard deviation and circles outliers (observations above/below $1.5 \cdot \text{IQR}$).

increasing complexity of the configuration space, as the scout units are not able to find good configurations. This shows an important advantage of the ant-inspired model: The robustness of the self-organized reconfiguration enables it to deliver nearly constant performance even in dynamic and complex configuration spaces.

6 Conclusion

In this paper, we proposed ant-inspired strategies for self-organized reconfiguration in a computing system with reconfigurable units. In particular, we used principles that are also used by the ant *Temnothorax albipennis* for house-hunting to design the proposed strategies. In our model, the system consists of scout units and worker units which execute different jobs coming into the system. The units are able to specialize on different types of jobs by adjusting their reconfigurable hardware, thereby optimizing the system's throughput. The ant-inspired strategies for making a decision on which configuration should be employed have been analyzed experimentally and compared to a non-adaptive reference strategy. The ant-inspired adaptive strategy proved to be versatile and very robust on all tested environments. In contrast to the reference strategy, the ant-inspired strategies allowed the scout units to find good configurations even in complex configuration spaces. They were able to reach a collective, decentralized decision on which configuration was acceptable in the given situation, and to reconfigure all of the cluster's units to it. A interesting similarity to the natural system has been identified, as the optimal percentage of scouts in the system is about the same as observed for the real ants.

Acknowledgements. This work was supported by the German Research Foundation (DFG) through the project “Organisation and Control of Self-Organising Systems in Technical Compounds” within SPP 1183.

References

1. IBM Research: Autonomic computing: IBM's perspective on the state of information technology. Technical report, IBM Research (2001)
2. Branke, J., Mnif, M., Müller-Schloer, C., Prothmann, H., Richter, U., Rochner, F., Schmeck, H.: Organic computing – addressing complexity by controlled self-organization. In: ISO LA 2006 (2006)
3. Würtz, R.P. (ed.): *Organic Computing*. Springer, Heidelberg (2008)
4. Compton, K., Hauck, S.: Configurable computing: A survey of systems and software. Technical report, Northwestern University, Department of ECE (1999)
5. Couzin, I.D., Krause, J., Franks, N.R., Levin, S.A.: Effective leadership and decisionmaking in animal groups on the move. *Nature* 433, 513–516 (2005)
6. Conradt, L., Roper, T.J.: Group decision-making in animals. *Nature* 421, 155–158 (2005)
7. Cicirello, V.A., Smith, S.F.: Wasp-like agents for distributed factory coordination. *Autonom. Agents and Multi-Agent Sys.* 8(3), 237–266 (2004)
8. Merkle, D., Middendorf, M., Scheidler, A.: Using decentralized clustering for task allocation in networks with reconfigurable helper units. In: de Meer, H., Sterbenz, J.P.G. (eds.) *IWSOS 2006. LNCS*, vol. 4124, pp. 137–147. Springer, Heidelberg (2006)
9. Krieger, M., Billeter, J.B.: The call of duty: Selforganised task allocation in a population of up to twelve mobile robots. *Robot. Autonom. Sys.* 30, 65–84 (2000)
10. Parker, C.A.C., Zhang, H.: Collective decision making: A biologically inspired approach to making up all of your minds. In: *IEEE Int. Conf. on Robotics and Biomimetics*, pp. 250–255 (2004)
11. Marshall, J.A.R., Dornhaus, A., Franks, N.R., Kovacs, T.: Noise, cost and speed-accuracy trade-offs: Decision-making in a decentralized system. *J. Roy. Soc. Interface* 3(7), 243–254 (2006)
12. Pratt, S.C., Sumpter, D.J.T., Mallon, E.B., Franks, N.R.: An agent-based model of collective nest choice by the ant *temnothorax albipennis*. *Anim. Behav.* 70(5), 1023–1036 (2005)
13. Pratt, S.: Quorum sensing by encounter rates in the ant *temnothorax albipennis*. *Behav. Ecol.* 16, 488–496 (2005)
14. Pratt, S.C., Mallon, E.B., Sumpter, J., Franks, N.R.: Quorum sensing, recruitment, and collective decision-making during colony emigration by the ant *leptothorax albipennis*. *Behav. Ecol. Sociobiol.* 52(2), 117–127 (2002)
15. Franks, N.R., Richardson, T.: Teaching in tandem-running ants. *Nature* 439(7073), 153 (2006)
16. Merkle, D., Middendorf, M., Scheidler, A.: Self-organized task allocation for computing systems with reconfigurable components. In: *IPDPS 2006*, pp. 25–29 (2006)

Modeling Phase Transition in Self-organized Mobile Robot Flocks*

Ali Emre Turgut, Cristián Huepe, Hande Çelikkanat,
Fatih Gökçe, and Erol Şahin

KOVAN Res. Lab., Dept. of Computer Engineering
Middle East Technical University, Turkey
`{aturgut,hande,fgokce,erol}@ceng.metu.edu.tr,`
`cristian@northwestern.edu`

Abstract. We implement a self-organized flocking behavior in a group of mobile robots and analyze its transition from an aligned state to an unaligned state. We briefly describe the robot and the simulator platform together with the observed flocking dynamics. By experimenting with robotic and numerical systems, we find that an aligned-to-unaligned phase transition can be observed in both physical and simulated robots as the noise level is increased, and that this transition depends on the characteristics of the heading sensors. We extend the Vectorial Network Model to approximate the robot dynamics and show that it displays an equivalent phase transition. By computing analytically the critical noise value and numerically the steady state solutions of this model, we show that the model matches well the results obtained using detailed physics-based simulations.

1 Introduction

Flocks of birds, herds of quadrupeds and schools of fish stand as fascinating examples of self-organized coordination, where groups of individuals coherently move and maneuver in space as a collective unit [1,2]. Although it has long been studied in biology, it was Reynolds [3] who first demonstrated flocking in artificial systems, showing that realistic flocking behavior can be obtained in computer animation using a number of simple behaviors. Reynolds' seminal work generated interest in many different fields.

In robotics, Mataric [4] made one of the earliest attempts to obtain flocking in a group of robots by combining safe-wandering, aggregation, dispersion, and homing behaviors. She was able to demonstrate that a group of robots can “flock” towards a common homing direction while maintaining a cohesive grouping.

* The works of A.E. Turgut, F. Gökçe and E. Şahin are supported by TÜBİTAK under grant no: 104E066. The work of C. Huepe is supported by the National Science Foundation under Grant No. DMS-0507745. H.Çelikkanat acknowledges the partial support of the TÜBİTAK graduate student research grant. F. Gökçe is currently enrolled in Faculty Development Program (ÖYP) in Middle East Technical University on behalf of Süleyman Demirel University.

In [5], Kelly and Keating used robots that can sense the obstacles as well as the relative range and bearing of their neighbors through a custom-made active infrared system. The robots used a radio-frequency system to elect one of them as the leader which would then wander in the environment while being followed by the others. In a recent study, Hayes et al. [6] proposed a “leaderless distributed flocking algorithm” consisting of two simpler behaviors: collision avoidance and velocity matching, using local center-of-mass calculations based on emulated range and bearing information. Additional studies in robotic flocking have been carried out in works such as [7,8].

Flocking has also attracted interest in physics, where various models have been proposed to study the emergence of *order* in such systems. The emergence of order corresponds to the collective self-alignment of the group to a common heading direction as a result of the interactions among its individuals. In a pioneering study, Vicsek et al. [9] proposed the Self-Driven Particles (SDP) model to explain the emergence of order in biological swarms. The SDP model uses massless and volumeless particles that move at a constant speed in a square arena with periodic boundary conditions. The heading of each particle is updated to the average direction of motion of its local neighbors and a noise term which is added to account for uncertainties in its inputs and control. Simulations of this model revealed that particles align if the system is above a critical mean density or if the magnitude of the noise is below a critical value. In a follow-up study, Gregoire et al. [10] extended the SDP model to add attraction and repulsion among the particles in their local neighborhood, thus achieving self-organized motion in an open domain.

Aldana et al. [11] proposed the Vectorial Network Model (VNM) to study the emergence of long-range order in systems with mostly local interactions. In the VNM, particles are placed in a two-dimensional lattice and their positions remain fixed. Each heading thus only determines here a pointing direction for a given lattice site. As in the SDP, headings are updated to the average pointing directions of its inputs, but here only some of these inputs are taken from neighboring lattice sites and the rest from randomly chosen (possibly distant) sites. The VNM simulations show that long-range order does not emerge at any nonzero noise value if the neighbors are chosen only locally. However, long-range order does emerge for sufficiently small noise values if a small fraction of the inputs are chosen from random particle sites. An interesting aspect of the VNM is that an analytic expression was obtained in [11] to describe its order-to-disorder phase transition for the case with no local interactions, where all inputs are taken from random particle sites.

Despite many efforts to control and model flocks in robotics and statistical physics, these two lines of research have remained relatively disconnected from each other. An exception can be found in the recent study in [12], where a link was established between the behavior of multi-robot systems and phase transitions. There have been two main reasons behind the failure to integrate both perspectives. First, until recently, true self-organized flocking behavior like the one observed in nature had not been achieved in robotic systems. Indeed,

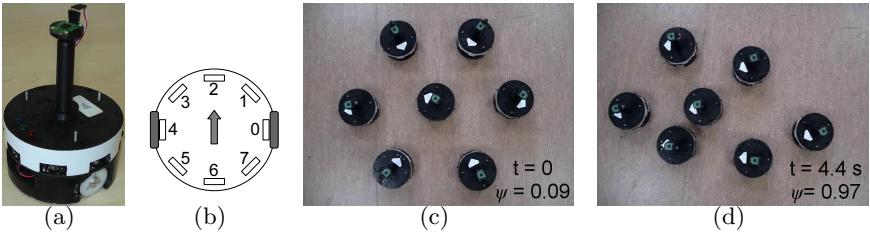


Fig. 1. (a) Photo of a Kobot. (b) Top-view of a Kobot sketch showing the body (circle), the IR sensors (small numbered rectangles), and the two wheels (grey rectangles). (c-d) Starting from a disordered state, 7 Kobots negotiate a common heading and advance. White arrows on the robots indicate the forward direction.

the previous experimental studies in robotics used either a virtual or an explicit leader [5] to guide the group or assumed that a target heading (or destination) was sensed by the whole group [4,6,13]. Moreover, in some of these studies [6], the authors used “emulated sensors”. Second, the assumptions required by the models developed in physics were often considered to be too unrealistic to be linked with studies conducted in robotics. The SDP model, for example, uses massless and volumeless mobile particles.

In [14], we reported the first true self-organized flocking on a group of mobile robots and showed that the robots can maneuver in an environment as a cohesive body while avoiding obstacles on their path. In this paper, we first demonstrate that the emergence of order in a robot flock depends on the amount of noise in the heading alignment behavior. We then extend the VNM to model the order-to-disorder phase transition that occurs in these robotic systems as the noise level is increased. Finally, we compare the predictions of the proposed model to the results obtained from robots.

2 Experimental Framework

We used a custom-built mobile robot platform, called Kobot, and its physics-based simulator, that we refer to as CoSS, in our experiments. Kobot is a CD-sized (with a 12 cm diameter), light-weight, differentially driven robotic platform (Figure 1(a)). It possesses an active Infrared Short-Range Sensing (IRSS) system designed for short-range proximity measurements. This system utilizes modulated infrared signals to minimize environmental interference and crosstalk among the robots. It consists of eight sensors placed evenly at 45° intervals (see Figure 1(b)), each of which is capable of sensing kin-robots and obstacles within a 21 cm range in seven discrete levels at 18 Hz.

The Virtual Heading Sensor (VHS) consists of a digital compass and a wireless communication module to receive the relative headings of neighboring robots. The VHS module measures its own heading with respect to the *sensed North* at each control step and broadcasts it to other robots through wireless communication. Each robot receives the broadcasted heading values within its

communication range. We define the set of robots that are “heard” by a given robot as its *VHS neighbors*. The angular difference between the broadcasted values and its own heading allows a robot to compute its relative heading with respect to its VHS neighbors and adjust it as needed. This operation of the VHS module assumes that the *sensed North* remains approximately the same among the robots within communication range.

Flocking Behavior. The flocking behavior [14] consists of a heading alignment and a proximal control combined in the weighted vector sum:

$$\mathbf{a} = \frac{1}{8} \mathbf{h} + \mathbf{p},$$

where \mathbf{h} is the heading alignment vector, \mathbf{p} is the proximal control vector, and \mathbf{a} is the desired acceleration vector. The heading alignment vector \mathbf{h} , which is used to align the robot with the average heading of its neighbors, is calculated as:

$$\mathbf{h} = \frac{\sum_{j \in \mathcal{N}} e^{i\theta_j}}{\|\sum_{j \in \mathcal{N}} e^{i\theta_j}\|}$$

where \mathcal{N} denotes the set of VHS neighbors, θ_j is the heading of the j^{th} neighbor and $\|\cdot\|$ calculates the Euclidean norm.

The proximal control behavior uses readings obtained from the IRSS to avoid collisions and to maintain cohesion between the robots. For each IR sensor, a virtual force proportional to the square of the difference between the measured distance and the desired distance is assumed. The desired distance (d_{des}) is defined as a finite value for other robots and ∞ for obstacles, in order to keep a fixed distance to its peers while moving away from obstacles. The normalized proximal control vector \mathbf{p} is therefore given by:

$$\mathbf{p} = \frac{1}{8} \sum_k f_k e^{i\phi_k}$$

where $k \in \{0, 1, \dots, 7\}$ denotes the sensor positioned at angle $\phi_k = \frac{\pi}{4}k$ with respect to the x -axis (see Figure 1(b)) and f_k is calculated as:

$$f_k = \begin{cases} -\frac{(d_k - d_{des})^2}{\frac{1}{8}} & \text{if } d_k \geq d_{des} \\ \frac{(d_k - d_{des})^2}{\frac{1}{8}} & \text{otherwise.} \end{cases}$$

The desired acceleration vector is mapped to the forward and angular velocities of the robot. The forward velocity u is modulated depending on the deviation of the desired acceleration vector from the current direction, as given by:

$$u = \begin{cases} 0.7 \left(\frac{\mathbf{a}}{\|\mathbf{a}\|} \cdot \hat{\mathbf{a}}_c \right) & \text{if } \frac{\mathbf{a}}{\|\mathbf{a}\|} \cdot \hat{\mathbf{a}}_c \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

where $\hat{\mathbf{a}}_c$ is the current direction of the robot parallel to the y -axis of the body-fixed reference frame. The angular velocity ω is controlled by a proportional controller:

$$\omega = \frac{1}{2}(\angle \hat{\mathbf{a}}_c - \angle \mathbf{a}).$$

The Order and Average Order Metrics. We evaluate the flocking performance by using the well-studied measure of order ψ [9], which corresponds to the average alignment of the group and is computed as:

$$\psi = \frac{1}{M} \left| \sum_{k=1}^M e^{i\theta_k} \right|,$$

where M is the number of robots and θ_k is the heading of the k^{th} robot. The order can take any value between 0 and 1. When individuals are aligned and the system is *ordered* we have $\psi \approx 1$, and when individuals are not aligned and the system is *disordered* we have $\psi \approx 0$. When the steady-state performance of the flocking behavior is considered, we will use the time average of the order, denoted by $\bar{\psi}$.

3 Modeling the Virtual Heading Sensor

The properties of the flocking behavior depend on two specific characteristics of the virtual heading sensor, namely: (1) the number of VHS neighbors that can be simultaneously detected, and (2) the nature and amount of noise in the digital compass measurements. Hence, we will study here the dependence of the system on these characteristics by modeling them with the CoSS simulator. The number of VHS neighbors N depends on the range of communication, the number of robots within this range, and the frequency and duration of this communication. Systematic analysis has shown that in the physical robots, the number of VHS neighbors can be as large as 20 in large groups.

The noise in VHS solely depends on the noise characteristics of the digital compass. In indoor environments, the presence of ferrous materials induce large amounts of noise on the digital compass. We model this noise by adding a vector of fixed magnitude in a random direction to each heading measurement [10,12]. The resulting noisy heading of the j^{th} neighbor received as input by a given

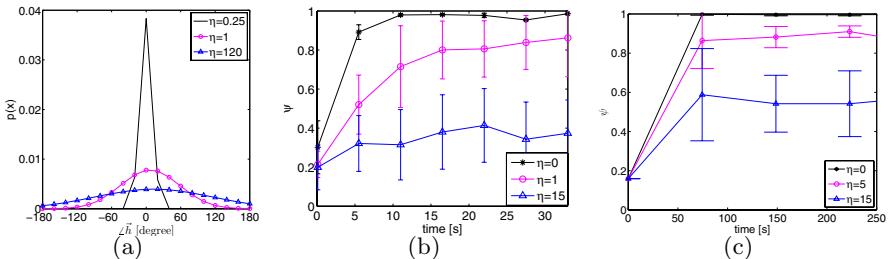


Fig. 2. (a) Probability density function of the simulated noisy measurement inputs of one VHS neighbor for different η values. (b) Evolution of ψ in time for 7 Kobots. (c) Evolution of ψ in time for 7 robots in the CoSS simulator. In this figure and subsequent figures the standard deviations are indicated using error bars.

robot is:

$$\theta_j = \angle(e^{i\theta_a} + \eta e^{i\xi})$$

where θ_a is the actual heading of the j^{th} neighbor, η is the magnitude of the noise vector, and $\angle(\cdot)$ is function that calculates the argument of a vector. The noise-vector direction ξ is a delta-correlated random variable with a uniform distribution in $[-\pi, \pi]$.

The probability density functions of the simulated noisy measurements is plotted in Figure 2(a) for various values of η . It is apparent that η determines the standard deviation of the resultant noisy headings, as expected.

4 Analysis of the Flocking Behavior

We investigate the effect of the sensing noise on the transient and on the steady-state to characterize the flocking phase transition in our system. The transient characteristics were investigated by conducting experiments with 7 Kobots and simulating 7 robots in CoSS for 1 VHS neighbor. The steady-state characteristics were investigated by simulating 100 robots in CoSS. The robots were initially placed in a regular hexagonal formation with a center-to-center distance of 25 cm having random orientations. Each experiment was repeated 10 times.

Transient Analysis. We varied the sensing noise on the VHS and measure the evolution of ψ in time for 7 Kobots in an actual experiment and for 7 robots in the CoSS simulator. In the Kobot experiments, the environmental noise is assumed to be negligible, so we set it to correspond to the $\eta = 0$ case. Higher η values are attained by adding noise artificially to each heading measurement. The evolution in time of ψ for the experimental and simulation cases are plotted on Figures 2(b) and 2(c). The results indicate that that ψ increases from a random initial condition, approaching approximately 1 when η is low, while settling to smaller ψ values when η is increased. Although the trends in the Kobot and CoSS cases are the same, Kobots perform worse than the robots in the CoSS simulation for the same amount of noise. This is probably due to the environmental noise in the Kobot experiments, that adds an unpredictable base noise level to the system.

Steady-State Analysis. We will now analyze the steady-state flocking behavior by finding the value of ψ at which the system settles for various noise levels. We consider the results of two sets of CoSS simulations: (1) runs with a small and a large number of robots (7 and 100, respectively), but with only 1 VHS neighbor, (2) runs with 100 robots and different numbers of VHS neighbors. Each simulation was conducted for 1000 s and its steady-state ψ value was calculated by averaging over the last 500 s. This guarantees that the steady-state condition has been achieved since it takes less than 250 s for ψ to converge. The results are plotted on Figures 3(a) and 3(b).

Figure 3(a) shows that, regardless of the flock size, an increase in η decreases the order. However, the order shows a much smaller decrease for the small group

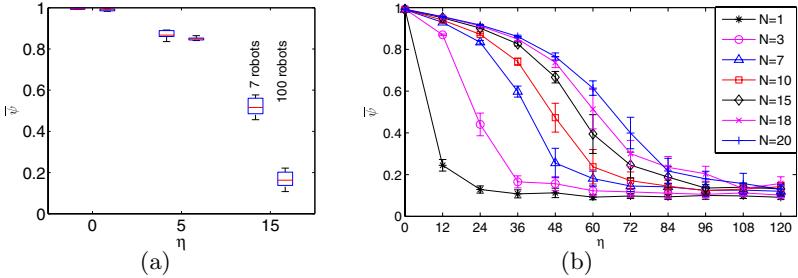


Fig. 3. (a) Plot of $\bar{\psi}$ measured in CoSS simulations containing 7 and 100 robots. The horizontal lines in the boxes, the box-ends, and the additional error bars correspond to the median and to its first and third quartiles, respectively. (b) Plot of $\bar{\psi}$ for 100 robots in CoSS for various N .

when $\eta = 15$. This is due to the finite size effects observed that are known to blur the difference between order and disordered collective states in small-sized statistical systems. In the rest, we will therefore focus on CoSS simulations with 100 robots, which enables reasonable run times while minimizing finite-size effects. It is apparent on Figure 3(b) that for all N values, ψ approaches 1 and the system organizes in a coherent flock as the noise level η is lowered to zero. If η is increased, ψ decreases and eventually approaches to 0. We also observe that increasing N increases the order of the flock for a constant η . The order-to-disorder transition that occurs by increasing the noise level in this robotic system is equivalent to the second-order phase-transition observed in various statistical physics systems. As in the physics context, we will refer to the η value at which the system loses all order as the critical noise level η_c .

5 Modeling the Phase Transition in Flocking

We are interested in modeling analytically the order-to-disorder phase transition observed in the flocking behavior of Kobots as a function of noise, and to determine η_c for a given number of VHS neighbors. We will consider a simple network-based model that includes the noise, interactions with randomly chosen agents, and an inertia-like term that captures qualitatively various local interactions that affect the Kobot dynamics. Our model does not, however, deal with all the complexities of the behavior. Indeed, in order to allow analytical solutions, it avoids any spatial description by representing agents as nodes interacting through random switching connections in a network. This model is an extension of the Vectorial Network Model (VNM) introduced in [11] as a network version of the SDP model in [9].

The original VNM can be solved analytically and is known to display an order-to-disorder transition similar to the one observed in Kobots, but it is not well adapted to describe some of the details of our robotic system. For example, the noise is introduced differently and the agents can turn in any direction at

every time-step, with no restrictions on the turning rate. We therefore introduce a new network system to model the Kobots, the Stiff-Vectorial Network Model (S-VNM), which we define as follows. At every time-step, each node updates its heading $h_j(t)$ based on N inputs chosen randomly from any node in the system, according to

$$\mathbf{h}_j(t+1) = \kappa e^{i\theta_j(t)} + \lambda \sum_{k=1}^N e^{i\theta_k(t)} + \eta \sum_{k=1}^N e^{i\xi_k(t)}. \quad (1)$$

Here, the heading h_j is a vector of arbitrary magnitude with $\angle[h_j] = \theta_j$, where the heading of Kobot j is given (in this network representation) by the angle θ_j associated to node j . The model parameters κ , λ , and η determine the relative importance of the persistence, interaction, and noise terms, respectively. The latter two correspond directly to the terms implemented in the VHS control part of the Kobot dynamics. Here, the noise is introduced as in [10], with a fixed magnitude η in a random direction given by ξ_k , a delta-correlated random variable uniformly distributed in $[-\pi, \pi]$. The persistence term models qualitatively an effective inertia that appears mainly due to the proximity interactions between Kobots. These make it harder for a given robot to turn in response to its VHS inputs or the noise since, if it is surrounded by other robots heading in the same direction, these will block it from shifting its heading.

Analytical Treatment of the S-VNM. One of the main appeals of the S-VNM lies in our ability to treat it analytically. We derive here a solution that allows us to compute the critical noise value η_c (at which the order-to-disorder transition occurs) in terms of κ , λ , and N . We will approach this problem as follows. First, we compute the probability density function (PDF) of each term in Equation (1). Then we impose that the PDF of θ_j (which is equal for all j) is the same at time t and at time $t+1$, which, together with Equation (1), provides us with a closed expression for the statistical steady state of this PDF. Finally, we find the value of η at which a constant distribution for θ_j becomes unstable. This corresponds to the critical noise level η_c above which there is a stationary distribution with h_j pointing in any direction with the same probability (the disordered state), and below which such solution is unstable, thus drifting the system to a distribution with a preferred direction for h_j (the ordered state). In what follows we will use the fact that the magnitude of h does not intervene in the dynamics of the S-VNM to rescale Equation (1) by dividing it by λ . We thus define $\tilde{\kappa} = \kappa/\lambda$ and $\tilde{\eta} = \eta/\lambda$ and use these rescaled variables below, dropping the tildes until the end of this calculation to simplify the notation.

We first consider the noise term, which can be viewed as the total displacement that results after taking N steps of length 1 in a two-dimensional random walk. Using this analogy, we can apply well-known random walk results and write the PDF of the position on the x y plane after N steps as

$$P_\eta(x, y) = \frac{1}{N\eta^2\pi} e^{-\frac{x^2+y^2}{N\eta^2}}. \quad (2)$$

We will need below the two-dimensional Fourier transform of Eq. 2, that can be readily computed to obtain

$$\hat{P}_\eta(\lambda_x, \lambda_y) = e^{-\frac{1}{4}N\eta^2(\lambda_x^2 + \lambda_y^2)}. \quad (3)$$

We now carry out the calculation of the PDF of the interaction term, which can be quite involved. Fortunately, this result was already obtained in [11]. Assuming that all the θ_k have the same PDF and are statistically independent (which is true if the N inputs are all picked at random from any node in a large system), the Central Limit Theorem is used in [11] for large enough N values to find an approximate Gaussian expression for this PDF. Computing again its Fourier transform we obtain:

$$\hat{P}_{N\theta}(\lambda_x, \lambda_y) = e^{iN(\Delta_{1,0}\lambda_x + \Delta_{0,1}\lambda_y) - \frac{N}{2}(\sigma_c^2\lambda_x^2 + \sigma_s^2\lambda_y^2 + 2\sigma_{cs}^2\lambda_x\lambda_y)} \quad (4)$$

where $\sigma_c^2(t) = \Delta_{2,0}(t) - [\Delta_{1,0}(t)]^2$, $\sigma_s^2(t) = \Delta_{0,2}(t) - [\Delta_{0,1}(t)]^2$, and $\sigma_{cs}^2(t) = \Delta_{1,1}(t) - \Delta_{1,0}\Delta_{0,1}$. Here, $\Delta_{m,n}(t)$ denotes the instantaneous cosine-sine moment of the angle distribution, given by

$$\Delta_{m,n}(t) = \int_{-\pi}^{\pi} P_\theta(\alpha; t) \cos^m(\alpha) \sin^n(\alpha) d\alpha,$$

in which $P_\theta(\alpha; t)$ is the PDF of the angle that describes the heading of each particle. Note that Eq. (4) converges very rapidly to the exact PDF of the interaction term as N is increased, providing a very good approximation for $N > 5$.

The derivations above furnish expressions for the PDF of every element of equation (1) in terms of the PDF of the direction of a single particle $P_\theta(\alpha; t)$. However, as they stand these expressions are far too complicated to find the functional form of $P_\theta(\alpha; t)$ as a stationary solution of Eq. (1). To continue the calculations, we will thus concentrate in solutions close to η_c . If $\eta > \eta_c$, the system is in the disordered regime and we know that all h_j must point in any direction with the same probability. This corresponds to having a uniform distribution $P_\theta(\alpha) = \frac{1}{2\pi}$. If $\eta < \eta_c$ we know that this PDF cannot be a stable stationary solution of the dynamics since the system becomes organized and the symmetry in the pointing directions of all h_j must be broken. Therefore, a small perturbation about the $P_\theta(\alpha)$ solution must grow in this regime. Without loss of generality, we can write a generic small perturbation of $P_\theta(\alpha)$ as:

$$P_\theta(\alpha) = \frac{1}{2\pi} + \delta \cos(\alpha), \quad (5)$$

where $\delta \ll 1$. Using this form for $P_\theta(\alpha; t)$, Eq. (4) becomes:

$$\hat{P}_{N\theta}(\lambda_x, \lambda_y) = e^{i\pi N \delta \lambda_x - \frac{1}{2}N \left[\left(\frac{1}{2} - \pi^2 \delta^2 \right) \lambda_x^2 + \frac{\lambda_y^2}{2} \right]}. \quad (6)$$

We now can find the combined PDF of the interaction and noise terms by computing the inverse Fourier transform of the product of $\hat{P}_\eta(\lambda_x, \lambda_y)$ times $\hat{P}_{N\theta}(\lambda_x, \lambda_y)$ to obtain

$$P_{N\theta\eta}(x, y) = \frac{1}{\pi N \sqrt{(1+\eta^2)(1+\eta^2 - 2\pi^2\delta^2)}} e^{-\frac{1}{N} \left[\frac{(x-N\pi\delta)^2}{1+\eta^2-2\pi^2\delta^2} + \frac{y^2}{1+\eta^2} \right]}. \quad (7)$$

From this result we find the PDF of the Right Hand Side (RHS) of equation (1) by calculating the convolution of $P_{N\theta\eta}$ with $P_\theta(\alpha)$ (the PDF of the persistence term). The resulting expression is:

$$P_{RHS}(x, y) = \int_{-\pi}^{\pi} P_{N\theta\eta}(x - \kappa \cos \theta, y - \kappa \sin \theta) P_\theta(\theta) d\theta. \quad (8)$$

Expanding Eq. 8 to first order in the small δ perturbation, then expressing the resulting equation in polar coordinates (R, Φ) and integrating over R we finally obtain the PDF of the RHS

$$P_{RHS}(\Phi) = \frac{1}{2\pi} + \delta \Gamma \cos(\Phi), \quad (9)$$

where

$$\Gamma = \frac{\sqrt{\pi}}{2\sqrt{N(1+\eta^2)}} e^{\frac{-\kappa^2}{2N(1+\eta^2)}} \left[(N + \kappa) I_0 \left(\frac{\kappa^2}{2N(1+\eta^2)} \right) + \kappa I_1 \left(\frac{\kappa^2}{2N(1+\eta^2)} \right) \right]. \quad (10)$$

Here $I_n(\cdot)$ are the Modified Bessel Functions of the first kind, usually defined mathematically as the solutions to the differential equation: $z^2 y'' + zy' - (z^2 + n^2)y = 0$.

In the stationary case, the LHS of Equation 1 is equal to $P_\theta(\phi)$. Hence, the LHS and the RHS have the same form and the condition for a statistically stationary solution $P_\theta(\Phi) = P_{RHS}(\Phi)$ becomes

$$\frac{1}{2\pi} + \delta \cos(\Phi) = \frac{1}{2\pi} + \lambda \Gamma \cos(\Phi). \quad (11)$$

The condition for $P_\theta(\alpha; t) = 1/(2\pi)$ to be a stable stationary solution of the probability distribution associated to the dynamics described in Eq. (1) is therefore given by $\Gamma < 1$. Setting Γ to 1, we thus find the critical noise level η_c at which the order-to-disorder transition occurs.

While $\Gamma = 1$ fully determines η_c implicitly in terms of the model parameters, this condition cannot be easily inverted to obtain an expression for η_c . We can find an explicit approximate form for η_c , however, by carrying out certain approximations in the regime that we are considering. We set κ and λ to 1.5 and 22, respectively, to capture the dynamics of the flocking behavior. For these coefficients, $\frac{\kappa^2}{2N(1+\eta^2)} \ll 1$, which makes $I_0() \sim 1$ and $I_1() \sim 0$. In the resulting expressions, the terms containing κ are small when compared to those containing other coefficients and, thus, they can be neglected. By then substituting $\eta = \tilde{\eta}\lambda$ we obtain the following simple approximate expression for η_c

$$\eta_c = \lambda \sqrt{\frac{N\pi}{4}}. \quad (12)$$

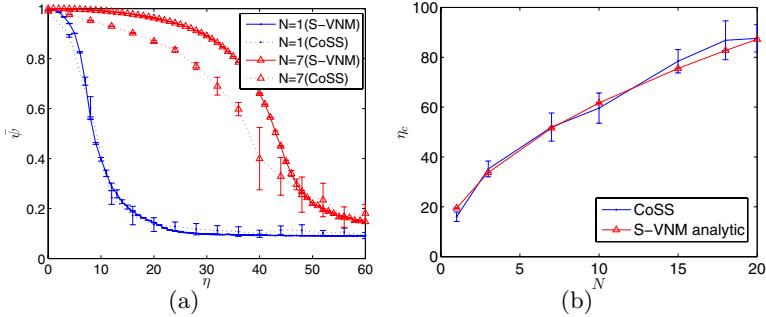


Fig. 4. (a) Phase transition diagram obtained numerically using S-VNM and CoSS simulations. (b) Critical noise values obtained using the analytical solution of S-VNM and CoSS simulations.

Results Using the S-VNM. The S-VNM can be utilized in two ways to predict the phase-transition in flocking. On one side, the S-VNM can be easily and efficiently implemented numerically to obtain the full phase-transition diagram for the stationary flocking solutions resulting from a given set of parameters. On the other side, we can use the analytical solution found above for the S-VNM to predict η_c as a function of N .

The steady-state response was investigated by simulating the S-VNM numerically. The simulations were performed with 100 particles for 10000 time-steps using $N = 1$ or $N = 7$. $\bar{\psi}$ for the last 5000 steps is plotted in Figure 4(a) as a function of η . On the same plot, $\bar{\psi}$ of analogous CoSS simulations is also displayed. It is apparent that predictions of the S-VNM are in close agreement with the CoSS results for $N = 1$. A slight deviation is observed in the $N = 7$ case as the system becomes organized.

Figure 4(b) displays the predicted critical noise value for the flocking behavior obtained using Eq. 12 together with results from CoSS simulations as a function of N . The two results are in close agreement both for small and large N values. However, we should note that Eq. 12 is actually only meant to be valid for relatively large values of N (typically at least $N > 5$) due to the use of the Central Limit Theorem in the analytical treatment.

6 Conclusion

In this paper we studied self-organized flocking in a group of mobile robots. We consider this work as a first step towards linking the mathematical models of flocking proposed in statistical physics with the results obtained in robotic systems. In this particular study, we showed the existence of an order-to-disorder phase transition in flocking and that the amount of noise as well as the number of neighbors with which each robot interacts determines the characteristics of this transition. We have extended the Vectorial Network Model to incorporate the dynamics of our robots and showed that the steady-state order characteristics

predicted by the model matches the ones obtained for the robotic system. This analysis shows that the proximal interactions among the robots can be crudely approximated by the inclusion of a stiffness term in the model.

References

1. Parrish, J.K., Viscido, S.V., Grünbaum, D.: Self-organized fish schools: An examination of emergent properties. *Biol. Bull.* 202, 296–305 (2002)
2. Buhl, J., Sumpter, D.J.T., Couzin, I., Hale, J., Despland, E., Miller, E., Simpson, S.J.: From disorder to order in marching locusts. *Science* 312, 1402–1406 (2006)
3. Reynolds, C.: Flocks, herds and schools: A distributed behavioral model. In: Proc. of SIGGRAPH 1987, pp. 25–34 (1987)
4. Mataric, M.J.: Interaction and Intelligent Behavior. PhD thesis, MIT (1994)
5. Kelly, I., Keating, D.: Flocking by the fusion of sonar and active infrared sensors on physical autonomous robots. In: Proc. of the 3rd Int. Conf. on M2VIP, vol. 1, p. 14 (1996)
6. Hayes, A., Dormiani-Tabatabaei, P.: Self-organized flocking with agent failure: Off-line optimization and demonstration with real robots. In: Proc. of ICRA 2002, pp. 3900–3905 (2002)
7. Fax, J.A., Murray, R.M.: Information flow and cooperative control of vehicle formations. *IEEE Trans. Automat. Contr.* 49(9), 1421–1603 (2004)
8. Sepulchre, R., Paley, D., Leonard, N.E.: Stabilization of planar collective motion: All-to-all communication. *IEEE Trans. Automat. Contr.* 52(5), 811–824 (2007)
9. Vicsek, T., Czirok, A., Ben-Jacob, E., Cohen, I., Shochet, O.: Novel type of phase transition in a system of self-driven particles. *Physical Review Letters* 75(6) (1995)
10. Gregoire, G., Chate, H., Tu, Y.: Moving and staying together without a leader. *Physica D* 181, 157–170 (2003)
11. Aldana, M., Huepe, C.: Phase transitions in self-driven many-particle systems and related non-equilibrium models: A network approach. *J. of Stat. Phy.* 112 (1/2), 135–153 (2003)
12. Baldassarre, G.: Self-organization as phase transition in decentralized groups of robots: A study based on boltzmann entropy. In: Mikhail, P. (ed.) *Advances in Applied Self-Organizing Systems*, pp. 127–146. Springer, Berlin (2008)
13. Campo, A., Nouyan, S., Birattari, M., Groß, R., Dorigo, M.: Negotiation of goal direction for cooperative transport. In: Dorigo, M., Gambardella, L.M., Birattari, M., Martinoli, A., Poli, R., Stützle, T. (eds.) *ANTS 2006. LNCS*, vol. 4150, pp. 191–202. Springer, Heidelberg (2006)
14. Turgut, A.E., Çelikkanat, H., Gökc  , F.,  ahin, E.: Self-organized flocking with a mobile robot swarm. In: Proc. of AAMAS 2008, pp. 39–46 (2008)

Molecular Structure Elucidation Using Ant Colony Optimization: A Preliminary Study

Caroline Farrelly, Douglas B. Kell, and Joshua Knowles

Manchester Interdisciplinary Biocentre, The University of Manchester
Manchester, UK
j.knowles@manchester.ac.uk

Abstract. Identifying the structure of unknown molecules is an important activity in the pharmaceutical industry where it underpins the production of new drugs and the analysis of complex biological samples. We present here a new method for automatically identifying the structure of an unknown molecule from its nuclear magnetic resonance (NMR) spectrum. In the technique, an ant colony optimization algorithm is used to search iteratively the highly-constrained space of feasible molecular structures, evaluating each one by reference to NMR information on known molecules stored (in a raw form) in a database. Unlike existing structure elucidation systems, ours: does not need prior training or use spectrum prediction; does not rely on expert rules; and avoids enumeration of all possible candidate structures. We describe the important elements of the system here and include results on a *preliminary* test set of molecules. Whilst the results are currently too limited to allow parameter studies or comparison to other methods, they nevertheless indicate the system is working acceptably and shows considerable promise.

1 Introduction

Analytical chemists exploit a variety of spectroscopic techniques in order to gain an insight into the structure of unknown molecules. They use the molecule's exact mass, available from mass spectrometry, to reveal the empirical formula (e.g. C₄H₈BrF), and then study the molecule's spectral fingerprint to understand something about how these atoms are arranged. With NMR spectroscopy, patterns of chemical shifts can reveal information about local structures, from which it is (theoretically) possible, often after considerable toil, to infer the global molecular form.

Computer assistance for the task of *structure elucidation* has been available for decades now, initially as a means of helping to enumerate parts of the structural space so that chemists would be sure not to overlook any of the exponentially many possible forms. More recently, various AI techniques have been employed to automate the process further (see Section 5). For the most part, these techniques work by enumerating possible structures and then predicting the spectra of each one, which is then compared to the observed spectrum of the unknown molecule. This approach requires training machine learning methods to perform spectrum

prediction, a science which is developing but still far from a solved problem. Moreover, the training process is intricate and time-consuming, and needs to be targeted to the particular kinds of molecules of interest. The quantitative comparison of observed and predicted spectra in these systems is also a nontrivial task which represents a further area under development.

In this paper, we investigate whether it may be feasible to tackle the structure elucidation problem *without* the use of spectral prediction methods. The approach we propose searches the space of possible structures iteratively using ant colony optimization [1], and evaluates candidate structures more directly by reference to a database of chemical shift patterns for known molecules. There is no explicit training necessary in our proposed method (in the sense of supervised learning), which potentially makes our system easier to update and less of a black box. From a machine learning perspective, the approach we use is similar to *lazy learning* [2]: we store our ‘prior knowledge’ in a fairly raw and uncompressed form and wait for a query before doing some work on the data to answer the query.

At the core of the system is a search of the candidate molecule space; the prior knowledge data is used mainly as an approximate evaluation function. Our motivations for choosing ant colony optimization as the search method are two-fold. First, there are many constraints involved in building the structures and a constructive method such as ACO allows straightforward building of feasible solutions. Secondly, much of the structural information in a candidate structure relates to the order with which small modules (or substructures) are put together. Thus, we can treat the problem as a pseudo-ordering problem. We know that ACO is good at ordering problems from its successes in TSP, assignment, and scheduling applications [1]. In addition, some local enumeration of molecular structures is necessary to ensure all possibilities have been exhausted; and we know that combinations of ACO and local search tend to perform well (e.g., see [3,4]).

The rest of the paper is organized as follows. Section 2 formulates the problem that we tackle in this work, and relates it to other problems in machine learning and optimization. The problem is addressed by the approach we set out in detail in Section 3. Section 4 presents results from running the proposed method on a number of real NMR spectrum-to-structure problems. We discuss related literature on small molecule structure prediction from NMR in Section 5 and in Section 6 we summarise the initial findings presented here and look ahead to further developments.

2 The Spectrum to Structure Problem

The ‘Spectrum to Structure Problem (SSP)’ asks for the chemical structure of a molecule, given the molecule’s spectral shift pattern and its empirical formula (EF). In the version of the problem we consider here, we are concerned with small organic molecules up to 500 molecular weight (MW) and the spectra are ^{13}C NMR spectra. The EF given denotes only the constituent atoms in the

molecule, not their arrangement. Different arrangements of the same constituent atoms are known as isomers; for even relatively small molecules there can be many isomeric forms, each giving rise to a slightly different NMR spectrum, e.g. the hydrocarbon C_9H_{16} has 1 902 isomeric forms. Furthermore, the number of isomers grows exponentially with the number of constituent atoms.

An example of two isomers and their spectra is given in Figure 1. Notice, we are concerned only with finding the 2D structure as represented by standard stick and ball diagrams. These structures have a one-to-one correspondence with the full chemical name of the molecule as given by the IUPAC nomenclature [5].

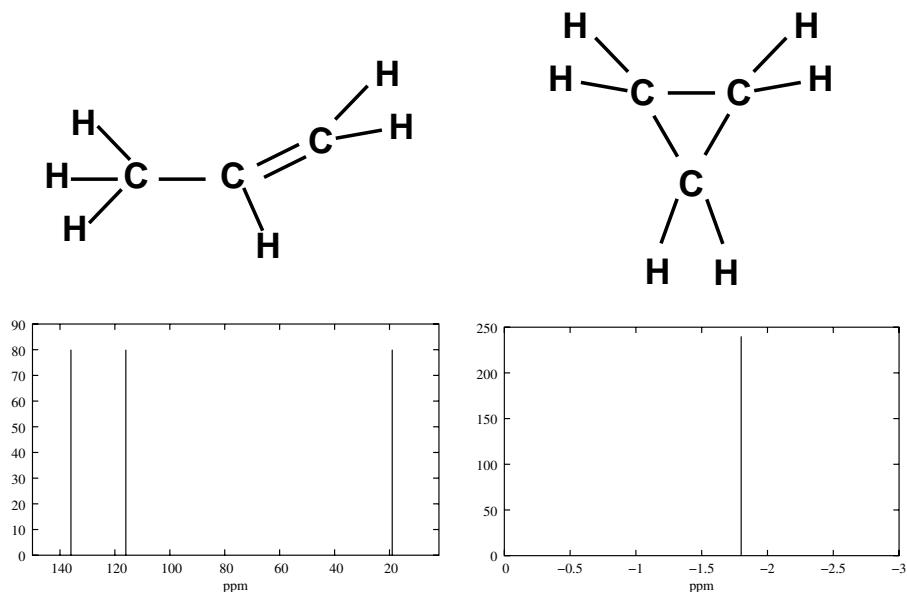


Fig. 1. Two-dimensional representations of the two constitutional isomers of the empirical formula C_3H_6 and their respective ^{13}C NMR spectra. In propane (left), three distinct shifts can be seen because each carbon atom's electronic environment is distinct. In cyclopropane (right), only one shift is visible because the 'view' from each carbon atom is identical.

In our version of the SSP, we assume that there is available some 'prior knowledge' in the form of a dataset giving the known 2D chemical structures and spectral shift values of organic molecules. Using this, we wish to infer the overall structure of an unknown molecule by identifying its likely substructural components — substructures that occur in molecules that exhibit similar NMR shifts to our unknown molecule.

A number of alternative formal definitions of the resulting inference problem could be given, based on measures such as 0-1 loss, or precision and recall (as used in classification problems). However, we choose here to allow that the inference method returns not one, but several attempts at inferring the structure. This

is because the problem is hard and a single attempted structure is unlikely to be correct, so measures of 0-1 loss would be unhelpful. Moreover, in practice, chemists would be happier to receive a small number of candidate molecular structures to investigate further, rather than one single answer that turns out to be inaccurate.

What we thus measure is the position of the true structure in a ranking (by internal fitness measure) of the candidate isomers generated by the inference method. Because our inference method is based on ACO, a stochastic meta-heuristic, we must run the algorithm several times to evaluate performance, so to account for this, we report the overall rank of the true structure within a list (sorted by fitness) of all the unique structures generated by the ACO over the multiple runs performed. We also indicate the fraction of runs (out of those performed) on which the true structure is generated.

3 Ant Colony Optimization Approach

The method that we propose for tackling the SSP has three main steps.

1. **Data preparation:** Identify all *substructures* up to a given size that exist in the database; construct a matrix recording the frequency with which each substructure and spectral shift co-occur in the data. Split this matrix into two, one pertaining to smaller substructures and one pertaining to larger ones. (This whole step need only be done once for a given database of molecules. And if new data becomes available, the matrices can be *incrementally* updated by a trivial procedure.)
2. **Set constraints based on the query:** Once we have a query — an unknown molecule to identify — use its empirical formula to remove from consideration any substructures in the matrices that cannot be a part of the final structure, i.e., those that contain an atom that is not part of the given molecule and/or substructures that never produce any of the observed spectral shifts.
3. **Directed search:** Search for complete structures that match the empirical formula. Ants construct candidate structures from the smaller substructures identified in Step 1. The candidate structures are evaluated with reference to the larger-substructure frequency matrix from Step 1 via a maximum weighted assignment algorithm (see Section 3.6).

The underlying ant algorithm that we use for the directed search part is based on the MAX-MIN Ant System [6] (see Algorithm 1).

3.1 Data Preparation: Frequency Matrices

We are interested in building an approximate probability distribution over the substructures contained within the database. To obtain this information, the graph structure of each molecule can be decomposed into its subgraphs. Only subgraphs of limited size need to be found, where the size refers to the number

Algorithm 1. Ant colony optimization algorithm for structure elucidation

Input query: An empirical formula and its ^{13}C NMR spectral shift pattern
 Prior knowledge: small substructures freq. matrix, large substructures freq. matrix
 Constrain the search: Delete incompatible rows in the frequency matrices
 Global best fitness $\leftarrow 0$

while Termination conditions not satisfied **do**

for $n = 1$ to n_{ants} **do**

Construct an ordered list of small substructures compatible with the empirical formula, using a pheromone matrix to guide choices

Make all structures that are chemically possible from the ordered list

for $j = 1$ to $n_{\text{structures}}$ **do**

Evaluate j th structure using a maximum weight assignment algorithm

end for

Record best fitness for this ant

end for

Record best fitness for this iteration

Update global best fitness

Update pheromone matrix with best fitness structure of the iteration

if global best structure \neq iteration best structure **then**

Update pheromone matrix with global best structure

end if

end while

Output: ranked list of candidate isomers and their estimated fitness values

of carbon atoms in the substructure. (All the non-carbon atoms bound to these carbons are also included). We use an algorithm that enumerates all valid substructures of sizes 2-carbon, 3-carbon and 4-carbon. Once this has been done for every molecule in the database, we are able to correlate substructures with spectral shifts. This is done by populating a matrix, which has rows representing substructures and columns representing shift frequencies (suitably binned into small value ranges) so that each element of the matrix records the number of co-occurrences of a substructure and a particular spectral shift. It is thus a representation of the joint probability of substructures and shifts (when correctly normalized).

We use this data in two ways in the ACO algorithm. We make one frequency matrix containing all 2- and 3-carbon substructures. These substructures are used as the solution components out of which the ants will construct full solutions. We make a second frequency matrix containing all the 4-carbon substructures only. This matrix is used to evaluate solutions (see Section 3.6).

3.2 Construction Graph Structure

The solution directly constructed by an ant is an ordered set of 2- and 3-carbon substructures, $s = \langle s_1, s_2, \dots, s_k \rangle$, having a variable number of elements k . An ant begins with a partial solution $s^p = \emptyset$ and selects s_1 from the pool of available small substructures (with replacement) and adds it to s^p . The ant then makes the choice of s_2 adds it to s^p , and so on. The pool of available substructures is

updated after each choice to reflect the constraint given by the empirical formula. The construction of a solution s is completed if the ant successfully completes a structure with the required empirical formula. It may also terminate construction, in the event that it is no longer possible to complete the empirical formula, which can occur if the addition of any substructure would result in exceeding the empirical formula in at least one atom type. In the case of terminating a solution without successfully completing it, the ant returns to the start of construction, setting $s^p = \emptyset$ and with the pool of available structures reset. An ant continues constructing solutions until it is successful.

The choice of substructure an ant makes at each step is mediated by both pheromone and heuristic information. Both of these sources of information help the ants to avoid making choices that lead to constructions ending in incomplete termination. A standard arc selection method is used [1], with the probability of selecting component c_{ij} , $i = 1, \dots, k$, $j = 1, \dots, |D_i|$ being given by

$$p(c_{ij}|s^p) = \frac{\tau_{ij}^\alpha \cdot \eta_{ij}^\beta}{\sum_{c_{il} \in N(s^p)} \tau_{il}^\alpha \cdot \eta_{il}^\beta}, \quad \forall c_{ij} \in N(s^p)$$

where D_i is the domain of the decision variable (the set of substructures available to go in position i), τ_{ij} and η_{ij} represent the pheromone and heuristic information, respectively, and α and β are used to set the influence of these; $N(s^p)$ represents the feasible neighbours of the partial solution s^p .

3.3 Local Search: Translating Ant Solutions to Full Structures

The ordered list of substructures generated by an ant does not uniquely define an isomeric structure. This is because the substructures could be joined to each other in numerous ways. The ordering of the substructures is, however, intended to encode at least partially the preferred way in which the substructures should be joined. Thus, the solution encoded by an ant is interpreted as an instruction to join s_2 to s_1 , then s_3 to s_2 , and so on. But there are still numerous chemically valid ways in which this can be done that lead to different structural forms. These structural forms can be enumerated, given the ant solution. Therefore, an ant's construction is regarded as defining an ensemble of possible structures and the later evaluation of the ant solutions is done with respect to the best solution in the ensemble. Explicit details of the procedure for performing this enumeration of structures are given in [7]; space limitations prevent us from giving them here.

3.4 The Pheromone Matrix and Its Initialization

The pheromone matrix has m rows and k_{max} columns, where m is the number of substructures in the pool initially (after constraining), and k_{max} is the maximum possible number of substructures that could be needed to construct a valid isomer. It is simple to see that k_{max} is upper bounded by the number of carbon atoms in the empirical formula divided by two, since each substructure that we use to construct solutions has at least two C atoms.

An ant choosing substructural element s_j looks in the j th column of the pheromone matrix. The pheromone is thus on the nodes of the construction graph, and represents the relative desirability of selecting a particular substructure at a particular position in an ant solution (which as stated above represents an ordering of selected substructures). The pheromone matrix is initialized here with the maximum value τ_{max} , following Max-Min Ant System.

3.5 Heuristic Information

The heuristic information η_{ij} is given by

$$\eta_{ij} = \max(1, I(c_{ij} \text{ completes EF}) \cdot 1000) \cdot \prod_{a \in A} h_{ij}^a,$$

where A is the set of different atom types in the target empirical formula,

$$h_{ij}^a = \frac{\sum_{c_{il} \in N(s^p)} 1}{\sum_{c_{il} \in N(s^p)} I(c_{il} \text{ contains atom type } a)}$$

and $I(\cdot)$ is the indicator function, which has value 1 if its argument is true, and zero otherwise. Thus, η_{ij} rewards a substructure c_{ij} if it contains an atom type a which is in the target empirical formula and if this atom is rare (or infrequent) in other available substructures. This encourages the picking of substructures containing rare atoms early on in solution construction, which helps prevent building candidate solutions that cannot be completed. The heuristic value of a substructure is further rewarded (by a factor of 1000) if its selection would complete the target empirical formula; this prevents making poor decisions towards the end of solution construction.

3.6 Evaluation Using the Maximum Weighted Assignment

To evaluate a candidate isomer, it is first mined for all its constituent 4-carbon substructures. A match between these larger substructures and those in the frequency matrix that co-occur frequently at similar spectral shifts would indicate a credible structure.

To assess the overall quality of these matches, we find the best assignment of shifts to substructures possible, and evaluate this assignment. Specifically, we have a set M of mined 4-C substructures and a set of observed shifts F . We have a weight matrix $W : M \times F \rightarrow R$ that stores the number of co-occurrences of each $m \in M$ and each shift $f \in F$ within the frequency matrix.

We would like to assign each shift precisely one carbon atom, but the substructures contain 4 carbons each. Therefore, we can allow each substructure to be matched with up to 4 shifts. To facilitate solving this as a standard bipartite graph matching problem, we can just copy each element of M four times to obtain an expanded set Q and expand our weight function to be $W : Q \times F \rightarrow R$, by simply repeating the weights four times. We now seek an assignment $g : F \rightarrow Q$ such that

$$\sum_{f \in F} W(f, g(f))$$

is maximized. This is a bipartite maximum weighted matching problem (or assignment problem) and can be solved by various methods including the Hungarian algorithm [8], though we used a restart hillclimbing method.

The solution to this problem here gives the most favourable interpretation of whether the set of substructures within the isomer could explain the shift pattern seen.

3.7 Pheromone Update

The elements in the pheromone matrix that appear in a solution to be rewarded (an iteration best ant or elite ant) are updated according to the following equation:

$$\tau_{i,j}(t+1) = (1 - \rho) \cdot \tau_{i,j}(t) + \Delta\tau_{ij}^{best}$$

where $\Delta\tau_{ij}^{best} = 0$ if c_{ij} is not a component used in the best ant, and is otherwise the raw score derived from the weighted assignment problem described above.

Pheromones are forced to remain within the ranges set by τ_{min} and τ_{max} , by setting values below (respectively above) these to the respective bounding value.

4 Preliminary Experimental Results

Our experiments were conducted with the parameters of the ACO set as shown in Table 1(i). The basis of our experiments was a database of molecules compiled by us, as described in Table 1(ii).

The performance of the structure elucidation method is evaluated in two ways here. First, we examine if it can recover the structure of a molecule that is in the prior knowledge data itself. This is already a hard problem (and is NOT equivalent to testing on the training set in a classification/supervised learning task, because the space of *possible* structures that we search is still very large — much larger than our whole database of known structures, so we are not just learning class labels). These results are reported in Table 2.

Table 1. (i) Parameters of the ACO algorithm; (ii) Details of the database of known molecules

(i)		(ii)	
Parameter	value	Training set info	
$nants$	5	Number of molecules	2 873
max iterations	80	Maximum MW	500
τ_{min}	0.5	Minimum MW	50
τ_{max}	10	Total number of atom types	16
ρ	0.01	Number of 2C and 3C substructures mined	2 881
α	1.0	Number of 4C substructures mined	5 926
β	1.0		

Secondly, we verify the performance on molecules not in the initial knowledge-base. This is achieved here by ‘holding out’ certain molecules we wish to test from contributing to the frequency matrices. Due to some limitations of our data-sets, we can only do this for two molecules at present (see Table 2, bottom).

Table 2. Test results on a range of small organic molecules. The target molecule is found in all cases and in almost all runs. Often the approximate fitness of the true structure means that it is ranked highly amongst the other candidates. Bottom: results on hold-out data.

No. of carbon atoms	Molecule name (IUPAC convention)	Empirical formula	No. of isomers as enum- erated by [9]	Rank by fitness (total no. of unique isomers gnrnd.)	Number of runs target gnrnd. / total runs
4	1-bromo-2-fluorobutane	C ₄ H ₈ BrF	12	1st (2)	27/27
	4-aminobutanenitrile	C ₄ H ₈ N ₂	633	1st (27)	17/17
	1-methoxypropan-2-ol	C ₄ H ₁₀ O ₂	28	1st (12)	18/18
5	(1E)-1,2-diiodopent-1-ene	C ₅ H ₈ I ₂	88	3rd (8)	14/14
6	1-(allyloxy)propan-2-ol	C ₆ H ₁₂ O ₂	1313	17th (396)	10/11
	1-propoxypalan-2-ol	C ₆ H ₁₄ O ₂	179	6th (127)	8/8
	1,1'-dithiodipropane	C ₆ H ₁₂ S ₂	timeout	1st (66)	15/15
7	1-butoxypropan-2-ol	C ₇ H ₁₆ O ₂	463	15th (292)	20/20
Hold-out data results:					
6	1-propoxypalan-2-ol	C ₆ H ₁₄ O ₂	179	2nd= (127)	5/5
7	1-butoxypropan-2-ol	C ₇ H ₁₆ O ₂	463	10th (292)	4/4

The results reported in Table 2 are currently limited by a couple of factors that have prevented a larger study. These are that: (i) our system of joining substructures cannot currently generate ring structures, which means that a significant fraction of structures cannot be tested yet; and (ii) at several points, our code calls proprietary software to convert between different representations of chemical structures (namely, SMILES strings and MOL files), which creates a substantial computational bottleneck that prevents us from testing the larger structures in our database. We are working to overcome both of these factors, which are certainly not inherent problems of the system.

Despite the limitations, the results are positive on the cases we have tested, with true structures being correctly recovered in all cases, and often ranked highly by the assignment method compared with other structures generated. On the hold-out data, the ACO system worked at least equally well when these isomers were removed from the prior knowledge database as when they were in it. Much more testing is required to understand the effect of the distribution of isomers stored in the database on performance; but this initial test indicates that it is not necessary to have seen the molecule before to predict its structure using our system.

5 Related Work on Structure Elucidation

In comparison to the number of applications available for spectrum prediction, the field of structure elucidation is relatively small and immature. Most attempts to address this issue are built upon an expert system with an inherent rule base.

A common feature is the requirement for an empirical formula. From this, all possible isomers are generated and a spectrum is predicted for each one, allowing for similarity ranking against the original query spectrum. Although this significantly narrows the search space, typically thousands of isomeric forms may remain. If there are significant distinctions between the spectra, the structure corresponding to the top ranking spectrum can be taken as the structure causing the experimental spectra. However, if several top-most ranking spectra are very similar, further analysis may be required. It should be noted at this point that both the accuracy of spectrum prediction and similarity ranking are of primary importance in structure elucidation, because the larger the margin of error in these, the more likely it will be that the predicted structure will be incorrect.

There are several factors to be taken into account during ranking, including matching the number of nuclei visible in the spectrum, chemical shifts values and scalar couplings. It can be difficult to determine corresponding shifts between predicted and experimental spectra, especially where multiple shifts occur within a small separation. A study has highlighted how matrices can be used to detect optimal matches between experimental and predicted spectra [10]. The first two expert systems developed for this area, CONGEN [11] and GENOA [12], generated isomeric forms, from which a specialist would select a likely structure. Both systems required considerable human interaction in forming lists of favoured or unlikely fragments, but GENOA allowed fragment overlap within the isomers constructed.

A more modern trend in structure elucidation applications is to utilize several different spectral types in order to perform elucidation, for example multiple dimensions, element types or analytical techniques. This rapidly reduces the chemical search space and facilitates ranking. Programs such as CHEMICS[13], X-PERT [14], and StrucEluc [15] use such supplementary data to determine specific libraries and rules which should be accessed in order to improve search results.

A more unusual approach is taken by the program Genius [16], which uses a genetic algorithm for structure generation. A neural network is used to categorise the electronic environment of each carbon in an isomer and then to predict its spectrum. Using a GA for structure generation means not all isomers need be initially generated, potentially narrowing the search space. The level of similarity to the query spectrum determines which chromosomes are allowed to mate and reproduce into the next generation. Runs can be stopped either by correct structure determination (matching chemical shifts), time limits (after a set number of generations), or accuracy limits (when chemical shifts lower than those in the experimental spectrum are achieved).

6 Summary and Future Work

A system for tackling the spectrum-to-structure problem based on ACO has been presented. The system does not use expert rules, nor does it rely on predicting spectra from structures; instead, iterative heuristic search is combined with the use of a knowledge-base of identified structures and their characteristic spectra. On the data used to test the system here, it produced sets of proposed structures that contained the true structure in all cases, even when the space of possible isomers was large (i.e. containing over a thousand feasible structures). On several occasions, the true structure was the isomer ranked highest by the system. Moreover, the set of structures that a chemist may regard as likely candidates can potentially be reduced further, by taking account of the pheromone trail information, rather than considering every structure generated. Testing has obviously been very limited to date so it is not possible to draw any more than preliminary conclusions from this. However, we are encouraged by these results to continue further investigations.

The system now needs to be extended to tackle different molecular forms, such as rings, which it is currently incapable of identifying (see [7] for more details). We need to test the system further and compare it with alternative approaches, including existing spectrum-to-structure methods and simple baseline approaches. Such testing will require us to gather more high-quality NMR spectral data for similar and larger molecules, to allow much larger studies to be done, with more quantitative reporting of success rates as well as computation times.

The spectrum-to-structure problem will continue to be an important one in the pharmaceutical and systems biology arena. There is a growing need for fast identification of molecules that have been manufactured artificially, such as candidates for active pharmaceuticals (drugs), or naturally-occurring molecules that have never been characterized before, such as many of the metabolic products of biological cells [17,18]. The work started here may eventually allow us to build systems that are more scalable — requiring less human input and expertise and less time-consuming training — than currently available ones.

Acknowledgments. Many thanks to Dr Lee Griffiths (Astra Zeneca, Alderley Park) and Dr Bryn Roberts (formerly Astra Zeneca, Alderley Park) for advice on NMR spectroscopy and for providing access to chemical shift data. Caroline Farrelly was supported by a CASE studentship from Astra Zeneca and EPSRC, UK. Joshua Knowles is supported by a David Phillips Research Fellowship from BBSRC, UK.

References

1. Dorigo, M., Stützle, T.: *Ant Colony Optimization*. MIT Press, Cambridge (2004)
2. Aha, D.: *Lazy Learning*. Kluwer Academic Publishers, Norwell (1997)
3. Stützle, T., Hoos, H.: MAX-MIN Ant system and local search for combinatorial optimization problems. *Meta-Heuristics: Advances and Trends in Local Search Paradigms for Optimization*, 313–329 (1999)

4. Gambardella, L., Dorigo, M.: An Ant Colony System Hybridized with a New Local Search for the Sequential Ordering Problem. *INFORMS Journal on Computing* 12(3), 237–255 (2000)
5. <http://www.chem.qmul.ac.uk/iupac/>
6. Stützle, T., Hoos, H.: MAX-MIN Ant System. *Future Generation Computer Systems* 16(8), 889–914 (2000)
7. Farrelly, C.: From Spectrum to Structure Using Machine Learning. PhD thesis, School of Chemistry, University of Manchester, UK (2008)
8. Munkres, J.: Algorithms for the Assignment and Transportation Problems. *Journal of the Society of Industrial and Applied Mathematics* 5(1), 32–38 (1957)
9. MOLGEN Tool, <http://molgen.de/?src=documents/molgenonline>
10. Griffiths, L., Bright, J.: Towards the automatic analysis of ¹H NMR spectra: Part 3. Confirmation of postulated chemical structure. *Magn. Reson. Chem.* 40, 623–634 (2002)
11. Carhart, R., Smith, D., Brown, H., Djerassi, C.: Applications of artificial intelligence for chemical inference. XVII. Approach to computer-assisted elucidation of molecular structure. *Journal of the American Chemical Society* 97(20), 5755–5762 (1975)
12. Carhart, R., Smith, D., Gray, N., Nourse, J., Djerassi, C.: GENOA: A Computer Program for Structure Elucidation Utilizing Overlapping and Alternative Substructures. *J. Org. Chem.* 46, 1708–1718 (1981)
13. Sasaki, S., Kudo, Y.: Structure elucidation system using structural information from multisources: CHEMICS. *Journal of Chemical Information and Computer Sciences* 25(3), 252–257 (1985)
14. Elyashberg, M., Martirosian, E., Karasev, Y., Thiele, H., Somberg, H.: X-PERT: a user-friendly expert system for molecular structure elucidation by spectral methods. *Analytica Chimica Acta* 337(3), 265–286 (1997)
15. Elyashberg, M., Blinov, K., Williams, A., Martirosian, E., Martin, G.: Application of a New Expert System for the Structure Elucidation of Natural Products from the 1D and 2D NMR Data. *J. Nat. Prod.* 65(5), 693–703 (2002)
16. Meiler, J., Will, M.: Genius: A genetic algorithm for automated structure elucidation from C-13 NMR spectra. *Journal of the American Chemical Society* 124(9), 1868–1870 (2002)
17. Kell, D.: Systems biology, metabolic modelling and metabolomics in drug discovery and development. *Drug Discovery Today* 11(23-24), 1085–1092 (2006)
18. Kell, D.: Metabolomic biomarkers: search, discovery and validation. *Exp Rev Mol Diagn* 7(4), 329–333 (2007)

Rigorous Analyses for the Combination of Ant Colony Optimization and Local Search*

Frank Neumann¹, Dirk Sudholt², and Carsten Witt²

¹ Max-Planck-Institut für Informatik, Saarbrücken, Germany
`fne@mpi-inf.mpg.de`

² Informatik 2, Technische Universität Dortmund, Dortmund, Germany
`{sudholt,cw01}@ls2.cs.uni-dortmund.de`

Abstract. Ant colony optimization (ACO) is a metaheuristic that produces good results for a wide range of combinatorial optimization problems. Often such successful applications use a combination of ACO and local search procedures that improve the solutions constructed by the ants. In this paper, we study this combination from a theoretical point of view and point out situations where introducing local search into an ACO algorithm enhances the optimization process significantly. On the other hand, we illustrate the drawback that such a combination might have by showing that this may prevent an ACO algorithm from obtaining optimal solutions.

1 Introduction

Ant colony optimization (ACO) is a metaheuristic that has been applied successfully to various combinatorial optimization problems. Often ACO is combined with local search methods [1,2,3]. Experimental investigations show that the combination of ACO with a local search procedure improves the performance significantly. On the other hand, there are examples where local search cannot help to improve the search process or even mislead the search process [4]. Therefore, it is interesting to figure out how the incorporation of local search into ACO algorithms can significantly influence the optimization process.

The goal of this paper is to investigate this effect from a theoretical point of view, using rigorous runtime analyses. The analysis of ACO algorithms with respect to their runtime behavior is a new research direction, first step to which appeared in [5,6]. Recently, initial results with respect to the runtime behavior of variants of the MAX-MIN Ant System (MMAS) [7] have been obtained [8,9,10,11].

Our aim is to point out situations where the effect of local search becomes visible in a way that can be tackled by rigorous arguments. Therefore we present functions where MMAS variants with and without local search show a strongly different runtime behavior. On one function, MMAS with local search outperforms MMAS without local search, while on a different function the effect is

* The work of D. Sudholt and of C. Witt was supported by the Deutsche Forschungsgemeinschaft (DFG) as a part of the Collaborative Research Center “Computational Intelligence” (SFB 531).

reversed. The differences shown in this paper are so drastic that the question of whether to use local search or not decides between polynomial and exponential runtimes.

The outline of this paper is as follows. In Section 2, we define MMAS variants with and without local search. Section 3 discusses different effects that a combination of ACO and local search can have. In Section 4, we present a rigorous analysis showing the benefits of such a combination. Contrarily, in Section 5 we investigate a different function and prove the opposite effect. We finish with some conclusions.

2 Algorithms

We consider the runtime behavior of two ACO algorithms for the optimization of pseudo-Boolean functions. Solutions for a given problem are constructed by a random walk on a so-called construction graph C according to pheromone values τ on the edges.

Algorithm 1 (Construct(C, τ))

- 1.) $v := s$, mark v as visited.
- 2.) While there is an unvisited successor of v in C :
 - a.) Let N_v be the set of unvisited successors of v and $T := \sum_{(v,w) | w \in N_v} \tau_{(v,w)}$.
 - b.) Choose $w \in N_v$ with probability $\tau_{(v,w)} / T$.
 - c.) Mark w as visited, set $v := w$ and go to 2.).
- 3.) Return the solution x and the path $P(x)$ constructed by this procedure.

We examine the construction graph displayed in Figure 1 and known as *Chain* [12]. Constructing bit strings of length n , the decision whether a bit x_i , $1 \leq i \leq n$, is set to 1 is made at node $v_{3(i-1)}$. In case the edge $(v_{3(i-1)}, v_{3(i-1)+1})$ (upwards) is chosen, x_i is set to 1 in the constructed solution. Otherwise the edge $(v_{3(i-1)}, v_{3(i-1)+2})$ (downwards) is taken, and $x_i = 0$ holds. After this decision has been made, the only available edge leads to the decision node for the next bit.

We ensure $\sum_{(u,\cdot) \in E} \tau_{(u,\cdot)} = 1$ for all decision nodes $u = v_{3i}$, $0 \leq i \leq n-1$. Let $p_i = \text{Prob}(x_i = 1)$ be the probability of setting the bit x_i to 1 in the next constructed solution. Due to our setting, we have $p_i = \tau_{(3(i-1), 3(i-1)+1)}$ and $1 - p_i = \tau_{(3(i-1), 3(i-1)+2)}$, i.e., the pheromone values correspond directly to the probabilities for choosing the bits in the constructed solution. In addition,

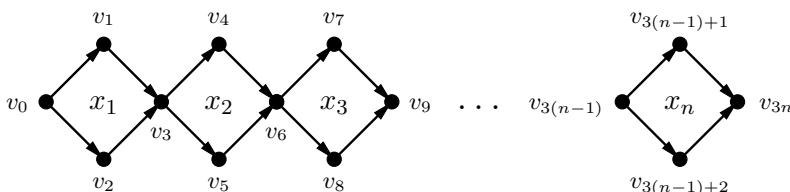


Fig. 1. Construction graph for pseudo-Boolean optimization

following the MAX-MIN ant system by Stützle and Hoos [7], we restrict each $\tau_{(u,v)}$ to the interval $[1/n, 1 - 1/n]$ such that every solution always has a positive probability of being chosen.

Depending on whether edge (u, v) is contained in the path $P(x)$ of the constructed solution x , the pheromone values are updated to τ' as follows:

$$\begin{aligned}\tau'_{(u,v)} &= \min \left\{ (1 - \rho) \cdot \tau_{(u,v)} + \rho, 1 - \frac{1}{n} \right\} && \text{if } (u, v) \in P(x) \text{ and} \\ \tau'_{(u,v)} &= \max \left\{ (1 - \rho) \cdot \tau_{(u,v)}, \frac{1}{n} \right\} && \text{if } (u, v) \notin P(x).\end{aligned}$$

The following algorithm, which we call MMAS*, has been defined by Gutjahr and Sebastiani [9] under the original name MMAS_{bs}. Here, in each generation the best solution obtained during the run of the algorithm, called best-so-far solution, is rewarded. Another property of the model is that the best-so-far solution may not switch to another one that has the same fitness.

Algorithm 2 (MMAS*)

- 1.) Set $\tau_{(u,v)} = 1/2$ for all edges (u, v) .
- 2.) Compute a solution x^* using $\text{Construct}(C, \tau)$.
- 3.) Update the pheromone values with respect to x^* .
- 4.) Compute a solution x using $\text{Construct}(C, \tau)$.
- 5.) If $f(x) > f(x^*)$, set $x^* := x$.
- 6.) Update the pheromone values with respect to x^* .
- 7.) Go to 4.).

We enhance the MMAS* with local search. In this work, $\text{LocalSearch}(x)$ is a procedure that, starting from x , repeatedly replaces the current solution by a Hamming neighbor with strictly larger fitness until a local optimum is found. We do not specify a pivoting rule, hence we implicitly deal with a class of algorithms.

Algorithm 3 (MMAS-LS*)

- 1.) Set $\tau_{(u,v)} = 1/2$ for all edges (u, v) .
- 2.) Compute a solution x using $\text{Construct}(C, \tau)$.
- 3.) Set $x^* := \text{LocalSearch}(x)$.
- 4.) Update the pheromone values with respect to x^* .
- 5.) Compute a solution x using $\text{Construct}(C, \tau)$.
- 6.) Set $z := \text{LocalSearch}(x)$.
- 7.) If $f(z) > f(x^*)$, set $x^* := z$.
- 8.) Update the pheromone values with respect to x^* .
- 9.) Go to 5.).

The fitness functions considered in the following only have a linear number of fitness values, hence the number of iterations in one local search call is bounded by $O(n)$. Depending on the pivoting rule, the number of fitness evaluations needed to find a better Hamming neighbor may vary; however, it is trivially bounded by n . Hence, the number of function evaluations is at most by a factor $O(n^2)$ larger than the number of generations.

We consider as performance measure the number of generations, also referred to as optimization time. This yields an advantage for MMAS-LS* w. r. t. fitness evaluations. However, the upcoming performance gaps are between polynomial and exponential values, and an advantage of order n^2 is negligible.

3 The Effect of Combining ACO and Local Search

The effect of using local search with ACO algorithms is manifold. Firstly, local search can help to find good solutions more quickly as it increases the “greediness” within the algorithm. Moreover, the pivoting rule used in local search may guide the algorithm towards certain regions of the search space. For example, first ascent pays more attention to the first bits in the bit string, which may induce a search bias. However, we will not deal with this effect in our study. In particular, our functions are designed such that the pivoting rule is not essential.

There is, however, another effect that we want to investigate more closely. The pheromone values induce a sampling distribution over the search space. On a typical fitness landscape, once the best-so-far solution has reached a certain quality, sampling new solutions with a high variance becomes inefficient and the current best-so-far solution x^* is maintained for some time. Previous studies on MMAS variants [9,10] have shown that then the pheromones quickly reach the upper and lower bounds corresponding to x^* . This means that the algorithm turns to sampling close to x^* . In other words, MMAS variants typically reach a situation where the “center of gravity” of the sampling distribution follows the current best-so-far solution and the variance of the sampling distribution is low.

When introducing local search into an MMAS algorithm, this may not be true. Local search is able to find local optima that are far away from the current best-so-far solution. In this case the “center of gravity” of the sampling distribution is far away from the best-so-far solution.

Assume there is a path of Hamming neighbors with increasing fitness leading to a local optimum. Assume further that all points close to the path have lower fitness. Then for MMAS* it is likely that the sampling distribution closely follows the path. The path of increasing fitness need not be straight. In fact, it can make large bends through the search space until a local optimum is reached. On the other hand, MMAS-LS*, when starting with the same setting, will reach the local optimum within a single iteration of local search. Then the local optimum becomes the new best-so-far solution x^* while the sampling distribution is still concentrated around the starting point. In the following generations, as long as the best-so-far solution is not exchanged, the pheromone values on all bits synchronously move towards their respective bounds in x^* . This implies for the sampling distribution that the “center of gravity” takes a (sort of) direct route towards the local optimum, irrespective of the bent path taken by local search. An illustration is given in Figure 2.

Consequences are that different parts of the search space are sampled by MMAS* and MMAS-LS*, respectively. Moreover, with MMAS* the variance in the solution construction is always quite low as the sampling distribution is

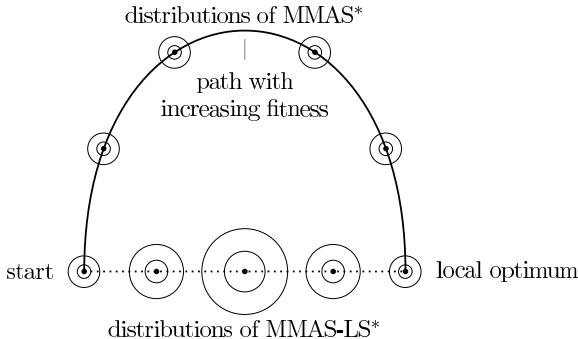


Fig. 2. A sketch of the search space showing the behavior of MMAS* and MMAS-LS*. The dots and circles indicate the sampling distributions of MMAS* and MMAS-LS*, resp., at different points of time. While the distribution of MMAS* tends to follow the fitness-increasing path from left to right, the distribution of MMAS-LS* takes a direct route towards the local optimum.

concentrated on certain points on the path. But when the best-so-far solution with local search suddenly moves a long distance, the variance in the solution construction may be very high as the bits differing between the starting point and x^* may have pheromones close to 1/2. These bits are assigned almost randomly, which strongly resembles a uniform crossover operation well-known in evolutionary computation.

Our aim in the following is to create functions where MMAS* and MMAS-LS* have a different runtime behavior. Moreover, we want the performance difference to be drastic in order to show how deep the impact of local search can possibly be. To this end, we exploit that the sampling distributions can follow different routes through the search space. For one function we place a target region with many global optima on the straight line between starting point and local optimum and turn the local optimum into a trap that is hard to overcome. In such a setting, we expect MMAS-LS* to drastically outperform MMAS*. These ideas are made precise in Section 4. On the other hand, if the region of global optima is made a region of traps and the global optimum is very close to the local optimum, MMAS* has a clear advantage over MMAS-LS*. Another function following this idea is defined and analyzed in Section 5.

4 Benefits of Combining ACO and Local Search

We now formally define a function where local search is beneficial according to the ideas from Section 3. It is named SP-Target (short path with target). The path with increasing fitness is given by the set $SP = \{1^i 0^{n-i} \mid 0 \leq i \leq n\}$. The path ends with the local optimum 1^n . A large target area containing all global optima is specified by $OPT = \{x \mid |x|_1 \geq (3/4) \cdot n \wedge H(x, SP) \geq n/(\gamma \log n)\}$, where $H(x, SP)$ denotes the Hamming distance of x to the closest search point of SP and $\gamma \geq 1$ is a constant to be chosen later. For all remaining search points,

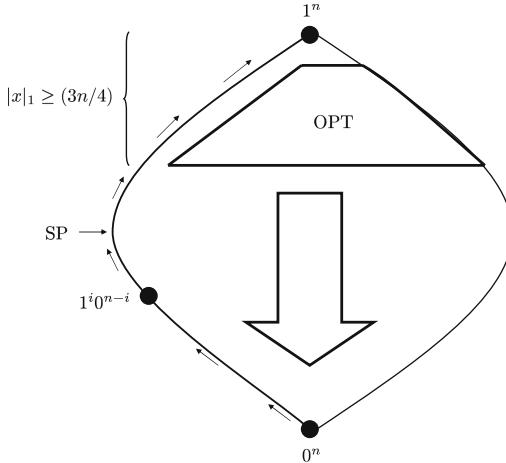


Fig. 3. Illustration of the Boolean hypercube and the function SP-Target. Arrows indicate gradients of increasing fitness.

the function SP-Target gives hints to reach 0^n , the start of the path SP. We denote by $|x|_0$ the number of zeros in x and by $|x|_1$ the number of ones in x .

$$\text{SP-Target}(x) := \begin{cases} |x|_0 & x \notin (\text{SP} \cup \text{OPT}) \\ n + i & x = 1^i 0^{n-i} \in \text{SP} \\ 3n & x \in \text{OPT}. \end{cases}$$

The function SP-Target is sketched in Figure 3. Note that we have actually defined a class of functions dependent on γ . All following results will hold for arbitrary constant $\gamma \geq 1$ unless stated otherwise.

The following theorem shows that MMAS* without local search is not successful. We restrict ourselves to polynomially large $1/\rho$ here and also in the following as otherwise the ACO component would be too close to random search.

Theorem 1. *Choosing $\rho = 1/\text{poly}(n)$, the optimization time of MMAS* on SP-Target is $2^{\Omega(n^{2/9})}$ with probability at least $1 - 2^{-\Omega(n^{2/9})}$.*

To prove the preceding theorem, we have to take into account situations where the pheromone values of MMAS* have not yet reached their bounds and the construction procedure samples with high variance. This is the case in particular after initialization. The following lemma will be used to check the probability of finding the optimum in the early steps of MMAS* on SP-Target.

Lemma 1. *If the best-so-far solution of MMAS* has never had more than $2n/3$ 1-bits, the probability of creating a solution with at least $3n/4$ 1-bits is $2^{-\Omega(n)}$ in each generation.*

Proof. The proof is an application of Chernoff bounds w. r. t. the number of ones in the solutions created by MMAS*. Let the potential $P_t := p_1 + \dots + p_n$ at

time t denote the current sum of the probabilities of sampling ones over all bits, which, by definition of the construction procedure, equals the expected number of ones in the next constructed solution. Observe that $P_t \leq 2n/3$ implies by Chernoff bounds that the probability of creating a solution with at least $3n/4$ 1-bits is $2^{-\Omega(n)}$. We now show: if all best-so-far solutions up to time t have at most $2n/3$ ones, then $P_i \leq 2n/3$ for $0 \leq i \leq t$. This will prove the lemma.

For the last claim, we denote by k the number of ones in the best-so-far solution according to which pheromones are updated. Due to the pheromone update mechanism, the new potential P_{i+1} is obtained from P_i and k according to $P_{i+1} = (1-\rho)P_i + k\rho$. Hence, if $P_i \leq 2n/3$ and $k \leq 2n/3$ then also $P_{i+1} \leq 2n/3$. The claim follows by induction since $P_0 = n/2 \leq 2n/3$. \square

Proof (of Theorem 1). We distinguish two phases in the run according to the best-so-far solution x^* . Phase 1 holds as long as $x^* \notin \text{SP}$ and $|x^*|_1 \leq 2n/3$, and Phase 2 applies as long as $x^* \in \text{SP}$. Our aim is to show that a typical run passes through the two phases in their order with a failure probability of $2^{-\Omega(n^{2/9})}$. The probability of finishing the second phase will be bounded by $2^{-\Omega(n^{2/9})}$ for each step of the phase. This implies the theorem as, by the union bound, the total probability in $2^{cn^{2/9}}$ generations, $c > 0$ a small constant, is still $2^{-\Omega(n^{2/9})}$.

Consider the first (and best-so-far) solution x^* created by MMAS*. By Chernoff bounds, $n/3 \leq |x^*|_1 \leq 2n/3$ with probability $1 - 2^{-\Omega(n)}$. There is only a single solution in SP for each value of $|x^*|_1$. By the symmetry of the construction procedure, we conclude $\text{Prob}(x^* \in \text{SP} \mid |x^*|_1 = k) = 1/\binom{n}{k}$. The last expression is $2^{-\Omega(n)}$ for $n/3 \leq k \leq 2n/3$. Hence, with probability $1 - 2^{-\Omega(n)}$, there is a non-empty Phase 1. By Lemma 1, the probability that a specific generation in Phase 1 creates an optimum is $2^{-\Omega(n)}$. Otherwise, the behavior is as for MMAS* on the function $|x|_0$. Using $\rho = 1/\text{poly}(n)$ and the analyses for the symmetric function $|x|_1$ from [10] and [9], the expected time until the first phase is finished is polynomial. By the law of total probability and the union bound, the total failure probability in Phase 1 is bounded by the product of its expected length and the failure probability in a single generation. Therefore, the total failure probability for the first phase is still of order $2^{-\Omega(n)}$.

In Phase 2 we have $x^* \in \text{SP}$. The goal is now to show that a solution from SP with high probability can only be created if the sampling distribution is sufficiently concentrated around solutions in SP. This in turn makes creating solutions of high Hamming distance from SP, including OPT, very unlikely.

We make this idea precise and consider a point $1^i 0^{n-i} \in \text{SP}$. This search point consists of a prefix of i ones and a suffix of $n-i$ zeros. For a newly constructed solution x we define $P(i) := p_1 + \dots + p_i$ as the expected number of ones in the prefix and $S(i) := (1-p_{i+1}) + \dots + (1-p_n)$ as the expected number of zeros in the suffix. The number of ones in the prefix plus the number of zeros in the suffix yields the number of bits equaling in $1^i 0^{n-i}$ and x , i.e., $n - H(1^i 0^{n-i}, x)$. We call $P(i)$ ($S(i)$) *insufficient* iff $P(i) \leq i - i^{2/3}$ ($S(i) \leq (n-i) - (n-i)^{2/3}$) holds. We now show that with insufficiencies it is very unlikely to create $1^i 0^{n-i}$. As this holds for all i , we conclude that if SP is reached after a certain number of generations, the pheromones do not have insufficiencies, with high probability.

Let $s(i)$ denote the probability of constructing the solution $1^i 0^{n-i}$. We distinguish three cases and apply Chernoff bounds to prove the following implications:

Case 1: $i < n^{2/3}$. Then insufficient $S(i)$ implies $s(i) = 2^{-\Omega(n^{1/3})}$.

Case 2: $i > n - n^{2/3}$. Then insufficient $P(i)$ implies $s(i) = 2^{-\Omega(n^{1/3})}$.

Case 3: $n^{2/3} \leq i \leq n - n^{2/3}$. Then insufficient $P(i)$ and insufficient $S(i)$ each imply $s(i) = 2^{-\Omega(n^{2/9})}$.

We assume that the described insufficiencies do not occur whenever a best-so-far solution $x^* = 1^i 0^{n-i}$ in Phase 2 is accepted. The failure probability is $2^{-\Omega(n^{2/9})}$ for each new best-so-far solution x^* . Generations in between two exchanges of x^* cannot create insufficiencies as $P(i)$ and $S(i)$ can only increase as long as x^* is maintained. Hence, we do not have insufficiencies in Phase 2 for at least $2^{\Omega(n^{2/9})}$ generations with probability at least $1 - 2^{-\Omega(n^{2/9})}$.

Being in Phase 2 without insufficiencies, we show depending on the three cases for the current $x^* = 1^i 0^{n-i}$ that creating an optimal solution has probability $2^{-\Omega(n^{2/9})}$. In the first case, the expected number of zeros in the suffix of x is at least $(n - i) - (n - i)^{2/3}$. By Chernoff bounds, the random number of zeros is at least $(n - i) - 2(n - i)^{2/3}$ with probability at least $1 - 2^{-\Omega(n^{1/3})}$. Along with $i < n^{2/3}$, it follows that then the solution has Hamming distance at most $3n^{2/3}$ from SP. By the definition of SP-Target, this is not enough to reach OPT. The second case is treated analogously. In the third case, the probability of obtaining less than $i - 2i^{2/3}$ ones in the prefix or less than $(n - i) - 2(n - i)^{2/3}$ zeros in the suffix is altogether bounded by $2^{-\Omega(n^{2/9})}$. Then the solution has Hamming distance at most $4n^{2/3}$ from SP, which is also not enough to reach the optimum. This finishes the analysis of the second phase, and, therefore, proves the theorem. \square

The following theorem proves the benefits of local search. Recall that the number of fitness evaluations is at most by a factor of $O(n^2)$ larger than the stated optimization time.

Theorem 2. *Choosing $1/\text{poly}(n) \leq \rho \leq 1/16$, the optimization time of MMAS-LS* on SP-Target is $O(1/\rho)$ with probability $1 - 2^{-\Omega(n)}$. If $\gamma \geq 1$ is chosen large enough but constant, the expected optimization time is also $O(1/\rho)$.*

Proof. The first solution x^* is either 1^n or a global optimum. In the first case all pheromone values increase simultaneously and uniformly from their initial value $1/2$ towards their upper bound $1 - 1/n$. We divide a run into two phases. The first phase ends when either all pheromones become larger than $27/32$ or when a global optimum has been found. The second phase ends when a global optimum has been found, hence it is empty if the first phase ended with an optimum.

We first bound the length of the first phase by the first point of time t^* where all pheromone values exceed $27/32$. Since the pheromone values are at least $\min\{1 - 1/n, 1 - (1/2)(1 - \rho)^t\}$ after t steps (cf. [10]), solving the equation

$$1 - \left(\frac{1}{2}\right)(1 - \rho)^t = 27/32 \iff (1 - \rho)^t = 5/16$$

yields the upper bound

$$t^* \leq \left\lceil \frac{\ln(5/16)}{\ln(1-\rho)} \right\rceil \leq \frac{\ln(16/5)}{\rho} + 1 = O(1/\rho).$$

The assumption $\rho \leq 1/16$ implies that at the last step in the first phase the pheromone value at any bit is within the interval $[25/32, 27/32]$, pessimistically assuming that a global optimum has not been found before. The new search point x then created fulfills the following two properties with probability $1 - O(2^{-n/2400})$:

1. $\frac{3n}{4} \leq |x|_1 \leq \frac{7n}{8}$,
2. $H(x, \text{SP}) \geq n/(\gamma \log n)$.

Using Chernoff bounds with $\delta := 1/25$, the failure probability for the first event is at most $2e^{-(25n/32)(\delta^2/3)} = 2e^{-n/2400}$. To bound the failure probability of the second event, given the first event, we exploit that all pheromone values are equal. Therefore, if we know that $|x|_1 = k$ then x is uniform over all search points with k ones. Since the number of search points with k ones is monotone decreasing for $3n/4 \leq k \leq 7n/8$, we only consider search points with $k = 7n/8$ ones as a worst case. The number of such search points is $\binom{n}{n/8}$, and the number of search points of Hamming distance at most $m := n/(\gamma \log n)$ from SP is at most $m \cdot \binom{n}{m}$. Altogether, the probability of $H(x, \text{SP}) \leq m$ given that $3n/4 \leq |x|_1 \leq 7n/8$ is bounded from above by

$$\frac{m \binom{n}{m}}{\binom{n}{n/8}} \leq \frac{m \left(\frac{en}{m}\right)^m}{\left(\frac{n}{n/8}\right)^{n/8}} \leq m \cdot 2^{o(n)} \cdot 8^{-n/8}.$$

The last expression is even $O(2^{-n/8})$. Altogether, the sum of the failure probabilities is $O(2^{-n/2400})$ as suggested, and the first statement follows.

For the second statement we estimate the time in the second phase, provided that the first phase has been unsuccessful. Using [10] and $\rho = 1/\text{poly}(n)$, the time to reach the pheromone bound is $O((\log n)/\rho) = \text{poly}(n)$, or an optimum is created anyway. With all pheromones at the upper bound, the solution construction process equals a standard mutation of 1^n , i. e., flipping each bit in 1^n independently with probability $1/n$. Flipping the first m bits results in a global optimum as $0^m 1^{n-m}$ has Hamming distance at least m to $1^i 0^{n-i}$ for any i . The probability of creating $0^m 1^{n-m}$ in a standard mutation is at least

$$\left(\frac{1}{n}\right)^{n/(\gamma \log n)} \left(1 - \frac{1}{n}\right)^{n-n/(\gamma \log n)} \geq e^{-1} \cdot 2^{-n/\gamma}.$$

This means that the expected time in the second phase is $O(\text{poly}(n)2^{n/\gamma})$. Using that the first phase is unsuccessful only with probability $O(2^{-n/2400})$ and applying the law of total probability, the expected optimization time altogether is $O(1/\rho) + O(2^{-n/2400}) \cdot O(\text{poly}(n)2^{n/\gamma})$. The latter is $O(1/\rho)$ for $\gamma > 2400$, which proves the second statement. \square

5 Drawbacks of Combining ACO and Local Search

Similarly to the function SP-Target, we design another function SP-Trap (short path with trap) where local search is detrimental, using ideas from Section 3. We take over the path with increasing fitness, $SP = \{1^i 0^{n-i} \mid 0 \leq i \leq n\}$, but in contrast to SP-Target, the former region of global optima now becomes a trap, $TRAP = \{x \mid |x|_1 \geq (3/4) \cdot n \wedge H(x, SP) \geq n/\log n\}$. The unique global optimum is placed within distance 2 from the local optimum: $OPT = \{0^2 1^{n-2}\}$. This ensures that local search climbing the path SP cannot reach the global optimum. All remaining search points give hints to reach the start of the path.

$$SP-Trap(x) := \begin{cases} |x|_0 & x \notin (SP \cup TRAP \cup OPT) \\ n+i & x = 1^i 0^{n-i} \in SP \\ 3n & x \in TRAP \\ 4n & x \in OPT. \end{cases}$$

The function SP-Trap is sketched in Figure 4.

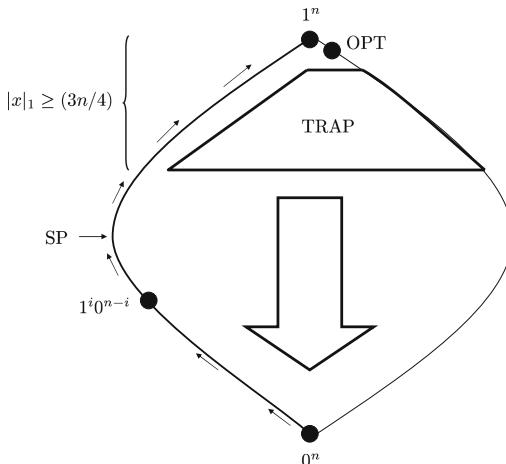


Fig. 4. Illustration of the Boolean hypercube and the function SP-Trap. Arrows indicate gradients of increasing fitness.

In the remainder of this section, we prove that MMAS* is efficient on SP-Trap while MMAS-LS* fails dramatically. Tuning the definition of SP-Trap, we could also extend the following theorem by a polynomial bound on the expected optimization time. We refrain from such modifications to illustrate the main effects.

Theorem 3. *Choosing $\rho = 1/\text{poly}(n)$, the optimization time of MMAS* on SP-Trap is $O((n \log n)/\rho + n^3)$ with probability $1 - 2^{-\Omega(n^{2/9})}$.*

Proof. By the argumentation from Theorem 1, the probability that a solution in TRAP is produced within $O(n^3)$ generations is at most $2^{-\Omega(n^{2/9})}$.

Under the assumption that TRAP is never reached until the global optimum is found, MMAS* behaves equally on SP-Trap and a modified function where $x \in \text{TRAP}$ receives fitness $|x|_0$. We apply fitness-level arguments from [9,10] to estimate the expected optimization time on the latter, easier function. The number of fitness levels is $O(n)$. On every fitness level, the number of generations until either all pheromones are frozen or the current best-so-far solution has improved is bounded by $O((\log n)/\rho)$ with probability 1 [10]. We pessimistically assume that an improvement can only happen once all pheromones have been frozen. Then the optimization time is bounded by $O((n \log n)/\rho)$ plus the sum of waiting times for improvements on all fitness levels. Showing that the latter quantity is bounded by $O(n^3)$ with probability $1 - 2^{-\Omega(n)}$ completes the proof.

After freezing, the solution construction process equals a standard mutation of the best-so-far solution x^* . The probability for an improvement from $x^* = 1^n$ is at least $1/(en^2)$. For all other $x^* \notin \text{TRAP}$, there is always a better Hamming neighbor, hence the probability for an improvement is at least $1/(en)$. Together, the expected waiting times for improvements on all fitness levels sum up to $en^2 + O(n) \cdot en = O(n^2)$. By Markov's inequality the probability of waiting more than cn^2 steps is at most $1/2$ for a suitable constant $c > 0$. Hence, the probability that more than n independent phases of length cn^2 are needed is bounded by $2^{-\Omega(n)}$. Therefore, the bound $O(n^3)$ holds with probability $1 - 2^{-\Omega(n)}$. \square

Theorem 4. *Choosing $1/\text{poly}(n) \leq \rho \leq 1/16$, the optimization time of MMAS-LS* on SP-Trap is $2^{\Omega(n)}$ with probability $1 - 2^{-\Omega(n)}$.*

Proof. We follow the lines of the proof of Theorem 2. As long as $\text{OPT} = 0^2 1^{n-2}$ is not created, the behavior of MMAS-LS* on SP-Trap and SP-Target is identical. Reconsider the first phase described in the proof of Theorem 2 (with the former OPT replaced by TRAP) and denote by $P := p_1 + \dots + p_n$ the sum of probabilities of sampling ones over all bits. Throughout the phase, $P \leq 27n/32$, hence the probability of sampling at least $n - 2$ ones, which is necessary to reach OPT, is $2^{-\Omega(n)}$ according to Chernoff bounds.

With probability $1 - 2^{-\Omega(n)}$, the first best-so-far solution 1^n is replaced by some $x^{**} \in \text{TRAP}$ where $|x^{**}|_1 \leq 7n/8$ when the first phase is ended. Due to strict selection, x^{**} then can only be replaced if OPT is created. The latter has probability $2^{-\Omega(n)}$ for the following reasons: the P -value is at most $27n/32 \leq 7n/8$ when x^{**} is accepted. Hence, following the argumentation from the proof of Lemma 1, the P -value will not exceed $7n/8$ unless x^{**} is replaced. With a P -value of at most $7n/8$, creating OPT has probability $2^{-\Omega(n)}$. \square

6 Conclusions

We have investigated the combination of ACO and local search from a theoretical point of view and pointed out how this combination can influence the search process. In particular, we have rigorously shown that the combination of both

methods can outperform ACO algorithms not using local search procedures. Furthermore, we have proven that the combination of ACO and local search may mislead the search process. Our results are a further step in the runtime analysis of ACO and its hybridizations. In the future, the analysis of ACO hybridizations using more than a single ant on more complicated problems would be desirable.

Acknowledgment. Thanks to the participants of SLS 2007 for stimulating discussions on the issues investigated in this paper.

References

1. Dorigo, M., Stützle, T.: *Ant Colony Optimization*. MIT Press, Cambridge (2004)
2. Hoos, H.H., Stützle, T.: *Stochastic Local Search: Foundations & Applications*. Elsevier/Morgan Kaufmann (2004)
3. Levine, J., Ducatelle, F.: Ant colony optimisation and local search for bin packing and cutting stock problems. *Journal of the Operational Research Society* (2004)
4. Balaprakash, P., Birattari, M., Stützle, T., Dorigo, M.: Incremental local search in ant colony optimization: Why it fails for the quadratic assignment problem. In: Proc. of ANTS Workshop 2006, pp. 156–166 (2006)
5. Merkle, D., Middendorf, M.: Modeling the dynamics of ant colony optimization. *Evolutionary Computation* 10, 235–262 (2002)
6. Gutjahr, W.J.: On the finite-time dynamics of ant colony optimization. *Methodology and Computing in Applied Probability* 8, 105–133 (2006)
7. Stützle, T., Hoos, H.H.: MAX-MIN ant system. *Journal of Future Generation Computer Systems* 16, 889–914 (2000)
8. Doerr, B., Neumann, F., Sudholt, D., Witt, C.: On the runtime analysis of the 1-ANT ACO algorithm. In: Proc. of GECCO 2007, pp. 33–40. ACM, New York (2007)
9. Gutjahr, W.J., Sebastiani, G.: Runtime analysis of ant colony optimization with best-so-far reinforcement. *Methodology and Computing in Applied Probability* (to appear, 2008)
10. Neumann, F., Sudholt, D., Witt, C.: Comparing variants of MMAS ACO algorithms on pseudo-Boolean functions. In: Stützle, T., Birattari, M., H. Hoos, H. (eds.) SLS 2007. LNCS, vol. 4638, pp. 61–75. Springer, Heidelberg (2007)
11. Neumann, F., Witt, C.: Runtime analysis of a simple ant colony optimization algorithm. In: Asano, T. (ed.) ISAAC 2006. LNCS, vol. 4288, pp. 618–627. Springer, Heidelberg (2006)
12. Gutjahr, W.J.: First steps to the runtime complexity analysis of ant colony optimization. *Computers and Operations Research* 35(9), 2711–2727 (2008)

Simple Dynamic Particle Swarms without Velocity

Jorge Peña

Institut de Mathématiques Appliquées (IMA), Université de Lausanne, Switzerland
jorge.pena@unil.ch

Abstract. The standard particle swarm optimiser uses update rules including both multiplicative randomness and velocity. In this paper, we look into a general particle swarm model that removes these two features, and study it mathematically. We derive the recursions and fixed points for the first four moments of the sampling distribution, and analyse the transient behaviour of the mean and the variance. Then we define actual instances of the algorithm by coupling the general update rule with specific recombination operators, and empirically test their optimisation efficiency.

1 Introduction

The standard Particle Swarm Optimiser (PSO) [1,2] uses an update rule in the form of a set of second order difference equations including additive and multiplicative stochasticity. In an effort to obtain particle swarms with reduced computational complexity or which are more suitable to mathematical analysis, several researchers have proposed simpler update rules that, intentionally or not, remove multiplicative randomness from their equations. In previous work [3], we provided a general framework for particle swarms ruled by second order difference equations with additive stochasticity only, and showed how some PSOs already proposed in the literature, as well as new variants, can be derived from this general model by the definition of particular *recombination operators*. This paper focuses on the inertialess or velocity-free case, for which update rules become first order difference equations with the result that the model becomes even simpler and more amenable to theoretical analysis.

The paper is organised as follows. Section 2 introduces the concept of simple dynamic particle swarms without velocity. Section 3 presents a detailed mathematical study (via moment analysis) of the sampling distribution of these PSOs. In Section 4 the optimisation efficiency of three different instances of the general model is tested. Conclusions are drawn in Section 5.

2 Particle Swarms

2.1 The Standard Particle Swarm

The position update rules for the Standard PSO are given by difference equations of the general form:

$$x_{t+1} = x_t + w(x_t - x_{t-1}) + \sum_{k=1}^K \frac{\phi}{K} u_k (p_k - x_t), \quad (1)$$

where p_k is the personal best of the k -th *informer* (there are K informers in total), $u_k \sim U[0, 1]$ is a random variable taken from a continuous uniform distribution in $[0, 1]$, w is the inertia weight, and ϕ is the acceleration coefficient. Informers can be given by the standard best-of-neighbourhood (BN), the fully informed (FI) [4], or any other model of influence. For the general case $w \neq 0$, the update rule is a second order equation including the velocity term $v_t = x_t - x_{t-1}$. Since random numbers multiply both the constant term p_k and the variable x_t , Eq. 1 is said to have both additive and multiplicative stochasticity [3].

2.2 Particle Swarms with Additive Stochasticity and Different Recombination Operators

A general particle swarm model that relies only on additive stochasticity has been recently proposed in [3]. In this model, the positions of the particles are updated according to

$$x_{t+1} = x_t + w(x_t - x_{t-1}) + \alpha(q - x_t), \quad (2)$$

where α is a constant *acceleration coefficient* and q a random variable derived from a *recombination operator* acting over the set of informers' personal bests. Suitable recombination operators are, for instance, linear stochastic combinations of the personal bests values or, more generally, probability distributions whose parameters are functions of these personal bests. With the corresponding setting of w , α and q , the presented model can be shown to recover some existing velocity-based particle swarms such as Kennedy's Gaussian-Dynamic Particle Swarm [5], Poli et al.'s Simpler PSO [6] and Peña et al.'s PSO-DR [7].

Table 1. Recombination Operators

Recombination Operator	Definition (for $K = 2$)
Standard	$q := \frac{u_1 p_1 + u_2 p_2}{u_1 + u_2}$
Rectangular	$q := up_1 + (1-u)p_2$
Discrete	$q := \eta_d p_1 + (1-\eta_d)p_2$

For the empirical studies presented in this paper, we focus on the three recombination operators whose definitions are given in Table 1 for the simplest case of $K = 2$.¹ The two informers could be, for instance, the self and the best neighbour in a BN particle swarm, or the left and right neighbours in a FIPS using a Ring topology without self-influence. In the following, we will respectively denote the personal bests of these two informers by p_1 and p_2 .

¹ Notice, however, that generalisation to a greater number of informers is possible and that the analytical results presented in the next sections do not depend on the specific number of informers or the particular recombination operator used.

The *Standard*, *Rectangular* and *Discrete* recombination operators consist of different linear stochastic combinations of the informers' personal bests. In Table 1 u_1 , u_2 and u are random variables taken from a continuous uniform distribution in $[0, 1]$ and $\eta_d \sim U\{0, 1\}$ is a random variable distributed according to a discrete uniform distribution in $\{0, 1\}$. Standard recombination can be shown to be implicitly implemented in the update equation of the Standard PSO [6,3]. Rectangular and Discrete recombination recover Poli et al.'s Simpler PSO [6] and Peña et al.'s PSO-DR [7], which are particle swarms that, despite their simplicity, have been shown to be competitive to Standard PSO in a variety of problems [7,6,8].

2.3 Particle Swarms without Velocity

Although particle swarms are essentially velocity-based algorithms, there have been proposals for velocity-free particle swarms in the literature. The most well known of these models are probably Kennedy's "bare bones" algorithms [9], in which position update rules are replaced by sampling from a probability distribution. The update rule of these algorithms can be understood as a zero-order difference equation, constituting a particular (static) case of the general model of Eq. 2, obtained with $w = 0$ and $\alpha = 1$. In this paper, we are interested in exploring simple velocity-free but still dynamic algorithms, derived from Eq. 2 with $w = 0$ and $\alpha \neq 0$. The update rule for these Simple Dynamic Particle Swarms (SDPS) is thus given by:

$$x_{t+1} = x_t + \alpha(q - x_t). \quad (3)$$

Bratton and Blackwell [8] studied empirically a simplified recombinant PSO that constitutes a particular case of Eq. 3 with Discrete recombination. Here, we analyse the more general model of Eq. 3 with any well defined recombination operator and empirically study the performance of SDPSs with the Standard, Rectangular and Discrete operators.

3 Mathematical Study

3.1 Recursions for the First Four Moments of the Sampling Distribution During Stagnation

In order to formally analyse SDPSs, we use the moment analysis introduced by Poli and Broomhead [10] to see how the sampling distribution of a PSO behaves during the stagnation phase. Let us firstly rewrite Eq. 3 as

$$x_{t+1} = (1 - \alpha)x_t + \alpha q. \quad (4)$$

Our objective is to calculate a recursion for the n -th central moment μ_n of x_{t+1} . To do this, first we calculate the raw moments μ'_n and then convert them to central moments. A difference equation for the n -th raw moment can be easily

derived by taking the n -th power of Eq. 4 and then applying the expectation operator ($\langle \cdot \rangle$):

$$\mu'_n(x_{t+1}) = \langle (x_{t+1})^n \rangle = \langle ((1 - \alpha)x_t + \alpha q)^n \rangle.$$

By making use of the binomial theorem, the linearity of the expectation operator, and assuming that x_t and q are statistically independent, we can write:

$$\begin{aligned} \mu'_n(x_{t+1}) &= \left\langle \sum_{k=0}^n \binom{n}{k} ((1 - \alpha)x_t)^k (\alpha q)^{n-k} \right\rangle \\ &= \left\langle \sum_{k=0}^n \binom{n}{k} (1 - \alpha)^k \alpha^{n-k} (x_t)^k q^{n-k} \right\rangle \\ &= \sum_{k=0}^n \binom{n}{k} (1 - \alpha)^k \alpha^{n-k} \langle (x_t)^k q^{n-k} \rangle \\ &= \sum_{k=0}^n \binom{n}{k} (1 - \alpha)^k \alpha^{n-k} \langle (x_t)^k \rangle \langle q^{n-k} \rangle \\ &= \sum_{k=0}^n \binom{n}{k} (1 - \alpha)^k \alpha^{n-k} \mu'_k(x_t) \mu'_{n-k}(q). \end{aligned}$$

The recursions for the central moments can be calculated using the binomial transform:

$$\mu_n(x_t) = \sum_{k=0}^n \binom{n}{k} (-1)^{n-k} \mu'_k(x_t) (\mu'_1(x_t))^{n-k}.$$

From this, the recursions for the first three central moments can be shown to be given by:

$$\mu_n(x_{t+1}) = (1 - \alpha)^n \mu_n(x_t) + \alpha^n \mu_n(q), \quad (5)$$

for $n = 1, 2, 3$, while the recursion for the fourth central moment is given by:

$$\mu_4(x_{t+1}) = (1 - \alpha)^4 \mu_4(x_t) + \alpha^4 \mu_4(q) + 6(1 - \alpha)^2 \alpha^2 \mu_2(x_t) \mu_2(q). \quad (6)$$

We are particularly interested in the mean, variance, skewness and kurtosis of the sampling distribution during stagnation. The mean and variance are equal to the first and second central moments (μ_1 and μ_2). The skewness γ_1 is a measure of the degree of asymmetry of a distribution. It is negative if the left tail is more pronounced than the right tail, and positive otherwise. Among different alternative definitions, we chose to use $\gamma_1 = \mu_3/\mu_2^{3/2}$. The kurtosis² γ_2 is a

² We follow standard practice and use the term “kurtosis” for referring to the *excess* kurtosis. The kurtosis of the normal distribution is equal to 0. A *leptokurtic* distribution has positive kurtosis, which means that it is more peaked and has fatter tails than the normal distribution. Conversely, a *platykurtic* distribution has negative kurtosis, which means that it is less peaked and has thinner tails than the normal distribution. Finally, a distribution with zero kurtosis (e.g. the normal distribution) is said to be *mesokurtic*.

measure of the degree of peakedness of a distribution as well as of the thickness of its tails, and is defined as $\gamma_2 = \mu_4/\mu_2^2 - 3$.

3.2 Fixed Points and Stability Analysis

Let us now calculate the fixed points of the recursions for the first four moments of the sampling distribution of SDPSs during stagnation. Let $\mu_n(x)^*$ be the fixed point of the recursion for $\mu_n(x_t)$. Letting $\mu_n(x_{t+1}) = \mu_n(x_t) = \mu_n(x)^*$ in Eq. 5, we obtain after little algebra

$$\mu_n(x)^* = \frac{\alpha^n}{1 - (1 - \alpha)^n} \mu_n(q),$$

which for $n = 1, 2, 3$, reduces to:

$$\mu_1(x)^* = \mu_1(q), \quad (7)$$

$$\mu_2(x)^* = \frac{\alpha}{2 - \alpha} \mu_2(q), \quad (8)$$

and

$$\mu_3(x)^* = \frac{\alpha^2}{3 - 3\alpha + \alpha^2} \mu_3(q). \quad (9)$$

To calculate the fixed point of $\mu_4(x_t)$ we assume that $\mu_2(x_t)$ has reached its fixed point $\mu_2(x)^*$. Performing the substitution in Eq. 6 and letting $\mu_4(x_{t+1}) = \mu_4(x_t) = \mu_4(x)^*$, we get after some simplifications:

$$\mu_4(x)^* = \frac{\alpha^3(2 - \alpha)\mu_4(q) + 6\alpha^2(1 - \alpha)^2(\mu_2(q))^2}{(2 - \alpha)^2(2 - 2\alpha + \alpha^2)}. \quad (10)$$

The fixed points of the skewness and the kurtosis can be calculated from Eq. 8, 9 and 10, and shown to be given by

$$\gamma_1(x)^* = \frac{(2 - \alpha)^{3/2}}{3 - 3\alpha + \alpha^2} \gamma_1(q) \quad (11)$$

and

$$\gamma_2(x)^* = \frac{\alpha(2 - \alpha)}{2 - 2\alpha + \alpha^2} \gamma_2(q), \quad (12)$$

where $\gamma_1(q)$ and $\gamma_2(q)$ are, respectively, the skewness and the kurtosis of q .

It is easy to show (by simple inspection or by a complete eigenvalue analysis) that the fixed points of the first four central moments (and thus also of $\gamma_1(x)^*$ and $\gamma_2(x)^*$) are stable if $|1 - \alpha| < 1$. This means that the regions of *order-1*, *-2*, *-3* and *-4* stability³ coincide and are equal to $0 < \alpha < 2$.

It can be seen from Eq. 7, 8, 11 and 12 that the equilibrium values of the mean, variance, skewness and kurtosis of the sampling distribution are proportional to the respective normalised moments of q . The proportional factor is

³ We follow Poli [10] and say that a sampling distribution is *order-1* stable if its first moment is stable, *order-2* stable if its first two moments are stable, etc.

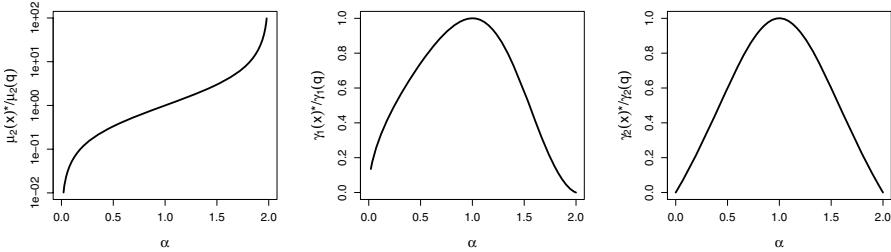


Fig. 1. Normalised equilibrium value of the variance (*left*), skewness (*centre*) and kurtosis (*right*) of the sampling distribution as a function of α (See Eq. 8, 11 and 12). Notice the logarithmic scale of the y axis in the *left* subfigure.

equal to 1 for the mean (i.e. the fixed point of the mean of x is the mean of q) and a function of α for the other three moments. Fig. 1 shows plots of these proportional factors. Notice that for $\alpha = 1$ all proportional factors are equal to 1. This is not surprising, since for $\alpha = 1$ the update rule reduces to the zero-order equation $x_{t+1} = q$, and the sampling distribution is thus equal to the probability function defining q . For $\alpha < 1$, the equilibrium value of the variance of the sampling distribution is smaller than the variance of q , whereas for $\alpha > 1$ the opposite happens. Observe the rapid growing (resp. falling) of the variance as α approaches 2 (resp. 0). Also notice that moving α from 1 towards 0 or 2 makes the skewness and the kurtosis tend to zero. Since the proportional factors for the skewness and the kurtosis are non-negative values less than one, the sampling distribution is always right-skewed (resp. left-skewed) if q is right-skewed (resp. left-skewed) and leptokurtic (resp. platykurtic) if q is leptokurtic (resp. platykurtic), though less pronouncedly, depending on α .

3.3 Transient Behaviour of $\mu_1(x_t)$ and $\mu_2(x_t)$

In this section we analyse the transient behaviour of the first two moments of the sampling distribution. Analytic solutions for the recursions for these moments are easy to obtain, since the recursions are particular cases of first order linear difference equations. The general solution of Eq. 5 is given by

$$\mu_n(x_t) = \mu_n(x)^* + (\mu_n(x_0) - \mu_n(x)^*) (1 - \alpha)^{nt},$$

from which particular solutions for the cases $n = 1$ and $n = 2$ can be obtained. After the proper substitutions, we get

$$\mu_1(x_t) = \mu_1(q) + (\mu_n(x_0) - \mu_1(q)) (1 - \alpha)^t,$$

and

$$\mu_2(x_t) = \frac{\alpha \mu_2(q)}{2 - \alpha} + \left(\mu_n(x_0) - \frac{\alpha \mu_2(q)}{2 - \alpha} \right) (1 - \alpha)^{2t}.$$

In order to analyse the transient behaviour of these equations we borrow the concept of *step response* from control theory and look into the dynamics of the

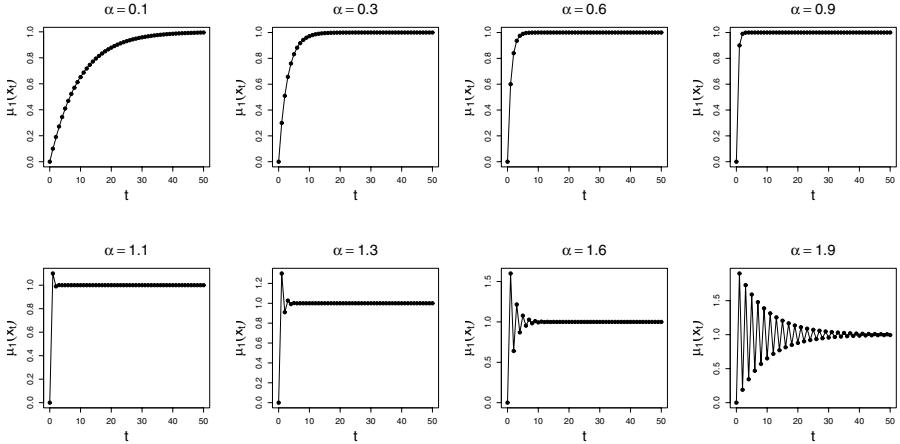


Fig. 2. Transient behaviour of the mean for $\mu_1(x)^* = 1$ and $\mu_1(x_0) = 0$

system when $\mu_n(x_0) = 0$ and $\mu_n(x)^* = 1$. Fig. 2 and 3 show the numerical results we obtained for different values of α . As it can be seen from the figures (and easily proved analytically), the step response of the mean exhibits no oscillations for $\alpha < 1$ and oscillates for $\alpha > 1$. In this last case, the percentage overshoot is proportional to α , and its maximum value is equal to $\alpha - 1$ (at $t = 1$). The step response of the variance exhibits no overshoot for the whole range of α . Finally, the duration of the transient state is inversely proportional to $|1 - \alpha|$ for the mean and to $(1 - \alpha)^2$ for the variance.

The transient behaviour of the mean of the sampling distribution has been posited to be important for the performance of PSOs, since oscillations of the mean allow for an amount of “extrapolation” that could be beneficial in some functions [6]. This feature, present in velocity-based particle swarms with standard stochasticity and with only additive stochasticity [3], is thus also shared by the simpler velocity-free algorithms studied in this paper.

3.4 Sampling Distributions of Particular SDPSs

Fig. 4 depicts the histograms of points tested in one million iterations by a particle ruled by Eq. 3 and Standard, Rectangular and Discrete recombination, for $x_0 = 0$, $p_1 = -1$ and $p_2 = 1$, and different values of α . Although the histograms were constructed considering points from consecutive time steps and include transient points as well, they represent good approximations of the steady state behaviour of the sampling distribution, since the number of tested iterations is far larger than the duration of the transients.

The mean and skewness of the random variable q are the same for the three recombination operators tested ($\mu_1(q) = (p_1 + p_2)/2$, $\gamma_1(q) = 0$). Their variance, fourth moment and kurtosis differ. These values are listed in Table 2. Notice that the three recombination operators give rise to platykurtic distributions since $\gamma_2(q) < 0$. Standard recombination has the smallest variance ($3/4 - \log(2) \approx$

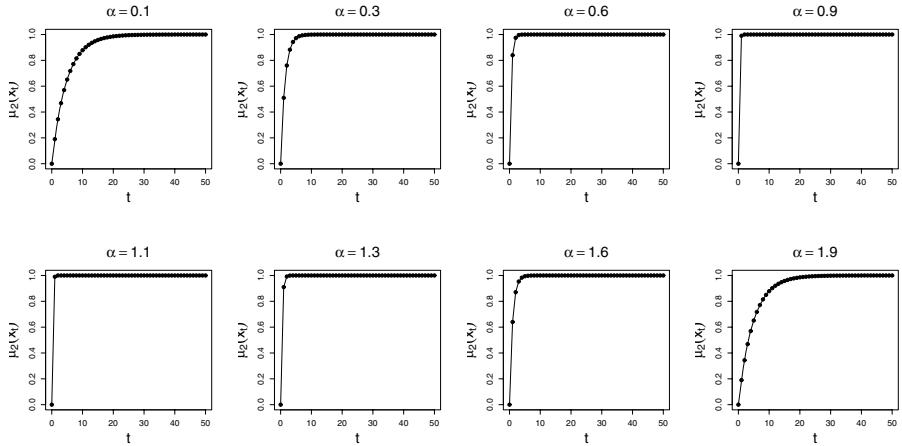


Fig. 3. Transient behaviour of the variance for $\mu_2(x)^* = 1$ and $\mu_2(x_0)$

Table 2. Statistical characterization of different recombination operators

Recombination	$\mu_2(q)$	$\mu_4(q)$	$\gamma_2(q)$
Standard	$(3/4 - \log 2)(p_1 - p_2)^2$	$(\frac{17}{48} - \frac{1}{2}\log 2)(p_1 - p_2)^4$	-0.650834
Rectangular	$\frac{1}{12}(p_1 - p_2)^2$	$\frac{1}{80}(p_1 - p_2)^4$	-1.2
Discrete	$\frac{1}{4}(p_1 - p_2)^2$	$\frac{1}{16}(p_1 - p_2)^4$	-2

0.057) and the least negative kurtosis, whereas Discrete recombination has the largest variance and the most negative kurtosis.

The effect of α in the empirical sampling distributions is well predicted by the equations derived previously. For $\alpha = 1$, the sampling distribution during stagnation is identical to the distribution of q . Incrementing α makes the probability density functions grow wider and more peaked. They are always symmetric, and centred midway between p_1 and p_2 . In the limit when $\alpha \rightarrow 2$ the distributions are practically very wide normal distributions. This is what Eq. 7, 8, 11 and 12, and Fig. 1 predict for the sampling distributions of SDPSs with recombination operators leading to symmetric, platykurtic distributions centred in $(p_1 + p_2)/2$. It is interesting to notice that the sampling distributions of all recombination operators are in general unimodal, except for Discrete that produces multimodal distributions until somewhat large values of α (somewhere between 1.6 and 1.8).

4 Experimental Results

In this section, we empirically study the optimisation efficiency of SDPSs with Standard, Rectangular and Discrete recombination. The experimental setup follows closely that proposed by Bratton and Kennedy [2] as a standard for comparing different PSOs. All tested algorithms used populations of 50 particles, a Ring topology and a FI model without self-influence. We used FI instead of

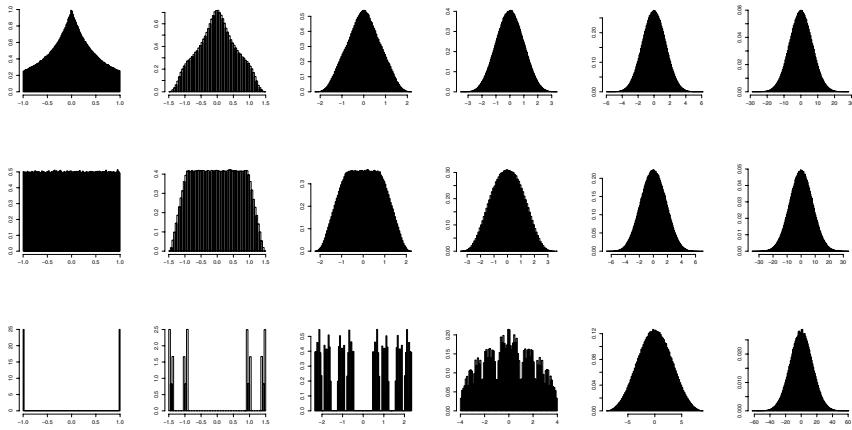


Fig. 4. Empirical sampling distributions of SDPSs with (from *top to bottom*) Standard, Rectangular and Discrete recombination operators for (from *left to right*) $\alpha \in \{1.0, 1.2, 1.4, 1.6, 1.8, 1.99\}$

BN particle swarms because of the better performance of the former in previous experiments [3].

Algorithms were run on a subset of the suite of benchmark functions proposed by Bratton and Kennedy in [2]. Functions belong to three distinguishable groups: $f_1 - f_3$ are unimodal, $f_4 - f_9$ are complex high-dimensional with many local minima, and $f_{10} - f_{12}$ are low-dimensional with few local minima. In order to remove any centrist bias, both the *region scaling* and the *center offset* techniques [2] were used for all functions, except for f_4 where only region scaling was applied. The center offset technique was implemented by shifting the function by a vector of uniform random values in $U[-0.25l, 0.25l]$ for each run, being l the size of the search space in each dimension. Particles flying out of the feasible bounds were not evaluated. Finally, the error $|f(x) - f(x^*)|$ found after 300,000 function evaluations was used as the measure for algorithm performance (or *fitness*), where $f(x^*)$ is the value of the objective function at the global minimum. Values less than 10^{-8} were rounded to 10^{-8} . For a description of the functions and the dimensionality, feasible bounds, location of the optimum and initialization ranges for each function, the reader is invited to refer to [2].

The performance as function of α for each of the three algorithms is shown in Fig. 5. Notice that the performance plots for Standard and Rectangular recombination almost overlap. This result is not unexpected given the somewhat similar sampling distributions produced by the two recombination operators. For the high-dimensional functions, the optimal region for Standard recombination is slightly shifted to the right with respect to that for Rectangular recombination. This could be explained from the fact that Standard recombination has smaller variance and kurtosis than Rectangular recombination, and thus a larger α is necessary when using the Standard operator in order to obtain the necessary levels of exploration needed to search in these high-dimensional spaces. For

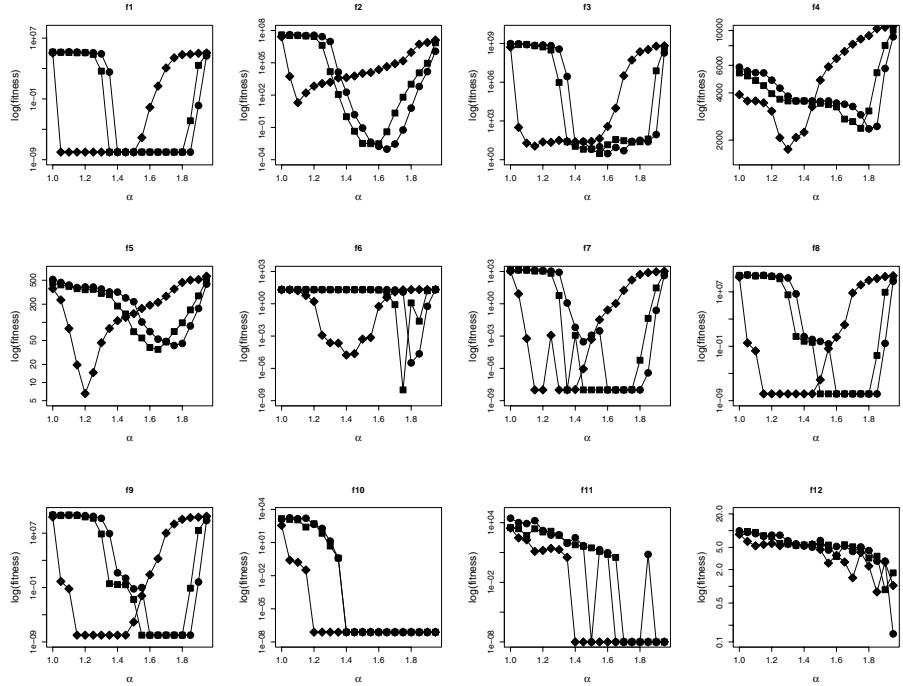


Fig. 5. Mean performance over 25 runs of SDPSs and Standard (*circles*), Rectangular (*squares*) and Discrete (*diamonds*) recombination, as a function of α

low-dimensional functions ($f_{10}-f_{12}$), higher values of α (thus wider and peaked distributions with large extrapolation) are favoured regardless of the recombination operator. Nevertheless, for the multimodal functions (f_1-f_9) the optimal region of α for Standard and Rectangular recombination ($1.5 < \alpha < 1.8$) is in general different from that for Discrete recombination ($1.1 < \alpha < 1.4$). This reveals two different heuristics for exploring the search space. On the one hand, normal-like distributions with important overshoots and settling times of the mean, and, on the other, multimodal distributions with low kurtosis and negligible extrapolation.

5 Conclusions

In this paper, the sampling distribution and the optimisation efficiency of simple dynamic particle swarms without neither multiplicative stochasticity nor velocity was analytically and empirically studied. The dynamic equations for the first four moments of the sampling distribution were derived and their fixed points calculated. Making use of these mathematical tools, the effect of the acceleration coefficient in the search behaviour of these algorithms was analysed. Finally, the optimisation efficiency of these algorithms was empirically tested over a set of common benchmark functions.

When optimising high-dimensional multimodal functions using SDPSs with Standard and Rectangular recombination, the optimal region of parameter α gives rise to unimodal sampling distributions centred midway between the personal bests of the informers, and relevant amounts of oscillations of the mean. These characteristics are shared by the Standard PSO and other ‘traditional’ particle swarms, both velocity-based and velocity-free. When using Discrete recombination, the optimal region of α produce sampling distributions that are multimodal, making the particle search focus on the regions near the locations of the personal bests. Additionally, extrapolation is almost negligible in these particle swarms. The competitiveness of these discretely recombined SDPSs, and their somewhat different search strategy, thus challenges some of the particle swarm lore, particularly the importance of extrapolation and focusing the search around a ‘social centre’.

Acknowledgements. This work is funded by the Future and Emerging Technologies programme IST-STREP of the European Community, under grant number IST-034632 (PERPLEXUS). The author gratefully acknowledge this financial support and thanks Isis Fuchs and the three anonymous reviewers for their comments on this paper.

References

1. Kennedy, J., Eberhart, R.C.: Particle swarm optimization. In: Proceedings of the IEEE International Conference on Neural Networks IV, pp. 1942–1948. IEEE Press, Piscataway (1995)
2. Bratton, D., Kennedy, J.: Defining a Standard for Particle Swarm Optimization. In: Proceedings of the IEEE Swarm Intelligence Symposium, pp. 120–127 (2007)
3. Peña, J.: Theoretical and Empirical Study of Particle Swarms with Additive Stochasticity and Different Recombination Operators. In: Proceedings of the 2008 GECCO Conference on Genetic and Evolutionary Computation (to appear, 2008)
4. Mendes, R., Kennedy, J., Neves, J.: The Fully Informed Particle Swarm: Simpler, Maybe Better. *IEEE Trans. Evolutionary Computation* 8, 204–210 (2004)
5. Kennedy, J.: Dynamic-probabilistic Particle Swarms. In: Proceedings of the 2005 Conference on Genetic and Evolutionary Computation, pp. 201–207. ACM Press, New York (2005)
6. Poli, R., Bratton, D., Blackwell, T., Kennedy, J.: Theoretical Derivation, Analysis and Empirical Evaluation of a Simpler Particle Swarm Optimiser. In: Proceedings of the IEEE Congress on Evolutionary Computation, pp. 1955–1962 (2007)
7. Peña, J., Upegui, A., Sanchez, E.: Particle Swarm Optimization with Discrete Recombination: An Online Optimizer for Evolvable Hardware. In: Proceedings of the First NASA/ESA Conference on Adaptive Hardware and Systems, AHS, pp. 163–170. IEEE Computer Society, Los Alamitos (2006)
8. Bratton, D., Blackwell, T.: A Simplified Recombinant PSO. *Journal of Artificial Evolution and Applications*, Article ID 654184 (2008)
9. Kennedy, J.: Bare Bones Particle Swarms. In: Proceedings of the 2003 IEEE Swarm Intelligence Symposium, pp. 80–87 (2003)
10. Poli, R., Broomhead, D.: Exact Analysis of the Sampling Distribution for the Canonical Particle Swarm Optimiser and its Convergence During Stagnation. In: Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation, pp. 134–141. ACM Press, New York (2007)

Swarming in a Virtual World: A PSO Approach to Virtual Camera Composition

Luca Di Gaspero¹, Andrea Ermetici², and Roberto Ranon²

¹ DIEGM, University of Udine, Udine, Italy

1.digaspero@uniud.it

² DIMI, University of Udine, Udine, Italy

roberto.ranon@dimi.uniud.it

Abstract. Camera placement in 3D scenes is a relevant issue in most 3D graphics interactive application, such as videogames, data visualization, and virtual tours. Virtual Camera Composition (VCC) consists in automatically positioning a camera in a virtual world, such that the resulting image satisfies a set of visual cinematographic properties [1]. We propose a Particle Swarm algorithm to solve the problem, which exhibits superior performances w.r.t. other approaches. The algorithm has been tested on a set of image descriptions on a complex 3D model.

1 Introduction

In 3D graphics interactive applications, such as videogames, data visualization, and virtual tours, user see the virtual environment through the “eye” of a virtual camera. Proper camera placement (and also control) is therefore fundamental for the user to understand the scene and, ultimately, effectively use the application. For example, in 3D data visualization, bad camera placements could cause the user to miss important details (e.g., because they are hidden behind objects or out of camera reach) with the risk of making wrong assumptions on the data themselves.

In most current 3D applications (e.g., 3D modelers), users directly position the virtual camera using a input device, such as a mouse, through a tedious and time-consuming process requiring a succession of “place the camera” and “check the result” operations [1]. In recent years, some researchers (e.g., [1,2]) have come up with methods to automatically position the camera that are inspired by how human cinematographers approach the problem of staging a camera to compose an image that highlights the important subjects in a scene. More particularly, the Virtual Camera Composition (VCC) problem consists in positioning a camera in a virtual world, such that the resulting image satisfies a set of visual cinematographic properties [1], e.g. subjects’ size and location in the obtained image (frame). VCC approaches aims at relieving the user from directly manipulating the camera, and typically model the VCC problem as an optimization or constraint-based system (some approaches use both) where image properties are represented as constraints or objective functions. A range of different solving techniques (which we review in Section 2) have been explored in the past,

but generating effective results in real-time (or near-real time) remains an issue. Therefore, it is worth exploring alternative strategies.

In this paper, we propose to apply a particle swarming approach to deal with the VCC problem. Although we cannot claim that our approach is better than all those in the literature (no benchmarks or public implementations are available), we show that the particle swarm approach is, performance-wise, significantly better than previously proposed complete approaches based on discretizing the space and then exhaustively searching, while being still able to produce good camera placements.

The paper is organized as follows. Section 2 reviews related work, while Section 3 describes the problem formulation. Then we present the Particle Swarm approach to the problem in Section 4 and in Section 5 we show the experimental results on a realistic 3D model. Finally, in Section 6 we outline some conclusion.

2 Related Work

A comprehensive survey of approaches to camera control can be found in [3]. In the following, we focus on approaches to the VCC problem that: (i) employ a declarative “cinematographic” style, i.e. where the VCC problem is expressed as a set of requirements on the obtained image, such as distance specification, relative viewing angles, occlusions, and (ii) model the problem as a constrain and/or optimization system. These approaches are by far the most general, and therefore interesting for a wide range of applications. On the other hand, they currently cannot be used for any situation where camera placements needs to be generated in real-time.

In constrained and/or optimization approaches to VCC, the properties of the image “seen” by the camera are expressed as numerical constraints on the camera variables, which typically include camera position, orientation, and focal angle (see the next Section for a detailed formulation of the problem). The main characteristics that differentiate these approaches relate to the richness of the language to express image properties (what and how properties can be expressed) and the properties and performances of the solving techniques.

Constraint-based approaches include:

- the CAMDROID system for automated camera planning [4], which uses the CFSQP numerical constraint solver package. As admitted by the authors, the solving process is sensitive to the initial configuration and is subject to local minima failures [1].
- CONSTRAINT CAM [5], which uses a partial constraint satisfaction system to provide alternate solutions when constraints cannot be completely satisfied. The approach is based on a limited subset of cinematographic properties (viewing angle, viewing distance and occlusion avoidance).

An example of a purely optimization-based approach is CAMPLAN system [6], which uses a metaheuristic search (genetic algorithms) method. However, the CAMPLAN genetic algorithm produces solutions in widely varying amounts of time and is also subject to the initial population of solutions [1].

Some approaches mix constraints and optimization. For example, Bares et al. [2] propose a heuristic-based complete search algorithm. The process is applied inside promising 3D areas computed through simple geometric intersections, and is based on discretizing the search space to increase efficiency. Similarly,

Pickering [7] uses constraints to create feasible regions of space that will serve as bounds for an optimization procedure. The search space is subdivided by a “shadow-volumes” algorithm based on the properties of the image, and the feasible regions are then discretized and stored in an octree structure. Each node of the octree is then used as a starting point for a genetic algorithm that tries to find a solution to the problem.

Some approaches try not to focus on finding an unique solution, but isolates identical possible solutions in 3D volumes with respect to their visual properties. For example, [1] relies on a space partitioning process derived from a study of possible camera locations w.r.t. to the objects in the scene, and local search numerical techniques to compute good representatives of each volume.

3 Virtual Camera Composition

The VCC problem is instanced by means of camera and scene objects’ models and by a declarative language for specifying the desired features of the frame.

The language includes a set of predicates to state a description of the resulting image on the basis of some geometric properties (e.g., object size, position, occlusion, . . .). The predicates can be combined by means of the classical logical operators ‘and’, ‘or’ and ‘not’. The predicates semantics is defined in terms of constraints and/or objectives that are imposed on the set of possible camera parameters to obtain the desired image.

3.1 Camera and Object Model

The camera model considered in this work is shown in Figure 1a and it is described by 6 extrinsic parameters defined as follows:

Camera position: $\mathbf{C} = (C_x, C_y, C_z) \in \mathbb{R}^3$;

Camera aim direction: $\mathbf{A} = (A_x, A_y, A_z) \in \mathbb{R}^3$, $\|\mathbf{A}\|_2 = 1$;

Camera roll angle: the angle is limited to valid head roll, $\psi \in [-20^\circ, 20^\circ]$ (which is the norm in camera composition)

Camera horizontal field of view (FOV) angle: the range of angles excludes telephoto or fisheye lenses, therefore $\phi \in [25^\circ, 100^\circ]$;

Aspect ratio: $\rho \in [1, 21/9]$;

Focus depth: $d \in \mathbb{R}^+$.

Each relevant object in the scene is described using a bounding sphere, which includes the whole object (see Figure 1b). The sphere is defined by its center position and the radius, but in order to retain some information about the object also a front direction and an up vector are added (these vectors allows one to express that an object should be viewed from a certain direction). In detail the bounding sphere parameters are the following:

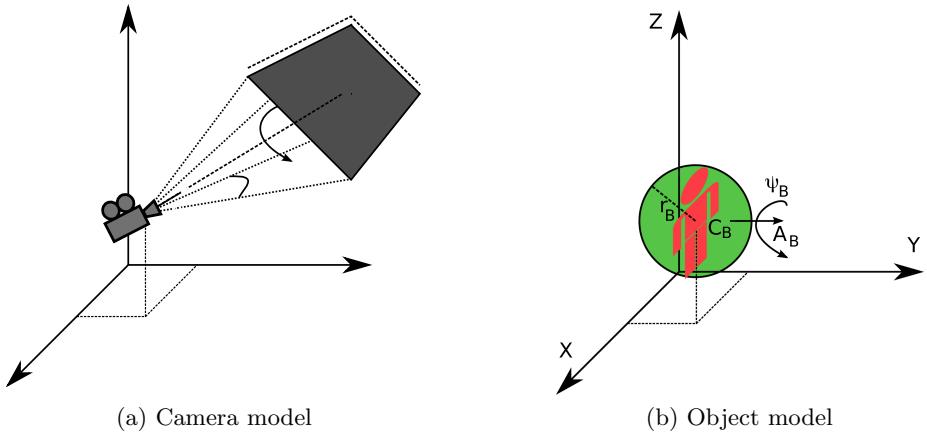


Fig. 1. The camera and object model

Center position: $\mathbf{C}_B = (C_{B_x}, C_{B_y}, C_{B_z}) \in \mathbb{R}^3$;

Radius: $r_B \in \mathbb{R}^+$;

Front direction: $\mathbf{A}_B = (A_{B_x}, A_{B_y}, A_{B_z}) \in \mathbb{R}^3$, $\|\mathbf{A}_B\|_2 = 1$;

Roll angle: $\psi_B \in [-180^\circ, 180^\circ]$.

3.2 Image/Camera Constraints

The set of basic constraints included in the description language allows to specify the image properties about view angles, inclusion, size, distance, position, and occlusion. In addition, also constraints on the camera parameters can be imposed.

In the following we outline the predicates of the image description language and their semantics. For the sake of conciseness we omit the details of the mathematical formulation of the constraints.

Object View Angles. The object view angle defines a zone of the 3D space in which the camera must be positioned to watch an object from a suitable view angle. The zone is described by means of two pair of angles in spherical coordinates w.r.t. the object's front and up directions (see Figure 2a).

- **Object view horizontal angle:** $v - angle(\text{Object } x, \text{angle } \theta_v, \text{angle } \theta'_v)$, requires the image of object x to be taken from an horizontal angle in the range $[\theta_v, \theta'_v]$, $\theta_v, \theta'_v \in [-180^\circ, 180^\circ]$;
- **Object view vertical angle:** $h - angle(\text{Object } x, \text{double } min_\alpha, \text{double } max_\alpha)$, requires the image of object x to be taken from a vertical angle in the range $[\phi_v, \phi'_v]$, $\phi_v, \phi'_v \in [-90^\circ, 90^\circ]$.

Object Inclusion. This constraint requires that at least a fraction of the bounding sphere lays inside the field of view of the camera. The problem to

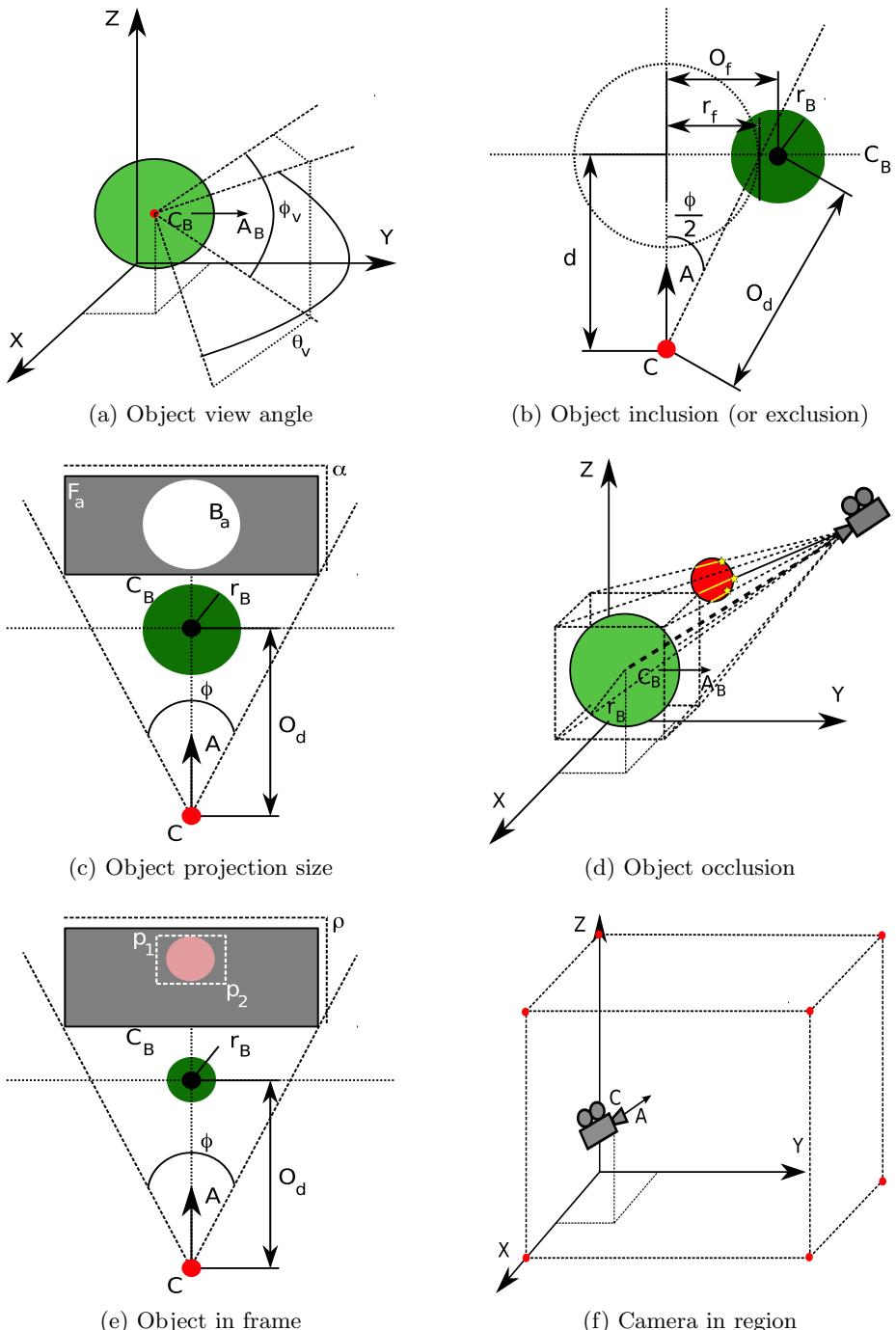


Fig. 2. A visual illustration of image and camera constraints

determine the precise fraction value is approximated by a pessimistic model in which the visible area is a circle whose diameter depends on the FOV angle. This way it is possible to compute this constraint by means of circle intersections, as shown in Figure 2b.

The only parameter for this constraint is the visible fraction required for the object bounding sphere:

- **Inclusion:** $include(\text{Object } x, \text{double } \nu)$, object x is required to appear in the image at least for the percentage $\nu \in]0, 1]$ of its full size.

Object Exclusion. This constraint can be modeled with the same approximation employed for the previous constraint.

- **Exclusion:** $exclude(\text{Object } x)$, object x is required not to appear in the image.

Object Projection Size. The projection of a given object in the frame is required to cover a given amount of the image:

- **Size fraction:** $size(\text{Object } x, \text{double } \sigma)$, object x is required to cover a percentage σ in the image, $\sigma \in]0, 1]$.

Distance. The distance between the object and the camera is constrained to lie within two given bounds d_{min} and d_{max} .

- **Object distance:** $distance(\text{Object } x, \text{double } d_{min}, \text{double } d_{max})$: the distance between the camera and the object x must be in the range $[d_{min}, d_{max}]$, $0 \leq d_{min} \leq d_{max} \leq +\infty$.

Position in Frame. The projection of an object must be placed within or outside a given area of the image, delimited by a bounding rectangle:

- **Include in frame:** $includeInFrame(\text{Object } x, \text{2DPoint } p_1, \text{2DPoint } p_2)$, object x must appear in the 2D frame comprised between p_1 and p_2 . The points p_1 and p_2 are expressed in relative coordinates w.r.t. the screen (i.e., $p_1, p_2 \in [0, 1] \times [0, 1]$).
- **Exclude from frame:** $excludeFromFrame(\text{Object } x, \text{2DPoint } p_1, \text{2DPoint } p_2)$: object x must not appear in the 2D frame comprised between p_1 and p_2 .

Object Occlusion. The evaluation of the object occlusion relies on a common *ray casting* technique [8]. A bounding box around the object is built and the rays connecting the camera with the box vertices and the object center are traced (see Figure 2d). The level of occlusion is estimated as the ratio between the weighted number of the traced rays that reach the object and the total number of rays. The weights are assigned so that the ray leading to the object center has higher influence.

- **Occlusion:** $occlusion(\text{Object } x, \text{double } \nu)$, requires that the occlusion level of object x to be less than $\nu \in [0, 1]$.

Camera Constraints. Camera constraints concern limitations in the camera positioning or orientation that can be imposed by physical features of the scene. For example it is meaningless to allow the camera to be placed inside the walls or other objects, and in most situations it is also unsuitable to have upside-down cameras.

The camera can be constrained to be placed in a region of the 3D scene specified by a world-coordinates aligned box (see Figure 2f). More in general, the region can be specified as a set union or intersection of a set of boxes. Each box is described by the following pair of coordinates: bottom-right-back, $\mathbf{p}_1 = (p_{1_x}, p_{1_y}, p_{1_z}) \in \mathbb{R}^3$ and top-left-front:] $\mathbf{p}_2 = (p_{2_x}, p_{2_y}, p_{2_z}) \in \mathbb{R}^3$. The constraints are:

- **Camera inside region:** $cameraInside(3DPoint p_1, 3DPoint p_2)$, the camera must be placed within the box defined by the points p_1 and p_2 .
- **Camera outside region:** $cameraOutside(3DPoint p_1, 3DPoint p_2)$: the camera must not be placed within the box defined by the points p_1 and p_2 .

Another possibility is to require the camera to be placed above a plane specified by its origin and the normal direction. This is useful, for example, to model the fact that the camera position is above a floor of the scene.

- **Camera above plane:** $cameraAbovePlane(3DPoint o, 3DPoint n)$, the camera must be placed above the plane whose origin is $\mathbf{o} \in \mathbb{R}^3$ and the normal vector is $\mathbf{n} \in \mathbb{R}^3$, $\|\mathbf{n}\|_2 = 1$.

Finally, other camera constraints can be used to fix the position or the orientation of the camera to specific values:

- **Fix camera Position:** $lockPosition(3DPoint p)$, the camera must be placed precisely in $\mathbf{p} \in \mathbb{R}^3$.
- **Fix camera Roll:** $lockRoll(angle \alpha)$, the angle of the aim direction w.r.t. the Z axis must be set to $\alpha \in [-20^\circ, 20^\circ]$.
- **Fix camera Yaw:** $lockYaw(angle \alpha)$, the angle of the aim direction w.r.t. the Y axis must be set to $\alpha \in [-180^\circ, 180^\circ]$.
- **Fix camera Pitch:** $lockPitch(angle \alpha)$: the angle of the aim direction w.r.t. the X axis must be set to $\alpha \in [-180^\circ, 180^\circ]$.

4 A Particle Swarm Approach

Particle Swarm Optimization (PSO) [9,10] is a population-based method for global optimization, which is inspired by the social behavior that underlies the movements of a swarm of insects or a birds' flock.

Given a D -dimensional (compact) search space $S \in \mathbb{R}^D$ and a scalar objective function $f : S \rightarrow \mathbb{R}$ that assesses the quality of each point $x \in S$ and (without loss of generality) has to be maximized, a *swarm* is made up of a set of N particles, which are located in that space. The i -th particle is described by three D -dimensional vectors, namely:

- the particle current *position* $\mathbf{x}_i = (x_{i_1}, x_{i_2}, \dots, x_{i_D})$;
- the particle *velocity* $\mathbf{v}_i = (v_{i_1}, v_{i_2}, \dots, v_{i_D})$, i.e., the way the particle moves in the search space;
- the particle *best visited position* (as measured by the objective function f) $\mathbf{P}_i = (p_{i_1}, p_{i_2}, \dots, p_{i_D})$, which is a memory of the best positions ever visited during the search.

The index of the particle that reached the global best visited position is denoted by g , that is, $g = \arg \max_{i=1, \dots, N} f(\mathbf{P}_i)$.

At the beginning of the search (step $n = 0$), the particles are set at random locations and with random velocities. The search is performed as an iterative process, which at step n modifies the velocity and position vectors of each particle on the basis of the values at step $n - 1$. The process evolves according to the following rules (superscripts denote the iteration number):

$$\mathbf{v}_i^n = w\mathbf{v}_i^{n-1} + c_1 r_1^{n-1} (\mathbf{p}_i^{n-1} - \mathbf{x}_i^{n-1}) + c_2 r_2^{n-1} (\mathbf{p}_g^{n-1} - \mathbf{x}_g^{n-1}) \quad (1)$$

$$\mathbf{x}_i^n = \mathbf{x}_i^{n-1} + \mathbf{v}_i^n \quad i = 1, 2, \dots, N \quad (2)$$

In the equations, the different parameters have to be interpreted as follows: The values r_1 and r_2 are two uniformly distributed random numbers in $[0, 1]$, whose purpose is to maintain population diversity. Constants c_1 and c_2 are respectively the so-called *cognitive* and *social* parameter, which are related to the speed of convergence, and in our experimentation are set to 0.5 according to the results of preliminary experiments for parameter tuning.

Parameter w is an *inertia* weight and it establishes the influence of the search history on the current move. A high weight is related to a global exploration, while a low weight allows a local exploration (also called exploitation). In our implementation this parameter varies from an initial value of 1.2 to 0.2, in order to balance between exploration and exploitation in the different stages of the search. We experimented with two different ways of varying the inertia weight, namely reducing it using a *linear* and an *exponential* scheme, however the linear reduction scheme has shown superior performances therefore in the following we report only its results.

4.1 Search Space and Objective Function

In order to adapt PSO for a specific problem we are required to specify the search space and the objective function.

In the case at hand, the search space encoding can be naturally composed by all camera parameters, which account for 10 dimensions. However, in order to make the search more effective, we decide to consider some of the constraints described above as *hard* constraints, i.e., constraints that must be always satisfied along the search. This allows to restrict the search space removing from it those regions in which we are sure that no suitable solution can be found.

The predicates modeled as hard constraints are those dealing with the camera properties either relative, such as *distance()*, *angle()*, *height()*, or absolute,

i.e., *cameraInside()*, *cameraOutside()*, *cameraAbovePlane()*, *lockPosition()*, *lockRoll()*, *lockYaw()*, *lockPitch()*. These properties determine an admissible volume for the camera parameters, which can be quite complex and made up of non-contiguous regions, giving rise to a disconnected search space. For this reason the choice of preventing the particles from going across those regions is not particularly adequate for this problem and we opt for allowing the particles to go over these boundaries but we assign them the worst possible value of the objective function (i.e., 0).

The remaining predicates are treated as *soft* constraints, that is, restrictions that can be violated at the price of deteriorating a quality function. The degree of satisfaction of the predicates is evaluated by means of a function $f : \Pi \times D \rightarrow [0, 1]$ whose semantics depends on the predicate $\pi \in \Pi$ at hand. In general, the function measures a relative difference between the desired value for the property and its actual value. The value of f is then normalized in order to obtain a real value in the range $[0, 1]$, where 1 represents the satisfaction of the associated constraint and 0 is the complete unfulfillment (in a way that is similar to the fuzzy logic). These functions are highly non-linear because camera projections are involved in their computation.

Since the single predicates can be combined in complex image descriptions by means of the logical operators, the objective function for the combination of predicates is computed according to the following rules:

$$f(\pi_1 \wedge \pi_2, \mathbf{x}) = w_1 f(\pi_1, \mathbf{x}) + w_2 f(\pi_2, \mathbf{x}) \quad (3)$$

$$f(\pi_1 \vee \pi_2, \mathbf{x}) = \max\{f(\pi_1, \mathbf{x}), f(\pi_2, \mathbf{x})\} \quad (4)$$

$$f(\neg\pi_1, \mathbf{x}) = 1 - f(\pi_1, \mathbf{x}) \quad \pi_1, \pi_2 \in \Pi, \mathbf{x} \in D \quad (5)$$

Notice that the conjunction of predicates is encoded as a weighted sum of the atomic predicates, where the relative influence of the specific property (i.e., the weights w_1 and w_2) can be set by the user in the image description.

In general, the evaluation of a particle requires the function f to be computed for all the properties of the image description. However, this process can be quite time consuming especially in the case of complex image descriptions or when properties that require a computationally intensive evaluation (e.g., occlusion) are specified. Therefore we adopt a *lazy* evaluation mechanism for the objective function, which relies on the monotonicity of the operators employed. We notice that the computation of f for the particle i can be stopped if the sum of the weights of the predicates that still have to be evaluated is smaller than the best objective value $f(\cdot, \mathbf{P}_i)$. Therefore, as a heuristic, it could be useful to sort the predicates leaving at the end the ones whose evaluation has a higher computational cost.

5 Implementation and Experimental Results

The PSO for VCC has been implemented as a part of a general C++ library called **Constraint Camera Library**, which can be used as a component in graphics

engines or 3D applications. The library contains a set of methods for reading an XML description of the camera placement problem (i.e., an image description composed using the language described above, see Listing 1 for an example), and the optimization suite that, at present, includes also an exhaustive search method based on a search space discretization. The exhaustive search algorithm is one of the methods currently used for placing the camera in many virtual environments, and in this work it has been employed as the baseline for the evaluation of PSO. In the tests the library has been compiled with Microsoft Visual Studio 2005 C++ compiler and run on an Intel Core 2 Duo 3GHz PC equipped with 2Gb of RAM and running Microsoft Windows Vista 64.

The PSO optimizer has been tested on a realistic 3D model of Venzone, a small medieval town surrounded by walls that was almost completely destroyed by two earthquakes in 1976. The scene employed in the experimentation is a $50m \times 10m \times 50m$ representation of the town hall and the main square. The model is quite complex: it is composed by about 52000 triangles and contains several non-convex objects (such as a portico). The methods have been tested on 4 different image descriptions of increasing complexity (numbered as 1-4 in Listing 1). The PSO population size is 40 and the procedure is iterated until a solution within 5% of the optimal best value has been found.

Listing 1. An example of image description for an Over The Shoulder shot

```

<scene>
  <cell id="main" bounds="20,70,15,25,-25,25">
    <object id="blueWarrior" frontVector="-1,0,1" upVector="0,1,0"/>
    <object id="redWarrior" frontVector="1,0,-1" upVector="0,1,0"/>
    <!-- 1. the blue warrior must be visible -->
    <property type="outside" ref="blueWarrior"/>
    <property type="occlusion" ref="blueWarrior" expectedValue="0" weight="1"/>
    <property type="include" ref="blueWarrior" weight="1"/>
    <!-- 2. also the red warrior must be visible -->
    <property type="outside" ref="redWarrior"/>
    <property type="occlusion" ref="redWarrior" expectedValue="0" weight="1"/>
    <property type="include" ref="redWarrior" weight="1"/>
    <!-- 3. the red warrior must be shot from a reverse angle -->
    <property type="angle" ref="redWarrior" min="0" max="-180" weight="1"/>
    <!-- 4. the red warrior must be shot at 100% (max) size -->
    <property type="objectSize" ref="redWarrior" expectedValue="1" weight="2"/>
  </cell>
</scene>
```

The results of the PSO optimizer are presented in Table 1. In the table there are reported the averages and the standard deviations (between parentheses) of a set of performance indicators, namely CPU running time, the value of the objective function, the number of iterations, and the number of objective function evaluations. The last two columns report the time and the objective function found by the exhaustive search procedure on a $20 \times 20 \times 20$ discretization.

We observe that the PSO optimizer is between one and two orders of magnitude faster than the exhaustive search procedure and it is able to reach comparable results, except in the description 4 where PSO outperforms the exhaustive search procedure that would need a finer discretization for reaching acceptable results. Concerning the number of evaluations of the objective function, it is

Table 1. Results of the PSO optimizer on the 3D model for different image descriptions

Desc	Time (ms)	PSO		f evaluations			Exhaustive	
		f	Iterations	Full	Lazy	Time (ms)	f	
1	1132 (67)	1 (0)	1.1 (0.32)	27.7 (4)	14.9 (6.87)	197529	1	
2	4359 (3032)	1 (0.01)	3.9 (4.46)	55.1 (33.9)	55.3 (73.37)	360372	1	
3	6558 (4525)	0.98 (0.01)	7.4 (4.81)	95.3 (67.06)	62.8 (80.69)	116753	0.99	
4	22055 (12638)	0.96 (0.01)	21 (7.6)	366.5 (231.64)	219.2 (80.2)	115201	0.72	



(a) The blue warrior must be visible (1)



(b) Both warriors must be visible (2)



(c) Red warrior shot from back (3)



(d) Red warrior shot at 100% size (4)

Fig. 3. Examples of shots computed by PSO for the different image descriptions

possible to see that the lazy evaluation has a noticeable impact on avoiding unnecessary computation.

6 Conclusions

We have presented a PSO approach for the VCC problem. The proposed method has shown to perform significantly better than an exhaustive search on a discretization of the 3D model. However, additional experiments on other models and image descriptions should be carried out to evaluate the method more thoroughly. Moreover, the PSO should be compared against alternative solution methods (e.g., local search or genetic algorithms).

Acknowledgments. The authors acknowledge the financial support of the Italian Ministry of Education, University and Research (MIUR) within the FIRB project number RBIN04M8S8.

References

1. Christie, M., Normand, J.M.: A semantic space partitioning approach to virtual camera control. *Computer Graphics Forum* 24(3), 247–256 (2005); Special Issue: Proceedings of the Eurographics Annual Conference
2. Bares, W., McDermott, S., Boudreault, C., Thainimit, S.: Virtual 3d camera composition from frame constraints. In: *MULTIMEDIA 2000: Proceedings of the 8th ACM International Conference on Multimedia*, New York, USA, pp. 177–186 (2000)
3. Christie, M., Olivier, P.: Automatic camera control in computer graphics. In: *Proceedings of the Annual Eurographics Conference 2006*, pp. 89–113 (2006)
4. Drucker, S.M., Zeltzer, D.: Camdroid: a system for implementing intelligent camera control. In: *SI3D 1995: Proceedings of the 1995 symposium on Interactive 3D graphics*, New York, USA, pp. 139–144 (1995)
5. Bares, W.H., Gregoire, J.P., Lester, J.C.: Realtime constraint-based cinematography for complex interactive 3d worlds. In: *AAAI/IAAI*, pp. 1101–1106 (1998)
6. Halper, N., Oliver, P.: CamPlan: A camera planning agent. In: *AAAI Workshop on Smart Graphics* (2000)
7. Pickering, J.H.: Intelligent camera planning for computer graphics. PhD thesis, Department of Computer Science, University of York (2002)
8. Roth, S.D.: Ray casting for modeling solids. *Computer Graphics and Image Processing* 18, 109–144 (1982)
9. Kennedy, J., Eberhart, R.C., Shi, Y.: *Swarm Intelligence*. Morgan Kaufmann Publishers, San Francisco (2001)
10. Eberhart, R.C., Kennedy, J.: Particle swarm optimization. In: *Proceedings of the IEEE International Conference on Neural Networks 1995*, Perth, WA, Australia, vol. 4, pp. 1942–1948 (1995)

The Binary Bridge Selection Problem: Stochastic Approximations and the Convergence of a Learning Algorithm

Armand M. Makowski

Department of Electrical and Computer Engineering
and Institute for Systems Research
University of Maryland, College Park, MD, USA
`armand@isr.umd.edu`

Abstract. We consider an ant-based algorithm for binary bridge selection, and analyze its convergence properties with the help of techniques from the theory of stochastic approximations.

1 Introduction

In [1], Deneubourg et al. presented a simple experimental setup to show that path selection to a food source in the Argentine ant *Linepithema humile* is based on self-organization. In this experiment, the food source is separated from the nest by a bridge with two equally long branches, say A and B [2,3]. Initially there is no pheromone on the branches which have therefore the same probability of being selected by the ants. However, random fluctuations will cause a few more ants to randomly select one branch, say A , over the other. Because ants deposit pheromone while walking, the greater number of ants on branch A determines a greater amount of pheromone on A , which in turn stimulates more ants to choose A , and so on, thereby leading to *reinforcement*. Deneubourg et al. developed a model of this phenomenon, and showed by *simulations* that its behavior closely matches the experimental observations, namely that *only one* of the branches is eventually used *most* of the time!

The model – The point of departure for the model is the simplifying assumption that the amount of pheromone on a branch is proportional to the number of ants that have previously traveled on the branch. This reflects situations where pheromone evaporation need not be taken into account as the experiment operates at a time scale (roughly one hour) much smaller than one where evaporation would be noticeable (e.g., many hours and possibly days). Deneubourg et al. posited that the probability of the $(n+1)^{rst}$ ant selecting a branch is determined by the number of ants that have previously used that branch. With this in mind, for each $n = 1, 2, \dots$, let A_n and B_n denote the number of ants that have used branch A and B , respectively, among the first n ants. The (conditional) probability p_n that the $(n+1)^{rst}$ ant selects the branch A , was taken to be

$$p_n := \frac{(K + A_n)^\nu}{(K + A_n)^\nu + (K + B_n)^\nu} \quad (1)$$

with scalars $K \geq 0$ and $\nu \geq 0$ held fixed throughout the discussion. The particular form (1) was obtained through experiments described in [4] with values $\nu \simeq 2$ and $K \simeq 20$ giving the best experimental fit. The parameter K quantifies the degree of attraction of an unmarked branch in that the larger the value of K , the greater the amount of pheromone needed to make the choice non-random. The value of ν determines the degree of nonlinearity of the choice function, and through it, behavior reinforcement – When ν is large, one branch needs only have slightly more pheromone than the other for the next ant to select it.

The *self-organization* observed in colonies of Argentine ants can be expressed as

$$\lim_{n \rightarrow \infty} \max \left(\frac{A_n}{n}, \frac{B_n}{n} \right) = 1 \quad a.s. \quad (2)$$

Indeed, given the constraint $A_n + B_n = n$ (see Section 2), this amounts to either $\frac{A_n}{n}$ or $\frac{B_n}{n}$ being close to unity, and captures the fact that one of the branches is eventually used most of the time. In [1] this was shown to be the case through simulations for the values $\nu \simeq 2$ and $K \simeq 20$.

Contributions – As we shall see shortly these limited simulation results may be misleading in that the convergence (2) may fail. More precisely, the a.s. convergence of $\{(\frac{A_n}{n}, \frac{B_n}{n}), n = 1, 2, \dots\}$ can be entirely characterized by the value of the parameter ν . Three different behaviors emerge depending on whether $\nu \leq 1$, $\nu = 1$ or $\nu > 1$, and are related to simple properties of the mapping $P_\nu : [0, 1] \times \mathbb{R}_+ \rightarrow [0, 1]$ given by

$$P_\nu(a, c) := \frac{(a+c)^\nu}{(a+c)^\nu + (1-a+c)^\nu}, \quad a \in [0, 1], \quad c \geq 0. \quad (3)$$

To see why this might be the case, we observed that the random variables (rvs) $\{\frac{A_n}{n}, n = 1, 2, \dots\}$ can be interpreted as the output sequence of the one-dimensional recursion

$$\frac{A_{n+1}}{n+1} = \frac{A_n}{n} + \frac{1}{n+1} \left(\mathbf{1}[U_{n+1} \leq p_n] - \frac{A_n}{n} \right), \quad n = 1, 2, \dots \quad (4)$$

with $p_n = P_\nu(\frac{A_n}{n}, \frac{K}{n})$ and the rvs $\{U_n, n = 1, 2, \dots\}$ are assumed to be i.i.d. rvs which are uniformly distributed on $[0, 1]$. This recursion can be viewed as a stochastic approximation of the Robbins-Monro type [5], albeit of a somewhat non-standard variety. Its limiting behavior is essentially determined by the stability of the limiting ODE

$$\dot{a}(t) = -a(t) + \frac{a(t)^\nu}{a(t)^\nu + (1-a(t))^\nu}, \quad t \geq 0 \quad (5)$$

and is known to be related to the solutions of the nonlinear equation

$$a = \frac{a^\nu}{a^\nu + (1-a)^\nu}, \quad a \in [0, 1]. \quad (6)$$

For $\nu \neq 1$, there are always three distinct roots, namely $a = 0, 1, \frac{1}{2}$ which are therefore the *potential* equilibrium points (hence possible limiting values for the sequence of rvs $\{\frac{A_n}{n}, n = 1, 2, \dots\}$): When $1 < \nu$, $a = \frac{1}{2}$ is a point of repulsion for the dynamics (4), while $a = 0, 1$ are each a point of attraction for it. As a result, one should expect the convergence

$$\lim_{n \rightarrow \infty} \frac{A_n}{n} \in \{0, 1\} \quad a.s. \quad (7)$$

and the desired convergence (2) follows. The situation is reversed when $0 < \nu < 1$ for then only $a = \frac{1}{2}$ constitutes a point of attraction while $a = 0, 1$ are both points of repulsion. This suggests that the convergence

$$\lim_{n \rightarrow \infty} \frac{A_n}{n} = \lim_{n \rightarrow \infty} \frac{B_n}{n} = \frac{1}{2} \quad a.s. \quad (8)$$

will take place, in which case the algorithm fails to mimic the experimental behavior. When $\nu = 1$, (6) has infinitely many solutions, and every point in the interval $[0, 1]$ is a possible limit.

The convergence properties of the modified Robbins-Monro algorithm can be studied with the help of martingale methods. These methods are modifications of well-known arguments used in the time-invariant case, and allow us to handle the time-varying aspects of the problem. Because of space limitations we omit most of the technical proofs which can be found in the extended version [6]. However, as an illustration of the power of martingale techniques, we outline arguments of a somewhat weaker result (Theorem 4) which identifies the possible accumulation points of the sequence $\{\frac{A_n}{n}, n = 1, 2, \dots\}$. When $\nu \neq 1$, this leads to the existence of an a.s. limit for this sequence (Theorem 5) and a finer analysis (available in [6]) is then required to identify the limits in each of the cases $0 < \nu < 1$ and $1 < \nu$.

Additional perspectives – Mathematical models for the binary bridge selection problem have appeared in literature, e.g., see [2] and [3], but without any analysis of the convergence properties of the underlying algorithm. We also note that multiple bridges can be handled by resorting to ideas from the theory of stochastic approximations [5], but this is beyond the scope of this short conference paper. Finally, links between the theory of stochastic approximations and processes with reinforcement have been pointed out by others, e.g., see the recent survey by Pemantle [9].

2 The Main Results on the Bridge Selection Algorithm

The scalars $K \geq 0$ and $\nu > 0$ are held fixed throughout the discussion. Let $\{U_n, n = 1, 2, \dots\}$ be a sequence of i.i.d. rvs which are uniformly distributed on the interval $[0, 1]$. With the two branches still denoted by A and B , we encode the branch selection for the $(n+1)^{rst}$ ant by means of the $\{A, B\}$ -valued rv S_{n+1} given by

$$S_{n+1} = A \quad \text{if and only if} \quad U_{n+1} \leq p_n \quad (9)$$

where p_n is selected according to (1). This gives rise to the two-dimensional recursion

$$\begin{aligned} A_{n+1} &= A_n + \mathbf{1}[U_{n+1} \leq p_n], \\ B_{n+1} &= B_n + \mathbf{1}[U_{n+1} > p_n], \end{aligned} \quad n = 1, 2, \dots \quad (10)$$

(with $\mathbf{1}[E]$ denoting the indicator function of the event E). We assume the \mathbb{R}_+^2 -valued initial condition (A_1, B_1) to be independent of the driving sequence $\{U_n, n = 1, 2, \dots\}$, and to satisfy

$$A_1 + B_1 = 1. \quad (11)$$

For the selection algorithm considered here, this assumption is satisfied by taking either $(A_1, B_1) = (1, 0)$ or $(A_1, B_1) = (0, 1)$.

Through (10) we see that the constraint (11) implies the relations

$$A_n + B_n = n, \quad n = 1, 2, \dots \quad (12)$$

For each $n = 1, 2, \dots$, the quantities A_n and B_n are each determined by the other. Thus, we need only consider the \mathbb{R}_+ -valued rvs $\{A_n, n = 1, 2, \dots\}$ given through the one-dimensional recursion

$$A_{n+1} = A_n + \mathbf{1}[U_{n+1} \leq p_n], \quad n = 1, 2, \dots \quad (13)$$

with

$$p_n = \frac{(K + A_n)^\nu}{(K + A_n)^\nu + (K + n - A_n)^\nu}, \quad (14)$$

where the $[0, 1]$ -valued rv A_1 is selected independently of the driving sequence $\{U_n, n = 1, 2, \dots\}$.

We now present the main convergence results for the stochastic recursion (13)-(14). Three cases emerge depending on the value of ν . No reinforcement takes place when $0 < \nu < 1$.

Theorem 1. *With $0 < \nu < 1$, it holds that*

$$\lim_{n \rightarrow \infty} \frac{A_n}{n} = \lim_{n \rightarrow \infty} \frac{B_n}{n} = \frac{1}{2} \quad a.s. \quad (15)$$

whence $\lim_{n \rightarrow \infty} \frac{A_n}{B_n} = 1$ a.s.

The case $\nu = 1$ is a boundary case.

Theorem 2. *With $\nu = 1$, the sequence of rvs $\{\frac{A_n}{n}, n = 1, 2, \dots\}$ converges a.s. to an $[0, 1]$ -valued rv a^* whose distribution depends on the initial condition A_1 .*

Learning or reinforcement occurs only when $1 < \nu$.

Theorem 3. *With $1 < \nu$, it holds that*

$$\lim_{n \rightarrow \infty} \max \left(\frac{A_n}{n}, \frac{B_n}{n} \right) = 1 \quad a.s. \quad (16)$$

with

$$\mathbb{P} \left[\lim_{n \rightarrow \infty} \frac{A_n}{n} = 1 \right] = \mathbb{P} \left[\lim_{n \rightarrow \infty} \frac{B_n}{n} = 1 \right] = \frac{1}{2}. \quad (17)$$

3 An Equivalent Stochastic Approximation

In order to study the asymptotic behavior of the sequence $\{A_n, n = 1, 2, \dots\}$ we make the change of variable

$$a_n := \frac{A_n}{n}, \quad n = 1, 2, \dots \quad (18)$$

The constraint (12) implies

$$0 \leq a_n \leq 1, \quad n = 1, 2, \dots \quad (19)$$

and the dynamics (13)-(14) can now be rewritten as

$$a_{n+1} = a_n + \frac{1}{n+1} (\mathbf{1}[U_{n+1} \leq p_n] - a_n), \quad n = 1, 2, \dots \quad (20)$$

where the $[0, 1]$ -valued rv a_1 is selected independently of the i.i.d. driving sequence $\{U_n, n = 1, 2, \dots\}$. With the mapping $P_\nu : [0, 1] \times \mathbb{R}_+ \rightarrow [0, 1]$ defined by (3), we note that

$$p_n = P_\nu \left(a_n, \frac{K}{n} \right). \quad (21)$$

In short, the rvs $\{a_n, n = 1, 2, \dots\}$ are generated through a stochastic approximation algorithm of the Robbins-Monro type where

$$\mathbb{E}[\mathbf{1}[U_{n+1} \leq p_n] - a_n | a_1, \dots, a_n] = p_n - a_n = P_\nu \left(a_n, \frac{K}{n} \right) - a_n \quad (22)$$

for all $n = 1, 2, \dots$. However, in contrast with the classical Robbins-Monro algorithm, the right-handside of (22) is a time-dependent function of a_n , namely $a \rightarrow P_\nu(a, \frac{K}{n}) - a$. Nevertheless, we still expect that any limit point a^* of the iterate sequence $\{a_n, n = 1, 2, \dots\}$ should be a “root” of this right-handside (at least in the limit), namely

$$P_\nu(a^*, 0) - a^* = 0. \quad (23)$$

If (23) had a *unique* solution in $[0, 1]$, say a^* , such that

$$(P_\nu(a, 0) - a)(a - a^*) < 0, \quad a \in [0, 1], \quad a \neq a^*, \quad (24)$$

then martingale arguments would readily imply that a^* is the a.s. limit of the iterate sequence $\{a_n, n = 1, 2, \dots\}$, e.g., see the classical references [7] and [8] for details. Here, depending on the value of ν , the negativity condition (24) may fail to hold, and a finer analysis is required to establish convergence. Moreover, when $\nu \neq 1$, the fact that (23) admits three distinct solutions, each either attractive or repulsive depending on the value of ν , further complicates matters.

4 A Preparatory Result and Its Consequences

Define the $[0, \frac{1}{4}]$ -valued rvs $\{V_n, n = 1, 2, \dots\}$ by

$$V_n := \left| a_n - \frac{1}{2} \right|^2, \quad n = 1, 2, \dots$$

Theorem 4. *Under the summability condition*

$$\sum_{n=1}^{\infty} \frac{1}{n+1} |(2a_n - 1)(p_n - a_n)| < \infty \quad a.s., \quad (25)$$

there exists an $[0, \frac{1}{4}]$ -valued rv V such that

$$\lim_{n \rightarrow \infty} V_n = V \quad a.s. \quad (26)$$

Before proving Theorem 4 in Section 5 we pause to derive some key consequences from it. To that end let $\text{Acc}(a_n, n = 1, 2, \dots)$ denote the set of accumulation points of the sequence $\{a_n, n = 1, 2, \dots\}$.

Corollary 1. *Assume $\nu \neq 1$. Under the assumption (25), we have*

$$\text{Acc}(a_n, n = 1, 2, \dots) \subseteq \{0, 1, \frac{1}{2}\} \quad a.s. \quad (27)$$

and the limiting rv V appearing in Theorem 4 is therefore an $\{0, \frac{1}{4}\}$ -valued rv.

Proof: By standard facts on summable series with non-negative terms, the convergence (25) yields $\lim_{n \rightarrow \infty} \frac{n}{n+1} |(2a_n - 1)(p_n - a_n)| = 0$ a.s. or equivalently,

$$\lim_{n \rightarrow \infty} (2a_n - 1)(p_n - a_n) = 0 \quad a.s. \quad (28)$$

Let Ω^* denote the event where both (26) and (28) hold; obviously $\mathbb{P}[\Omega^*] = 1$. Pick ω in Ω^* , and let α denote an accumulation point of the bounded sequence $\{a_n(\omega), n = 1, 2, \dots\}$. Thus, along a subsequence $\{n_k, k = 1, 2, \dots\}$ (which may depend on ω), we have $\alpha = \lim_{k \rightarrow \infty} a_{n_k}(\omega)$. Taking the limit in (28) along this subsequence, we get $\lim_{k \rightarrow \infty} (2a_{n_k}(\omega) - 1)(p_{n_k}(\omega) - a_{n_k}(\omega)) = 0$. By continuity,

$$\lim_{k \rightarrow \infty} p_{n_k}(\omega) = \lim_{k \rightarrow \infty} P_\nu \left(a_{n_k}(\omega), \frac{K}{n_k} \right) = P_\nu(\alpha, 0),$$

and the relation

$$(2\alpha - 1)(P_\nu(\alpha, 0) - \alpha) = 0 \quad (29)$$

follows. Consequently, any accumulation point α of $\{a_n(\omega), n = 1, 2, \dots\}$ is necessarily a solution of (29). With $\nu \neq 1$, direct inspection shows that (29) has exactly three solutions, namely $\alpha = 0, \frac{1}{2}, 1$ (see also Section 6), and the inclusion (27) is established.

We also note that

$$\lim_{k \rightarrow \infty} V_{n_k}(\omega) = \left| \lim_{k \rightarrow \infty} a_{n_k}(\omega) - \frac{1}{2} \right|^2 = \left| \alpha - \frac{1}{2} \right|^2, \quad (30)$$

and the convergence (26) (asserted in Theorem 4) now implies

$$\lim_{n \rightarrow \infty} V_n(\omega) = V(\omega) = \begin{cases} 0 & \text{if } \alpha = \frac{1}{2} \\ \frac{1}{4} & \text{if } \alpha = 0, 1. \end{cases} \quad (31)$$

The desired conclusion follows. \blacksquare

In Section 6 we outline arguments to show that the condition (25) indeed holds in each of the cases $0 < \nu < 1$ and $1 < \nu$; see [6] for full details. With this result in mind, we can now leverage Corollary 1 in order to prove the existence of an a.s. limit for the sequence $\{a_n, n = 1, 2, \dots\}$ when $\nu \neq 1$.

Theorem 5. *When $\nu \neq 1$, the sequence of rvs $\{a_n, n = 1, 2, \dots\}$ converges a.s. to an $\{0, \frac{1}{2}, 1\}$ -valued rv a^* .*

Proof: Assume $\nu \neq 1$ and pick ω in Ω^* (as defined earlier). By Theorem 4 we already know that $\lim_{n \rightarrow \infty} V_n(\omega)$ exists, and we now turn to establishing the existence of $\lim_{n \rightarrow \infty} a_n(\omega)$. Two cases are possible:

If $\frac{1}{2}$ is a point of accumulation for the sequence $\{a_n(\omega), n = 1, 2, \dots\}$, then there exists a subsequence $\{n_k, k = 1, 2, \dots\}$ (possibly dependent on ω) with $\lim_{k \rightarrow \infty} n_k = \infty$ such that $\lim_{k \rightarrow \infty} a_{n_k}(\omega) = \frac{1}{2}$, whence $\lim_{n \rightarrow \infty} V_n(\omega) = \lim_{k \rightarrow \infty} V_{n_k}(\omega) = 0$ by virtue of (30) and (31). Thus, $\frac{1}{2}$ is the only point of accumulation for the sequence $\{a_n(\omega), n = 1, 2, \dots\}$ and $\lim_{n \rightarrow \infty} a_n(\omega) = \frac{1}{2}$.

If $\frac{1}{2}$ is not a point of accumulation for the sequence $\{a_n(\omega), n = 1, 2, \dots\}$, then $\text{Acc}(a_n(\omega), n = 1, 2, \dots) \subseteq \{0, 1\}$. If this set of accumulation points coincides with $\{0, 1\}$, then there exist two distinct subsequences $\{n_k, k = 1, 2, \dots\}$ and $\{m_\ell, \ell = 1, 2, \dots\}$ (possibly dependent on ω) with $\lim_{k \rightarrow \infty} n_k = \infty$ and $\lim_{\ell \rightarrow \infty} m_\ell = \infty$ such that $\lim_{k \rightarrow \infty} a_{n_k}(\omega) = 0$ and $\lim_{\ell \rightarrow \infty} a_{m_\ell}(\omega) = 1$. Therefore, for arbitrary ε in the interval $(0, \frac{1}{2})$, there exist finite integers $k_*(\varepsilon)$ and $\ell_*(\varepsilon)$ such that $a_{n_k}(\omega) \leq \varepsilon$ for $k \geq k_*(\varepsilon)$ and $1 - \varepsilon \leq a_{m_\ell}(\omega)$ for $\ell \geq \ell_*(\varepsilon)$. Now, write $\mathbb{N}(\varepsilon) := \{n_k, k \geq k_*(\varepsilon)\} \cup \{m_\ell, \ell \geq \ell_*(\varepsilon)\}$, and with $p_*(\varepsilon) = \max(n_{k_*}(\varepsilon), m_{\ell_*}(\varepsilon))$, consider the set of integers given by $\mathbb{N}^*(\varepsilon) := \{p \notin \mathbb{N}(\varepsilon) : p > p_*(\varepsilon)\}$. From these definitions it is plain that

$$\varepsilon < a_p(\omega) < 1 - \varepsilon, \quad p \in \mathbb{N}^*(\varepsilon). \quad (32)$$

It is easy to see that the set $\mathbb{N}^*(\varepsilon)$ is *countably infinite*; see [6] for full details. Therefore, (32) would yield the inequalities $\varepsilon \leq \liminf_{j \rightarrow \infty} a_{p_j}(\omega) \leq \limsup_{j \rightarrow \infty} a_{p_j}(\omega) \leq 1 - \varepsilon$ where $\{p_j, j = 1, 2, \dots\}$ is the monotone labelling of the elements of $\mathbb{N}^*(\varepsilon)$, i.e., $p_j < p_{j+1}$ for $j = 1, 2, \dots$. But $\liminf_{j \rightarrow \infty} a_{p_j}(\omega)$ and $\limsup_{j \rightarrow \infty} a_{p_j}(\omega)$ are both accumulation points of the sequence $\{a_n(\omega), n = 1, 2, \dots\}$, and this contradicts the assumption $\text{Acc}(a_n(\omega), n = 1, 2, \dots) = \{0, 1\}$.

Thus, either $\text{Acc} (a_n(\omega), n=1, 2, \dots) = \{0\}$ or $\{1\}$, and $\lim_{n \rightarrow \infty} a_n(\omega)$ exists. ■

When $\nu = 1$, because $P_\nu(a, 0) = a$ for all a in the range $[0, 1]$, the relation (29) yields no information concerning the accumulation points of the sequence $\{a_n(\omega), n = 1, 2, \dots\}$ for each ω in Ω^* . This suggests already that the case $\nu = 1$ will require an approach different from the one used for either cases $0 < \nu < 1$ and $1 < \nu$; for lack of space we will not pursue this case any further.

5 A Proof of Theorem 4

We begin with some useful notation and facts: For each $n = 1, 2, \dots$, let \mathcal{F}_n denote the σ -field generated by the mutually independent rvs U_1, \dots, U_n and a_1 . The relation

$$\mathbb{E} [\mathbf{1}[U_{n+1} \leq p_n]^r | \mathcal{F}_n] = p_n, \quad r > 0 \quad (33)$$

will be used repeatedly.

We also introduce the \mathbb{R} -valued rvs $\{M_n, n = 1, 2, \dots\}$ given by $M_1 := 0$ and

$$M_{n+1} = \sum_{m=1}^n \frac{m}{(m+1)^2} (2a_m - 1) (\mathbf{1}[U_{m+1} \leq p_m] - p_m) \quad (34)$$

for all $n = 1, 2, \dots$. The rvs $\{M_n, n = 1, 2, \dots\}$ form a zero-mean \mathcal{F}_n -martingale with

$$\mathbb{E} [|M_{n+1}|^2] = \sum_{m=1}^n \frac{m^2}{(m+1)^4} \mathbb{E} [(2a_m - 1)^2 p_m (1 - p_m)]$$

for all $n = 1, 2, \dots$. Therefore, the \mathcal{F}_n -martingale $\{M_n, n = 1, 2, \dots\}$ is L_2 -bounded, hence a.s. convergent to some a.s. finite rv M [10, Thm. 9.4.5, p. 336] [11, Cor. 2.2, p. 18] with

$$\lim_{n \rightarrow \infty} M_n = M \quad a.s. \quad (35)$$

Fix $n = 1, 2, \dots$. From (20) we have

$$\left(a_{n+1} - \frac{1}{2} \right) = \left(a_n - \frac{1}{2} \right) + \frac{1}{n+1} (\mathbf{1}[U_{n+1} \leq p_n] - a_n). \quad (36)$$

Squaring both sides of this last relation and rearranging terms, we get

$$\begin{aligned} V_{n+1} &= V_n + \frac{1}{n+1} (2a_n - 1) (\mathbf{1}[U_{n+1} \leq p_n] - p_n) + \frac{1}{n+1} (2a_n - 1) (p_n - a_n) \\ &\quad + \frac{1}{(n+1)^2} (\mathbf{1}[U_{n+1} \leq p_n] (1 - a_n)^2 + \mathbf{1}[U_{n+1} > p_n] a_n^2). \end{aligned}$$

Simplifying the last term in this last expression, and iterating the resulting relation, we conclude with the help of (34) that

$$\begin{aligned} V_{n+1} &= V_1 + M_{n+1} + \sum_{m=1}^n \frac{1}{m+1} (2a_m - 1) (p_m - a_m) \\ &\quad + \sum_{m=1}^n \frac{1}{(m+1)^2} (p_m (1 - a_m)^2 + (1 - p_m) a_m^2). \end{aligned} \quad (37)$$

Next, we observe the monotone convergence

$$\begin{aligned} & \lim_{n \rightarrow \infty} \sum_{m=1}^n \frac{1}{(m+1)^2} (p_m(1-a_m)^2 + (1-p_m)a_m^2) \\ &= \sum_{n=1}^{\infty} \frac{1}{(n+1)^2} (p_n(1-a_n)^2 + (1-p_n)a_n^2) \end{aligned} \quad (38)$$

to a finite non-negative rv (bounded by $\sum_{n=1}^{\infty} n^{-2}$), while under the absolute summability (25), the a.s. convergence

$$\lim_{n \rightarrow \infty} \sum_{m=1}^n \frac{1}{m+1} (2a_m - 1)(p_m - a_m) < \infty \quad (39)$$

also takes place to an a.s. finite rv. With these remarks in place, let n go to infinity in (37). We readily get (26) upon combining (35), (38) and (39) because

$$0 \leq V_n \leq 1, \quad n = 1, 2, \dots \quad (40)$$

by virtue of (19). ■

6 Establishing the Summability Condition (25)

We establish the summability condition (25) by proving the stronger condition

$$\mathbb{E} \left[\sum_{n=1}^{\infty} \frac{1}{n+1} |(2a_n - 1)(p_n - a_n)| \right] < \infty. \quad (41)$$

As a first step, take expectations on both sides of the relation (37) so that

$$\begin{aligned} \mathbb{E}[V_{n+1}] &= \mathbb{E}[V_1] + \mathbb{E} \left[\sum_{m=1}^n \frac{1}{m+1} (2a_m - 1)(p_m - a_m) \right] \\ &\quad + \mathbb{E} \left[\sum_{m=1}^n \frac{1}{(m+1)^2} (p_m(1-a_m)^2 + (1-p_m)a_m^2) \right] \end{aligned} \quad (42)$$

for all $n = 1, 2, \dots$. From (19) we also note that

$$\sup_{n=1,2,\dots} \mathbb{E} \left[\sum_{m=1}^n \frac{1}{(m+1)^2} (p_m(1-a_m)^2 + (1-p_m)a_m^2) \right] \leq \sum_{n=1}^{\infty} n^{-2} < \infty. \quad (43)$$

Thus, establishing (41) (via (42) with the help of (43)) requires determining the sign and behavior of the terms

$$(2a_n - 1) \left(P_{\nu} \left(a_n, \frac{K}{n} \right) - a_n \right), \quad n = 1, 2, \dots$$

This leads naturally to considering the equation

$$P_\nu(a, c) = a, \quad a \in [0, 1] \quad (44)$$

for a given scalar $c \geq 0$ (held fixed throughout the discussion). We are interested in determining all the solutions to (44), such solutions being denoted $a(c)$. This set of solutions coincide with the set of solutions to the simpler equation

$$(a + c)^\nu(1 - a) = (1 - a + c)^\nu a, \quad a \in [0, 1]. \quad (45)$$

Thus, $a(c) = \frac{1}{2}$ is always a solution to (44). Moreover, with $\nu \neq 1$, if $c = 0$, then $a(c) = 0, 1$ are also solutions. It is a simple matter to check that if $a(c)$ is a solution to (44) for some $c \geq 0$, then $1 - a(c)$ is also a solution to it. The main technical fact needed to identify the solutions of the equation (44) is given next.

Proposition 1. *Fix $c \geq 0$. For each $\nu > 0$, the mapping $a \rightarrow P_\nu(a, c)$ is strictly increasing on the interval $[0, 1]$. If $0 < \nu < 1$, this mapping is strictly concave (resp. convex) on the interval $[0, \frac{1}{2}]$ (resp. $[\frac{1}{2}, 1]$) while if $1 < \nu$, the mapping is strictly convex (resp. concave) on the interval $[0, \frac{1}{2}]$ (resp. $[\frac{1}{2}, 1]$).*

6.1 Establishing (25) When $0 < \nu < 1$

We begin with an easy lemma that builds on Proposition 1.

Lemma 1. *Assume $0 < \nu < 1$. The equation (44) has exactly three solutions $a(c) = 0, \frac{1}{2}, 1$ if $c = 0$, and exactly one solution $a(c) = \frac{1}{2}$ if $c > 0$. Moreover, we have*

$$(2a - 1)(P_\nu(a, c) - a) \leq 0, \quad a \in [0, 1], \quad c \geq 0 \quad (46)$$

with strict inequality if and only if $a \neq 0, \frac{1}{2}, 1$ when $c = 0$ (resp. $a \neq \frac{1}{2}$ when $c > 0$).

Fix $n = 1, 2, \dots$. Lemma 1 implies $(2a_m - 1)(p_m - a_m) \leq 0$ for $m = 1, \dots, n$ so that $(2a_m - 1)(p_m - a_m) = -|(2a_m - 1)(p_m - a_m)|$, and the relation (42) can be rewritten as

$$\begin{aligned} & \mathbb{E}[V_{n+1}] + \mathbb{E}\left[\sum_{m=1}^n \frac{1}{m+1} |(2a_m - 1)(p_m - a_m)|\right] \\ &= \mathbb{E}[V_1] + \mathbb{E}\left[\sum_{m=1}^n \frac{1}{(m+1)^2} (p_m(1-a_m)^2 + (1-p_m)a_m^2)\right]. \end{aligned} \quad (47)$$

Using (40) and (43) we then get from (47) that

$$\sup_{n=1,2,\dots} \left(\mathbb{E}\left[\sum_{m=1}^n \frac{1}{m+1} |(2a_m - 1)(p_m - a_m)|\right] \right) \leq 1 + \sum_{n=1}^{\infty} n^{-2} < \infty.$$

The validity of (41) follows by monotone convergence, and the absolute summability (25) is now established when $0 < \nu < 1$. ■

6.2 Establishing (25) When $1 < \nu$

The needed facts complementing Proposition 1 are presented next

Lemma 2. Assume $1 < \nu$. The equation (44) has exactly one solution $\frac{1}{2}$ if $\frac{\nu-1}{2} \leq c$, and three solutions $a_-(c)$, $\frac{1}{2}$ and $a_+(c)$ if $0 \leq c < \frac{\nu-1}{2}$ with $a_-(c) < \frac{1}{2} < a_+(c)$ and $a_+(c) = 1 - a_-(c)$. If $c = 0$, then $a_-(c) = 0$ and $a_+(c) = 1$, while if $0 < c < \frac{\nu-1}{2}$, then $0 < a_-(c) < \frac{1}{2}$ with $\lim_{c \downarrow 0} a_-(c) = \lim_{c \downarrow 0} (1 - a_+(c)) = 0$. Moreover, when $0 < c < \frac{\nu-1}{2}$, with $I(c) := (a_-(c), a_+(c))$, it holds that

$$\max_{a \notin I(c)} |P_\nu(a, c) - a| = P_\nu(0, c) = \frac{c^\nu}{c^\nu + (1-c)^\nu}. \quad (48)$$

Set $n^* := \lceil \frac{2K}{\nu-1} \rceil$, and pick $n = n^* + 1, n^* + 2, \dots$, in which case $\frac{K}{n} < \frac{\nu-1}{2}$. Let I_n denote the interval $I(\frac{K}{n})$. By Lemma 2, we have $(2a_n - 1)(p_n - a_n) > 0$ if $a_n \in I_n$, but $(2a_n - 1)(p_n - a_n) \leq 0$ if $a_n \notin I_n$. Using these facts in (42) and rearranging terms, we find

$$\begin{aligned} & \mathbb{E}[V_{n+1}] + \mathbb{E} \left[\sum_{m=n^*+1}^n \frac{1}{m+1} |(2a_m - 1)(p_m - a_m)| \mathbf{1}[a_m \notin I_m] \right] \\ &= \mathbb{E}[V_1] + \mathbb{E} \left[\sum_{m=1}^{n^*} \frac{1}{m+1} (2a_m - 1)(p_m - a_m) \right] \\ & \quad + \mathbb{E} \left[\sum_{m=n^*+1}^n \frac{1}{m+1} |(2a_m - 1)(p_m - a_m)| \mathbf{1}[a_m \in I_m] \right] \\ & \quad + \mathbb{E} \left[\sum_{m=1}^n \frac{1}{(m+1)^2} (p_m(1-a_m)^2 + (1-p_m)a_m^2) \right]. \end{aligned} \quad (49)$$

By virtue of (40) and (43), we readily conclude from (49) that

$$\mathbb{E} \left[\sum_{m=1}^{\infty} \frac{1}{m+1} |(2a_m - 1)(p_m - a_m)| \mathbf{1}[a_m \notin I_m] \right] < \infty \quad (50)$$

if and only if

$$\mathbb{E} \left[\sum_{m=1}^{\infty} \frac{1}{m+1} |(2a_m - 1)(p_m - a_m)| \mathbf{1}[a_m \in I_m] \right] < \infty, \quad (51)$$

in which case the summability condition (41) holds.

For each $m = n^* + 1, n^* + 2, \dots$, Lemma 2 also yields

$$|(2a_m - 1)(p_m - a_m)| \leq 2P_\nu(0, \frac{K}{m}) \leq 2^\nu \left(\frac{K}{m} \right)^\nu, \quad a_m \notin I_m \quad (52)$$

as we make use of the convexity of the mapping $x \rightarrow x^\nu$. Consequently,

$$\sum_{m=1}^{\infty} \frac{1}{m+1} |(2a_m - 1)(p_m - a_m)| \mathbf{1}[a_m \notin I_m] \leq n^* + (2K)^\nu \sum_{m=n^*+1}^{\infty} \frac{1}{m^{\nu+1}}$$

and (50) therefore holds (since $\nu + 1 > 2$). This leads to (41), hence to the absolute summability (25) when $1 < \nu$. ■

References

1. Deneubourg, J.L., Aron, S., Goss, S., Pasteels, J.M.: The self-organizing exploratory pattern of the Argentine ant. *Journal of Insect Behavior* 3, 159–168 (1990)
2. Bonabeau, E., Dorigo, M., Theraulaz, G.: *Swarm Intelligence: From Natural to Artificial Systems*, Santa Fe Institute Studies in the Sciences of Complexity. Oxford University Press, Oxford (1999)
3. Dorigo, M., Stützle, T.: *Ant Colony Optimization*. MIT Press, Cambridge (2004)
4. Pasteels, J.M., Deneubourg, J.L., Goss, S.: Self-organization mechanisms in ant societies (I): Trail recruitment to newly discovered food sources. *Experientia Suppl.* 54, 155–175 (1987)
5. Kushner, H.J., Yin, G.: *Stochastic Approximation Algorithms and Applications*. Springer, New York (1997)
6. Makowski, A.M.: The bridge selection problem and the convergence properties of a learning algorithm – Stochastic approximations to the rescue (in preparation, 2008)
7. Gladyshev, E.G.: On stochastic approximation. *Theory of Probabilities and Applications* 10, 275–278 (1965)
8. Robbins, H., Siegmund, D.: A convergence theorem for non-negative almost supermartingales and some applications. In: Rustagi, J.S. (ed.) *Optimizing Methods in Statistics*, pp. 233–257. Academic Press, New York (1972)
9. Pemantle, R.: A survey of random processes with reinforcement. *Probability Surveys* 4, 1–70 (2007)
10. Chung, K.L.: *A Course in Probability Theory*, 2nd edn. Academic Press, New York (1974)
11. Hall, P., Heyde, C.C.: *Martingale Limit Theory and Its Applications*, Probability and Mathematical Statistics. Academic Press, New York (1980)

Two-Level ACO for Haplotype Inference Under Pure Parsimony

Stefano Benedettini¹, Andrea Roli¹, and Luca Di Gaspero²

¹ DEIS, Campus of Cesena

Alma Mater Studiorum Università di Bologna, Cesena, Italy

{stefano.benedettini, andrea.roli}@unibo.it

² DIEGM, University of Udine, Udine, Italy

l.digaspero@uniud.it

Abstract. Haplotype Inference is a challenging problem in bioinformatics that consists in inferring the basic genetic constitution of diploid organisms on the basis of their genotype. This information enables researchers to perform association studies for the genetic variants involved in diseases and the individual responses to therapeutic agents.

A notable approach to the problem is to encode it as a combinatorial problem under certain hypotheses (such as the *pure parsimony* criterion) and to solve it using off-the-shelf combinatorial optimization techniques. At present, the main methods applied to Haplotype Inference are either simple greedy heuristic or exact methods, which are adequate only for moderate size instances.

In this paper, we present an iterative constructive approach to Haplotype Inference based on ACO and we compare it against a state-of-the-art exact method.

1 Introduction

The role of genetic variation and inheritance in human diseases is extremely important, though still largely unknown [1]. To this aim, the assessment of a full Haplotype Map of the human genome has become one of the current high priority tasks of human genomics [2]. A haplotype is one of the two non identical copies of a chromosome of a diploid organism, i.e., an organism that has two copies of each chromosome, one inherited from the father and one from the mother. The haplotypes information makes it possible to perform association studies for the genetic variants involved in diseases and the individual responses to therapeutic agents. Technological limitations make it currently impractical to directly collect haplotypes by experimental procedures, but it is possible to collect *genotypes*, i.e., the conflation of a pair of haplotypes. Moreover, instruments can easily identify only whether the individual is *homozygous* (i.e., the alleles are the same) or *heterozygous* (i.e., the alleles are different) at a given site. Therefore, haplotypes have to be inferred from genotypes in order to reconstruct the detailed information and trace the precise structure of DNA variations in a population. This process is called *Haplotype Inference* (also known as *haplotype*

phasing) and the goal is to find a set of haplotype pairs (i.e., a *phasing*) so that all the given genotypes are *resolved*, that is, they can be obtained (or explained) by combining a pair of haplotypes from the set.

The main methods to tackle the Haplotype Inference are either combinatorial or statistical. Both, however, being of non-experimental nature, need some genetic model that could provide criteria for evaluating the solution returned with respect to actual genetic plausibility. In the case of the combinatorial methods, which are the subject of this work, one common criterion is *pure parsimony* [3], i.e., to search for the smallest collection of distinct haplotypes that solves the Haplotype Inference problem. This criterion is consistent with current observations in natural populations for which the actual number of haplotypes is vastly smaller than the total number of possible haplotypes, therefore the solutions found to this model are considered as good and informative phasings (see [3] for a discussion on the adequacy of this model).

Current approaches for solving the problem under the pure parsimony hypothesis (HI_{pp}) include simple greedy heuristic [4] and exact methods such as Integer Linear Programming [3,5,6,7], Semidefinite Programming [8,9], SAT models [10,11] and Pseudo-Boolean Optimization algorithms [12]. At present, complete approaches, i.e., the ones that guarantee to return an optimal solution, such as SAT-based ones, are very effective but they seem not to be particularly adequate very-large size instances. Hence, the need for approximate algorithms, such as metaheuristics, that trade completeness for efficiency. Moreover, a motivation for studying and applying approximate algorithms is that the criteria used to evaluate the solutions provide an approximation of the actual solution quality, therefore a proof of optimality is not particularly important.

The method we present in this work is a two-level ACO metaheuristic. To the best of our knowledge, besides [13], the only attempt to employ metaheuristic techniques for HI_{pp} is a recently proposed Genetic Algorithm [14] that is, however, not applied on real size instances.

The problem is formally stated in Sect. 2, along with the basic related concepts. In Sect. 3 we describe the two-level ACO we devised and in Sect. 4 we show the results of the experimental analysis in which we first compare the different variants of the two-level ACO, then we assess its performance by comparing it against state-of-the-art exact techniques.

2 The Haplotype Inference Problem

In the Haplotype Inference problem we deal with *genotypes*, that is, strings of length m that correspond to a chromosome with m sites. Each value in the string belongs to the alphabet $\{0, 1, 2\}$. A position in the genotype is associated with a site of interest on the chromosome and it has value 0 (wild type) or 1 (mutant) if the corresponding chromosome site is a homozygous site (i.e., it has that state on both copies) or the value 2 if the chromosome site is heterozygous. A *haplotype* is a string of length m that corresponds to only one copy of the chromosome (in diploid organisms) and whose positions can assume the symbols 0 or 1.

2.1 Genotype Resolution

Given a chromosome, we are interested in finding an unordered¹ pair of haplotypes that can explain the chromosome according to the following definition:

Definition 1 (Genotype resolution). *Given a chromosome g , we say that the unordered pair $\langle h, k \rangle$ resolves g , and we write $\langle h, k \rangle \triangleright g$ (or $g = h \oplus k$), if the following conditions hold (for $j = 1, \dots, m$):*

$$g[j] = 0 \Rightarrow \quad h[j] = 0 \wedge k[j] = 0 \quad (1a)$$

$$g[j] = 1 \Rightarrow \quad h[j] = 1 \wedge k[j] = 1 \quad (1b)$$

$$g[j] = 2 \Rightarrow (h[j] = 0 \wedge k[j] = 1) \vee (h[j] = 1 \wedge k[j] = 0) \quad (1c)$$

If $\langle h, k \rangle \triangleright g$ we indicate the fact that the haplotype h (respectively, k) contributes in the resolution of the genotype g writing $h \trianglelefteq g$ (resp., $k \trianglelefteq g$). We also say that h is a resolvent of g . This notation can be extended to set of haplotypes, writing $H = \{h_1, \dots, h_l\} \trianglelefteq g$, with the meaning that $h_i \trianglelefteq g$ for all $i = 1, \dots, l$. The operator \oplus is defined accordingly.

Conditions (1a) and (1b) require that both haplotypes must have the same value in all homozygous sites, while condition (1c) states that in heterozygous sites the haplotypes must have different values.

Observe that, according to the definition, for a single genotype string the haplotype values at a given site are predetermined in the case of homozygous sites, whereas there is a freedom to choose between two possibilities at heterozygous places. This means that for a genotype string with l heterozygous sites there are 2^{l-1} possible pairs of haplotypes that resolve it.

As an example, consider the genotype $g = (0212)$, then the possible pairs of haplotypes that resolve it are $\langle (0110), (0011) \rangle$ and $\langle (0010), (0111) \rangle$.

After these preliminaries we can state the *Haplotype Inference* problem as follows:

Definition 2 (Haplotype Inference problem). *Given a population of n individuals, each of them represented by a genotype string g_i of length m we are interested in finding a set ϕ of n pairs of (not necessarily distinct) haplotypes $\phi = \{\langle h_1, k_1 \rangle, \dots, \langle h_n, k_n \rangle\}$, so that $\langle h_i, k_i \rangle \triangleright g_i$, $i = 1, \dots, n$. We call H the set of haplotypes used in the construction of ϕ , i.e., $H = \{h_1, \dots, h_n, k_1, \dots, k_n\}$.*

From the mathematical point of view, there are many possibilities for building the set H , since there is an exponential number of possible haplotypes for each genotype. Therefore, a criterion should be added to the model for evaluating the solution quality.

One natural model of the Haplotype Inference problem is the already mentioned *pure parsimony* approach that consists in searching for a solution that minimizes the total number of distinct haplotypes used or, in other words, $|H|$, the cardinality of the set H . A trivial upper bound for $|H|$ is $2n$ in the case of all

¹ In the problem there is no distinction between the maternal and paternal haplotypes.

genotypes resolved by a pair of distinct haplotypes. It has been shown that the Haplotype Inference problem under the pure parsimony criterion is APX-hard [6] and therefore NP-hard.

2.2 Compatibility and Complementarity

It is possible to define a graph that expresses the compatibility between genotypes, so as to avoid unnecessary checks in the determination of the resolvents. In the graph $\mathcal{G} = (G, E)$, the set of vertices coincides with the set of the genotypes. Two genotypes g_1, g_2 are connected by an edge if they are *compatible*, i.e., one or more common haplotypes can resolve both of them. The formal definition of this property is as follows.

Definition 3 (Genotypes compatibility). Let g_1 and g_2 be two genotypes, g_1 and g_2 are compatible if, for all $j = 1, \dots, m$, the following conditions hold:

$$g_1[j] = 0 \Rightarrow g_2[j] \in \{0, 2\} \quad (2a)$$

$$g_1[j] = 1 \Rightarrow g_2[j] \in \{1, 2\} \quad (2b)$$

$$g_2[j] = 2 \Rightarrow g_1[j] \in \{0, 1, 2\} \quad (2c)$$

The same concept can be expressed also between a genotype and a haplotype as in the following definition.

Definition 4 (Compatibility between genotypes and haplotypes). Let g be a genotype and h a haplotype, g and h are compatible if, for all $j = 1, \dots, m$, the following conditions hold:

$$g[j] = 0 \Rightarrow h[j] = 0 \quad (3a)$$

$$g[j] = 1 \Rightarrow h[j] = 1 \quad (3b)$$

$$g[j] = 2 \Rightarrow h[j] \in \{0, 1\} \quad (3c)$$

We denote this relation with $h \mapsto g$, and we write $h[j] \mapsto g[j]$ when the conditions hold for the single site j . Moreover with an abuse of notation we indicate with $h \mapsto \{g_1, g_2, \dots\}$ the set of all genotypes that are compatible with haplotype h .

Notice that the set of genotypes that are compatible with a haplotype can contain only mutually compatible genotypes (i.e., they form a clique in the compatibility graph). As an example of compatibility graph, consider the set of genotypes in Figure 1a, which corresponds to the compatibility graph in Figure 1b.

We also point out that disconnected components of the compatibility graph are necessarily resolved by distinct haplotypes, therefore the optimal set of haplotypes is the union of the optimal sets of each disconnected subgraph. This property is exploited in a specific preprocessing phase of our algorithm.

Another useful property, which is going to be used in our algorithms, is the following:

Proposition 1 (Haplotype complement). Given a genotype g and a haplotype $h \mapsto g$, there exists a unique haplotype k such that $h \oplus k = g$. The haplotype k is called the complement of h with respect to g and is denoted with $k = g \ominus h$.

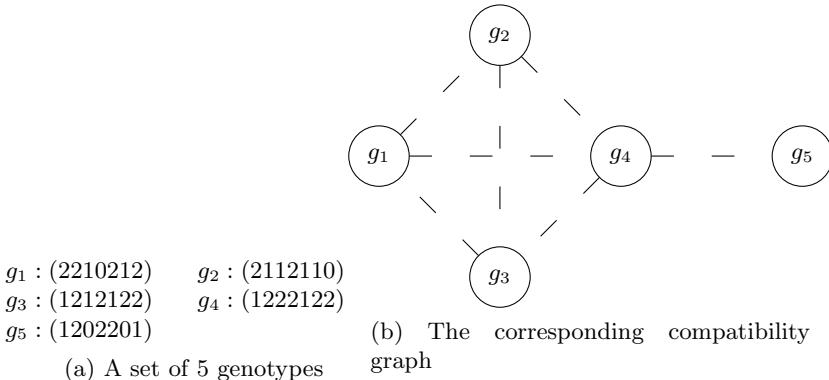


Fig. 1. An example of compatibility graph for a set of genotypes

3 Two-Level ACO for the Haplotype Inference Problem

The Haplotype Inference problem definition makes constructive procedures very promising. Indeed, a constructive procedure can incrementally build a set H of haplotypes which, taken in pairs, resolve the genotypes. Such a procedure can start from an empty set and add one or two haplotypes at a time, while it scans the set of genotypes G . The objective is to build H as small as possible, i.e., to find a minimal cardinality set of haplotypes that composes the phasing. To this aim, new haplotypes should be added to H only when necessary, i.e., when no pair of haplotypes already in H resolves the current genotype g .

The algorithm we propose is an instance of the Ant Colony Optimization metaheuristic [15] and is composed of two levels: the higher one employs an ACO for finding a good visiting order of genotypes while the lower level, also based on ACO, searches for the haplotypes to be added to H . The two levels are of course coupled, as the order in which genotypes are considered is influenced by the current set of haplotypes in H and, conversely, a generic step in the construction of H depends on the previously resolved genotypes.

Before applying the two-level ACO, the problem instance is preprocessed by a procedure that eliminates replicates among genotypes and identifies disconnected parts in the compatibility graph that can then be treated as independent instances.

In Algorithm 1 we provide the general search scheme that is going to be detailed in the following.

3.1 Preprocessing Phase

The instances of the Haplotype Inference problem can be reduced by analyzing their structure, while preserving the property that a solution to the reduced instance is a solution to the original one. The first preprocessing step consists in eliminating duplicated genotypes. Furthermore, the analysis of the structure of the compatibility graph enables us to identify independent sub-instances. Indeed, the genotypes belonging to an isolated sub-graph, i.e., a disconnected

Algorithm 1. ACO-HI

```

1: A: set of ants; G: set of genotypes
2: Preprocessing phase
3: while terminating conditions not met do
4:   for all  $a \in A$  do
5:     while not all genotypes are resolved do
6:        $g \leftarrow \text{chooseNode}(G)$ 
7:       resolve genotype  $g$ 
8:       propagate resolvents
9:     end while
10:   end for
11:   pheromoneUpdate()
12: end while

```

component, identify a sub-instance that can be solved independently. Therefore, a solution to the original instance can be found by separately solving the sub-instances composing it. A special case of independent instance is represented by isolated nodes, i.e., genotypes that are not compatible with any other genotype. The contribution of such a genotype to the solution of the Haplotype Inference instance is composed by a pair of haplotypes that, by definition of compatibility, cannot be used to resolve any other genotype.

3.2 Lower Level: Genotype Resolution

As depicted in Algorithm 1, an ant a builds a solution by considering in turn each genotype $g \in G$ (the order is defined in the higher-level, see Sect. 3.3) and finding resolvent haplotypes for it. The basic heuristic for this phase consists in trying to resolve g with haplotypes already in H and add new haplotypes only if necessary. When a new resolvent has to be added to H , the values of its heterozygous sites are chosen on the basis of pheromone values.² For each (heterozygous) site j on genotype i , we have two pheromone components, $\tau_{i,j}^0$ and $\tau_{i,j}^1$, corresponding to values 0 and 1, respectively. The value assigned to the haplotype site is chosen with probability $p_{i,j}(v) = \tau_{i,j}^v / (\tau_{i,j}^0 + \tau_{i,j}^1)$, with $v \in \{0, 1\}$.

Excluding the case in which H already contains a pair of haplotypes resolving g , there are three different cases to be considered for the resolution of a genotype g in this step of the algorithm: (i) no resolving candidates in H , (ii) one candidate, (iii) more than one candidate. In the following, we detail the procedure defined for these cases:

Case (i): A haplotype h is built by a pheromone guided construction procedure, as previously described. Then, $k = g \ominus h$, the complement of h , is built and both are added to H . Then, these two new haplotypes are *propagated* along the compatibility graph in order to update the list of resolving candidates for the genotypes.

² Homozygous sites do not represent choice points as they are directly assigned because the haplotype we are constructing must resolve g .

Case (ii): When one resolving candidate is already available, its complement w.r.t. g is built and this step completed as in the previous case.

Case (iii): When there are two or more candidates that can resolve g , but no pair of them can resolve it, we have to choose one among these haplotypes. We implement this operation by iteratively considering each site and applying the following procedure: if, among the candidates, the homologous sites have different values (i.e., at least in a pair there are both values 0 and 1) one of the two is chosen probabilistically (using pheromone values) and all the candidates with a different value are discarded. The procedure ends when only one candidate is left and the final steps of the previous cases are performed again.

The algorithm, named *ACO-HI*, can be further improved by slightly modifying the procedure implemented for cases *(i)*. In fact, since the new haplotype added to H must resolve the current genotype g , a heuristic bias toward the construction of a haplotype that also resolves another genotype compatible with g can be beneficial. Thus, the genotype g' that has to be visited after g is determined by the higher level and a haplotype is probabilistically constructed (as in the original procedure) that not only resolves g , but also g' . Therefore, the number of sites to be assigned on the basis of the pheromone values is restricted to the set of sites which are ambiguous in both the genotypes g and g' . In this way, haplotype construction is still guided by pheromone only, but a simple kind of heuristic criterion is introduced to avoid building a new haplotype compatible only with genotype g . A similar procedure is also applied, with slight modifications, also in case *(iii)*. We will refer to the improved version of *ACO-HI* as *ACO-HI⁺*.

3.3 Higher Level: Genotypes Visiting Order

The order in which genotypes are visited has a strong influence on solution quality, therefore the higher level of the algorithm tries to learn a good genotype visiting order. This learning mechanism is primarily guided by pheromone associated to the edges of the compatibility graph. In this way, pairs of consecutive genotypes in the series are learnt. It would be possible to learn larger building blocks, such as triplets, but we decided to limit the case to pairs because of efficiency reasons. Formally, every edge (i, j) of the compatibility graph is associated to a pheromone value τ_{ij} and the probability to move from node i to node j is given by:

$$p(i, j) = \begin{cases} \frac{\tau_{ij}}{\sum_{l \in adj(i)} \tau_{il}}, & \text{if } j \in adj(i) \\ \frac{1}{|U|}, & \text{if } j \in U \wedge adj(i) = \emptyset \\ 0, & \text{otherwise} \end{cases} \quad (4)$$

where $adj(i)$ is the set of nodes adjacent to i (i.e., the compatible genotypes) still unresolved and U is the set of currently unvisited genotypes not compatible with genotype corresponding to i . In such a way, if i has adjacent unresolved nodes, then one among them is chosen according to pheromone values; otherwise,

the next genotype in the sequence is chosen randomly among the remaining unresolved genotypes.

3.4 Pheromone Update

Our algorithm is implemented according to the Hyper-cube framework [16]. The objective function of the problem is the cardinality of H , that has to be minimized. Therefore, as a quality function used for the reinforcement, we chose the function $F(H) = 2n - |H|$. Pheromone is updated in the two levels with the same evaporation parameter and quality function. The only difference is that the solution components of the higher level are edges of the compatibility graph, while in the lower level they are nodes representing values to assign to haplotype sites.

4 Experimental Analysis

With the aim of understanding the contribution of each algorithmic component, we compared *ACO-HI* and its improved version, *ACO-HI⁺*, against two versions of the algorithm with the learning mechanism disabled: *ACO-HI-random* that chooses the sequence of genotypes to be visited randomly and *ACO-HI^{+(no learning)}* that is a version of *ACO-HI⁺* equipped only with heuristic haplotype construction (ties are broken randomly) and random sequence of visited genotypes.

The sets of instances chosen for the experimental analysis are the common benchmark used in previous works and they are composed of two parts. The first one, composed of the sets Harrower uniform and Harrower hapmap, is the benchmark used in [7]. The second part of the instances, namely Marchini SU1, Marchini SU2, Marchini SU3 and Marchini SU-100kb, were taken from the website <http://www.stats.ox.ac.uk/~marchini/phaseoff.html>. The main features of the instance sets are summarized in Table 1.

Table 1. A summary of the main features of the benchmarks

Benchmark set	Number of instances	Number of genotypes	Number of sites
Harrower uniform	200	10÷100	30÷50
Harrower hapmap	24	5÷68	30÷75
Marchini SU1	100	90	179
Marchini SU2	100	90	171
Marchini SU3	100	90	187
Marchini SU-100kb	29	90	18

We present a comparison of the different versions of ACO in Figure 2, in which statistics on solution quality are plotted. The boxplots represent statistics over 10 independent runs of the algorithms on all the instances of each set. The solution value considered for the statistics is the sum of solutions returned on all

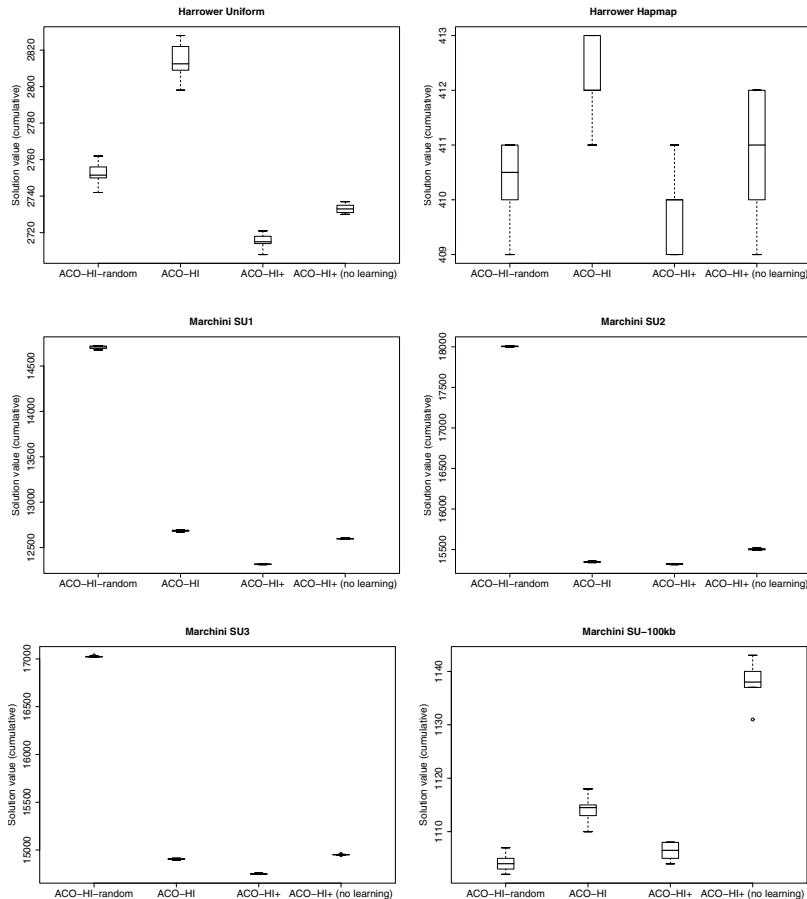


Fig. 2. Comparison of the main ACO variants w.r.t. solution quality

the instances of each benchmark.³ Algorithms are stopped after 300 iterations of the main construction loop; this value of maximum iterations is set in such a way that the algorithm reaches stagnation. Pheromone evaporation has been set to 0.1, according to a *brute force* analysis over a representative sample of the instances. The algorithms have been implemented in C++ and run on a 1GHz Intel Core Duo with 2GB of RAM and Linux Ubuntu 7.10 (kernel 2.6.22).

We can observe that, except for the set Marchini SU-100kb, *ACO-HI⁺* is superior to the other variants and the synergy between its constructive heuristic and learning mechanism based on pheromone (both for higher and lower levels) is quite effective. We conjecture that the good performance of *ACO-random* on the set Marchini SU-100kb is caused by the structure of the instances; indeed,

³ Detailed results are omitted because of lack of space and are available from the authors upon request.

Table 2. Cumulative statistics on the running time of algorithm $ACO-HI^+$. The total running time (in seconds) for solving all the instances of each set is considered.

Benchmark set	Min.	1st Q.le	Median	Mean	3rd Q.le	Max.
Harrower uniform	54.00	55.50	65.00	64.30	72.75	75.00
Harrower hapmap	13.00	21.00	27.00	30.40	34.75	59.00
Marchini SU1	1634	1743	1808	1797	1861	1948
Marchini SU2	466.0	516.2	538.0	533.9	546.8	584.0
Marchini SU3	1401	1452	1488	1487	1541	1549
Marchini SU-100kb	148.0	155.0	169.5	165.8	174.8	182.0

Table 3. Solution quality of $ACO-HI^+$ and local search [13] w.r.t. optimal solution values

Benchmark set	<i>rpol</i>	Sum of solution values (Perc. error)	
		<i>ACO-HI⁺</i>	<i>HI-Tabu search [13]</i>
Harrower uniform	2689	2694 (0.186)	3252 (21.0)
Harrower hapmap	321	321 (0.0)	343 (6.854)
Marchini SU1	2453	2483 (1.223)	3456 (40.89)
Marchini SU2	14794	15102 (2.081)	17735 (19.88)
Marchini SU3	2113	2121 (0.379)	2333 (10.41)
Marchini SU-100kb	661	667 (0.009)	755 (14.22)

most of these instances have a very sparse compatibility graph and this characteristic makes the high level learning component much less effective, maybe even misleading for the search.

For lack of space, we omit detailed data on running times and we just report the cumulative statistics on the running time of algorithm $ACO-HI^+$ in Table 2, in which the running time to the best solution, in seconds, is considered.

5 Comparison Against the State of the Art

Algorithm $ACO-HI^+$ has a very good performance, both in terms of quality and time. In this section we compare its performance against the state-of-the-art exact solver, in order to assess its effectiveness. To the best of our knowledge, the best complete solver for the Haplotype Inference problem is *rpol* [12]. We run the solver on the same benchmark instances and on the same machine. We allotted *rpol* 24 hours of computation for each instance. The instances of the set Harrower uniform, Harrower hapmap, Marchini SU1 and Marchini SU2 were completely solved. From Marchini SU3 only 89 over 100 instances were solved and from Marchini SU-100kb were solved 23 over 29 instances. Overall, most of the instances could be solved with a runtime higher than 12 hours. In Table 3 we provide the comparison between the solutions returned by $ACO-HI^+$ and the optimal one provided by *rpol* (when available) for each benchmark. The solution values have been summed up over the instances composing the set (for $ACO-HI^+$

we considered the best among the 10 runs) and also the error w.r.t. the sum of optimal solutions is reported. We can observe that *ACO-HI⁺* achieves a very good performance in terms of solution quality, as the error w.r.t. the optimum is rather small. Furthermore, *ACO-HI⁺* also compares quite favourably with the state-of-the-art local search for *HI_{pp}* [13] in terms of overall solution quality.⁴

6 Conclusions and Future Work

We have presented an adaptive constructive approach for the Haplotype Inference problem under the pure parsimony criterion, which relies on a two-level ACO procedure for determining first the genotype to be resolved and then choosing the haplotypes to resolve it.

The experimental evaluation of the algorithm has shown that the algorithm is very effective for solving medium- to large-scale instances of the problem on common benchmarks and that its running time scales well with the dimension of the instances.

In future developments we plan to enhance the behavior of the ACO metaheuristic by testing hybrid approaches. A possible research direction we intend to pursue is to couple the ACO with a Local Search procedure [13] with the aim improving the solution found by each ant. Other possibilities include the replacement of the ACO for genotype ordering with other metaheuristics, such as evolutionary algorithms or again local search.

Acknowledgments. We thank Inês Lynce and Ana Sofia Graça for kindly providing us their instances and solvers, and we also thank Ian M. Harrower for sending us his datasets.

References

1. The International HapMap Consortium: A haplotype map of the human genome. *Nature* 437 (2005)
2. The International HapMap Consortium: The international HapMap project. *Nature* 426, 789–796 (2003)
3. Gusfield, D.: Haplotype inference by pure parsimony. In: Baeza-Yates, R., Chávez, E., Crochemore, M. (eds.) CPM 2003. LNCS, vol. 2676, pp. 144–155. Springer, Heidelberg (2003)
4. Clark, A.G.: Inference of haplotypes from PCR-amplified samples of diploid populations. *Molecular Biology and Evolution* 7, 111–122 (1990)
5. Halldórsson, B.V., Bafna, V., Edwards, N., Lippert, R., Yooseph, S., Istrail, S.: A survey of computational methods for determining haplotypes. In: Istrail, S., Waterman, M.S., Clark, A. (eds.) DIMACS/RECOMB Satellite Workshop 2002. LNCS (LNBI), vol. 2983, pp. 26–47. Springer, Heidelberg (2002)

⁴ We omit the complete direct comparison of the two techniques because of limited space and we refer the reader to [13].

6. Lancia, G., Pinotti, M.C., Rizzi, R.: Haplotyping populations by pure parsimony: Complexity of exact and approximation algorithms. *INFORMS Journal on Computing* 16(4), 348–359 (2004)
7. Brown, D.G., Harrower, I.M.: Integer programming approaches to haplotype inference by pure parsimony. *IEEE/ACM Transactions on Computational Biology and Bioinformatics* 3(2), 141–154 (2006)
8. Kalpakis, K., Namjoshi, P.: Haplotype phasing using semidefinite programming. In: *BIBE*, pp. 145–152. IEEE Computer Society, Los Alamitos (2005)
9. Huang, Y.T., Chao, K.M., Chen, T.: An approximation algorithm for haplotype inference by maximum parsimony. In: *Proceedings of the 2005 ACM Symposium on Applied Computing (SAC 2005)*, pp. 146–150. ACM, New York (2005)
10. Lynce, I., Marques-Silva, J.: SAT in bioinformatics: Making the case with haplotype inference. In: Biere, A., Gomes, C.P. (eds.) *SAT 2006. LNCS*, vol. 4121, pp. 136–141. Springer, Heidelberg (2006)
11. Lynce, I., Marques-Silva, J.: Efficient haplotype inference with boolean satisfiability. In: *Proceedings of the 21st National Conference on Artificial Intelligence and the Eighteenth Innovative Applications of Artificial Intelligence Conference*. AAAI Press, Menlo Park (2006)
12. Graça, A., Marques-Silva, J., Lynce, I., Oliveira, A.L.: Efficient haplotype inference with pseudo-boolean optimization. In: Anai, H., Horimoto, K., Kutsia, T. (eds.) *Ab 2007. LNCS*, vol. 4545. Springer, Heidelberg (2007)
13. Di Gaspero, L., Roli, A.: Stochastic local search for large-scale instances of the haplotype inference problem by pure parsimony. *Journal of Algorithms in Logic, Informatics and Cognition* (2008) doi:10.1016/j.jalgor.2008.02.004
14. Wang, R.S., Zhang, X.S., Sheng, L.: Haplotype inference by pure parsimony via genetic algorithm. In: *Operations Research and Its Applications: the Fifth International Symposium (ISORA 2005)*, Tibet, China, August 8–13. Lecture Notes in Operations Research, vol. 5, pp. 308–318. Beijing World Publishing Corporation, Beijing (2005)
15. Dorigo, M., Stützle, T.: *Ant Colony Optimization*. MIT Press, Cambridge (2004)
16. Blum, C., Dorigo, M.: The hyper-cube framework for ant colony optimization. *Transactions on Systems, Man, and Cybernetics – Part B* 34(2) (2004)

What Hides in Dimension X? A Quest for Visualizing Particle Swarms

Namrata Khemka and Christian Jacob

Evolutionary and Swarm Design Group, Dept. of Computer Science
University of Calgary, Alberta, Canada
`{nkhemka, cjacob}@ucalgary.ca`

Abstract. The way we perform evolutionary experiments is all influenced by visualizing multi-dimensional solutions, analyzing the extent to which the search space is explored, displaying the gross population statistics, determining clustering and building blocks, and finding successful combinations of parameter values. Through visualization we can gain valuable insights to enhance our knowledge about particle swarm optimizers, in particular, and the search space that is being explored. In this paper, we focus on different visualization techniques for particle swarm systems. We investigate the advantages of a range of graphical data representation methods by example of the two- and four-dimensional sphere function, the two-dimensional simplified foxholes function, and a 56-dimensional real-world example in the context of muscle stimulus patterns.

1 Introduction

A picture can be worth much more than a thousand words. One benefit of visualization methods is to communicate ideas universally. However, another important purpose of a picture is to provide means to create and discover the idea itself [1]. Using visualization techniques, we can assemble thousands of individuals of Particle Swarm Optimization (PSO) [2] into ‘pictures’, thereby revealing hidden patterns such as building blocks and clusters. The main purpose of visualization is to gain insights, discovering new patterns, offering knowledge about the explored regions within the n -dimensional search spaces, determining the successful combinations of parameter values, finding the parameter values where the population has converged, or understanding how partial solutions are created.

In the past, we have worked on a real-world problem called the “Soccer Kick Simulation” [3] [4], where the goal was to find optimized settings (via Particle Swarm Optimization) for control parameters for a kinematic model of 17 leg muscles, such that a kicked ball travels as far and as fast as possible. The model included 56 parameters (dimensions). In the course of our investigations and experiments we realized the importance of visualization in order to efficiently and successfully tackle such a large-scale optimization problem. The soccer kick simulation (or in general any real-world system) produces a large number of

potential solutions (over 120,000 individuals) each having 56 dimensions. Monitoring and making sense of such large groups of dynamic real-time data poses various challenges. Presenting this information to the user as raw data (a series of numbers) would make it difficult if not impossible for the user to observe and understand the progression of the search algorithm. Normally, we evaluate the quality of the optimizers by the fitness of their solutions generated. A similar approach was taken for the soccer kick simulation, where we focused on the overall performance and gross population statistics, such as comparing the solutions found through different runs, the number of fitness function evaluations that were required to find a good solution, and the convergence rate of the algorithms. This methodology is a “black box” approach that solely focuses on the actual outcome, but mainly ignores the behavior of the algorithms.

Visualization methods can help us to analyze, in great depth, the potential solutions and results discovered. Visualization of particle swarm algorithms can therefore allow the user to make inferences that are not easy to accomplish otherwise. Visualizing multi-dimensional individuals, analyzing the extent to which an algorithm has explored the search space, the effects of inherent parameters, analyzing gross population statistics, determining clusters and building blocks, and finding the successful combination of parameter values can influence our choice of experiments. We also gain further insights into particle swarm systems and the search space of an optimization task in general. In this paper we explore and introduce the relevance of visualization techniques for particle swarm systems that investigate the questions touched upon above. The rest of the paper is organized as follows. We discuss the background work regarding visualization of evolutionary and swarm systems in Section 2. Section 3 introduces our visualization techniques on two benchmark functions and the soccer kick simulation. We also examine and analyze the results of the visualization techniques in Section 3. Finally, we conclude our work in Section 4.

2 Related Work

An overview of the most commonly used summary graphs (such as the convergence plots) is provided by Pohlheim [5], whose visualization toolbox helps to observe both the “course” and the “state” of an evolutionary algorithm. These visualizations include the fitness of individuals, distances between individuals, and certain statistics that can be tracked over a single generation or multiple experiments. The more advanced *GEATbx* [6] builds on Pohlheim’s work by providing summary graphs for visualizing the convergence behavior of evolutionary algorithms.

Attempts to visualize multiple dimensions has led to the development of ‘multi-dimensional scaling’ techniques that transform search spaces of multi-dimensional into lower dimensions. The most well-known technique in the realm of visualizing multi-dimensional population-based techniques is Sammon Mapping [5]. This method places points on a two-dimensional canvas, which represent

vectors in the higher dimensional space. The idea is to then iteratively move points closer together in the two-dimensional space, if they are close together in the multi-dimensional space. Although this technique works, it has quadratic time complexity in the number of points, and so can become demanding for large search spaces [7]. To overcome this problem, search space matrices [8] provide a technique which maps all possible individuals of a genetic algorithm onto a two-dimensional canvas, such that the Hamming distance between neighboring points is minimized. This mapping is simple and the algorithm scales linearly with the number of points in the search space [7]. We can also display multi-dimensional data through the use of glyphs. Chernoff faces [9] are an example of glyphs where data is represented by a sketch drawing of a human face. Each attribute in the data maps to different items on the face such as the mouth, nose, separation between eyes, etc. Although with Chernoff faces one can easily distinguish specific features in the data points, they do not scale well for larger dimensions.

GONZO, another visualization tool for genetic algorithms, was developed by Collins [8]. This system displays population summary graphs along with the genotype and parental information of an individual. *GONZO* displays the search space by using search space matrices. The key advantage of this system is that it allows for both online (while the genetic algorithm is in progress) and offline (after the genetic algorithm has completed) visualization, thereby supporting the user to interactively modify the inherent parameters.

Each of these visualization frameworks have their strengths. However, none of these systems provide an exploratory and inquiry platform for PSO. These systems do not answer any of the visualization questions related to gaining further insights into the performance of particle swarm optimizers. In the remainder of this paper, we introduce density and range plots, along with parallel coordinates for visualizing multi-dimensional data generated by PSO experiments.

3 Visualizing Particle Swarms: A Detective's Playground

Our investigations and experiments with the soccer kick simulation made us realize the importance of visualization in order to solve large-scale optimization problems. A range of graphical methods can aid the user to analyze gross population statistics, determine clusters and building blocks, track successful combinations of parameter values, find the extent to which the search space is explored, and visualize multi-dimensional data produced by a PSO. In this section, we first discuss phenotype plots (Sect. 3.1) that are restricted to three-dimensions along with the common fitness curves (Sect. 3.2). Using two-dimensional benchmark functions (Table 1) such as sphere (simple, symmetric, smooth, unimodal) and foxholes (multimodal and modified such that all peaks have the same height) we introduce density plots (Sect. 3.3), parallel coordinate plots (Sect. 3.4), and range plots (Sect. 3.5).

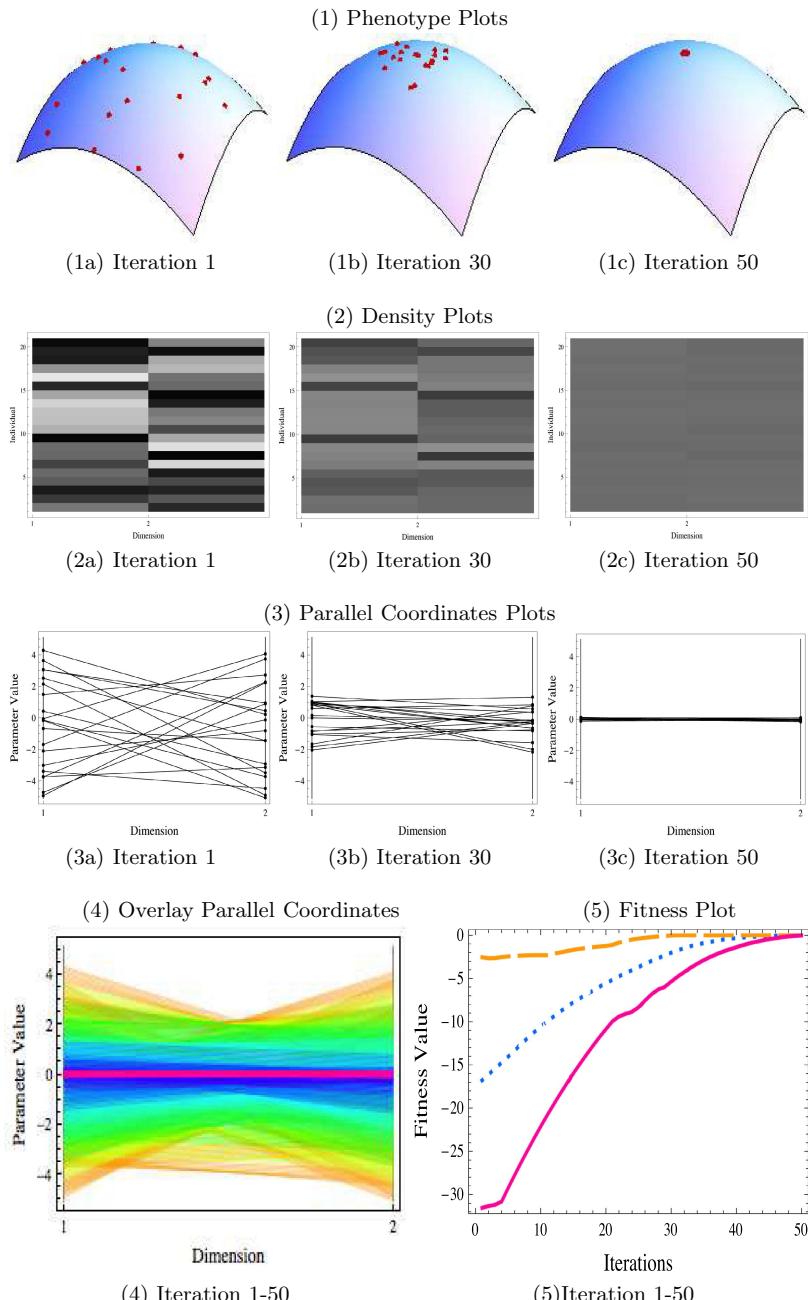


Fig. 1. Snapshots from a PSO evolution over the two-dimensional sphere function. (1) Phenotype plots at iteration 1, 30, and 50. (2) Density plots at iteration 1, 30, and 50. (3) Parallel coordinates plots at iteration 1, 30, and 50. (4) Overlay Parallel Coordinates. (5) Fitness plot (solid line-worst, dotted line-average, long dashed line-best).

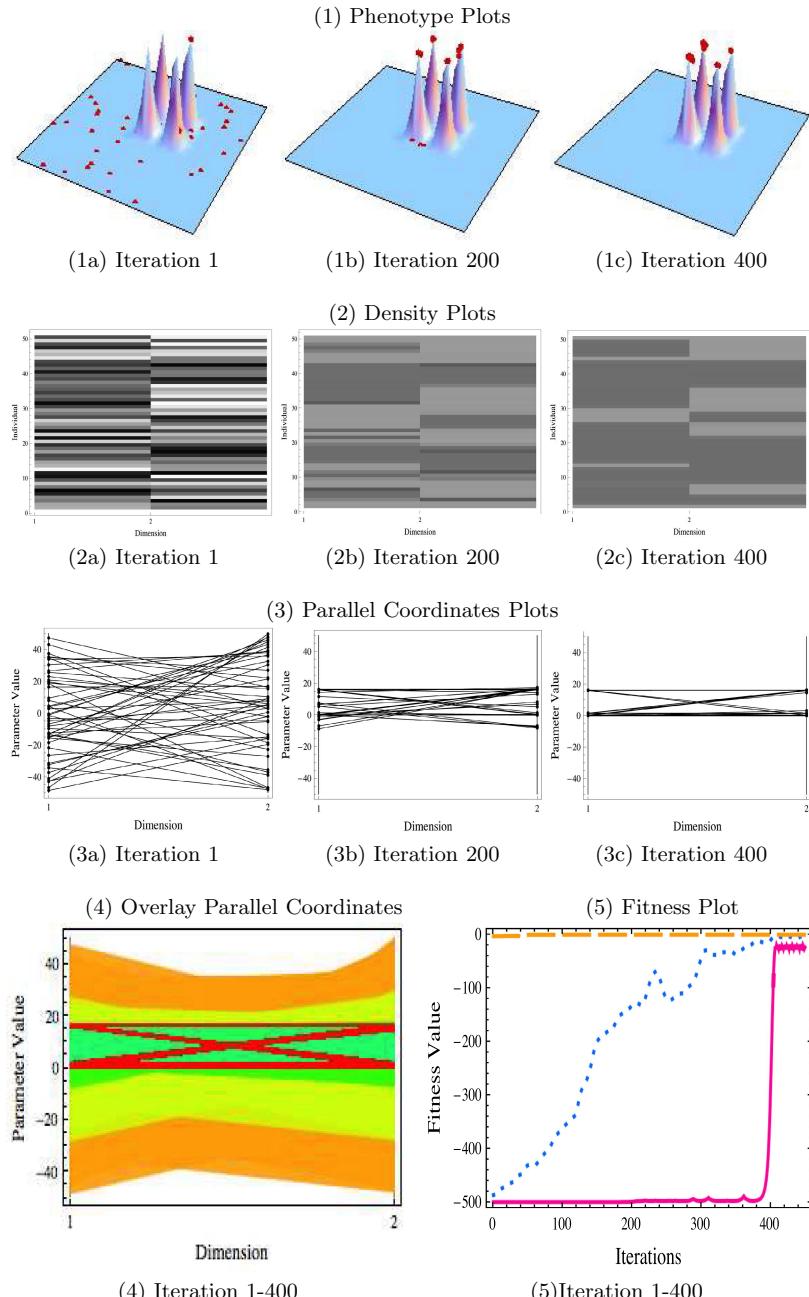


Fig. 2. Snapshots from a PSO evolution over the two-dimensional foxholes function. (1) Phenotype plots at iteration 1, 200, and 400. (2) Density plots at iteration 1, 200, and 400. (3) Parallel coordinates plots at iteration 1, 200, and 400. (4) Overlay Parallel Coordinates. (5) Fitness plot (solid line-worst, dotted line-average, long dashed line-best).

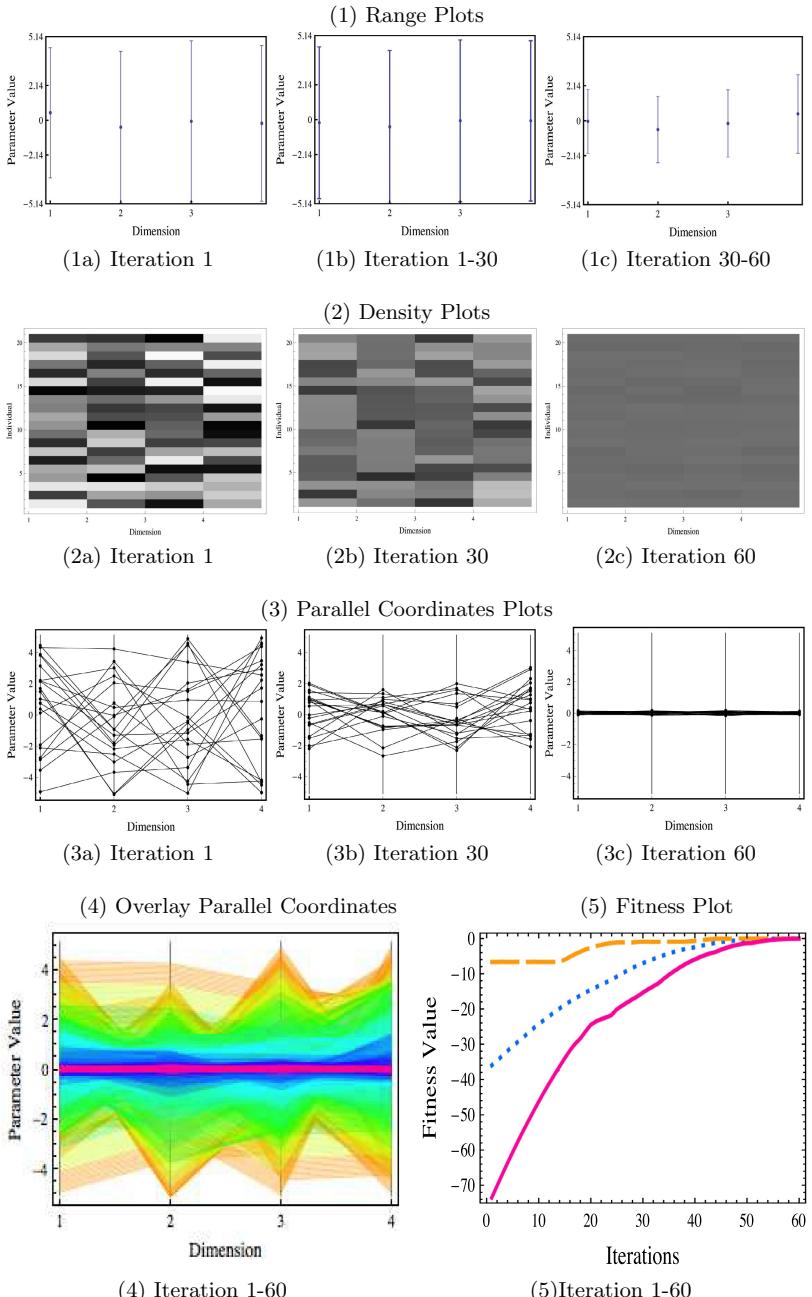


Fig. 3. Snapshots from a PSO evolution over the four-dimensional sphere function. (1) Phenotype plots at iteration 1, 30, and 60. (2) Density plots at iteration 1, 30, and 60. (3) Parallel coordinates plots at iteration 1, 30, and 60. (4) Overlay Parallel Coordinates. (5) Fitness plot (solid line-worst, dotted line-average, long dashed line-best).

Table 1. Benchmark functions used in the experiments are denoted in column 1. The range column states the value each dimension j of a function is valid between, and the last column shows the best reported optimal value a function has. Note that we are maximizing each of these functions.

Function	Range	$f(\vec{x}^*)$
Sphere: $f_1 = -\sum_{j=1}^d x_j^2$	$[-5.12, 5.12]$	0
Foxholes: $f_2 = (0.002 + \sum_{j=1}^{Length[a]} \frac{1}{j + \sum_{i=1}^{Length[x]} (x[[i]] - a[[j]][[i]])^6})^{-1}$ a = 0, 0, 0, 16, 16, 0, 16, 16	$[-50, 50]$	0

3.1 Phenotype Plots: Where Are the Particles Heading?

Figures 1 and 2 show examples of the population dynamics resulting from particle swarms over a number of iterations applied to two typical benchmark functions for particle swarm optimizers (Table 1). The particles are represented as dots. The behavior of the particles is seen at different iterations, making it easy to compare and contrast the movement of the individuals and study their behavior. These phenotype plots also provide information on finding multiple solutions, as seen in Figure 2, where the particles have discovered all four peaks. Although these graphs provide visual clues on the performance of the algorithm, they are limited to two-dimensional search problems, which are only of minor practical relevance.

3.2 Fitness Curves: Any Improvement in the Algorithm over Time?

A widely used and straight-forward form of visualization in population-based methods consists of two-dimensional population statistics graphs. These include the fitness of the best individual, the average fitness value of all individuals, and the worst fitness values at each time step (Fig(s). 1, 2, and 3). These plots give a good indication of whether an algorithm is improving over time and provide information on the overall behavior, such as the convergence and divergence of the algorithm. However, these plots do not provide any information regarding the parameter values or the convergence of the parameter values.

3.3 Density Plots: Are the Particles Converging?

Density plots such as the ones in Figures 1, 2, and 3 further illustrate whether a population has converged. However, they go a step further; they can even tell us the values towards which the parameters have converged over time. Figure 1 demonstrates this idea, where the parameter (dimensional) values for each individual are plotted as a gray-scale rectangle. For example, the rectangle at the lower left corner of the plot (Fig. 1) is associated with the value of the first particle at the first dimension. We observe that at iteration 1 (Fig(s). 1, 2, and 3) the values are uniformly random, indicating that we have diversity in our population. Over time all the parameter values have converged to a particular value indicated by the shared gray-scale for each individual (Fig(s). 1 and 3).

A different effect is illustrated in Figure 2. Not all individuals have converged to a particular value as there are four distinct peaks with the same or very similar fitness values. We do see combinations of values (0,0), (0,1), (1,0), and (1,1) represented by their respective gray-scale patterns. This is in sync with the phenotype plots for this particular experiment, where the individuals have converged to four different peaks (Fig. 2).

3.4 Parallel Coordinates: Are There Any Patterns, Trends, and Clusters Among the Particles?

Parallel Coordinates are a two-dimensional visual representation proposed by Inselberg [10] as a way to represent general multi-dimensional data sets. In a parallel coordinates visualization system, the d -dimensional structure of an individual is projected onto the two-dimensional space of the graphical window (monitor) through a set of d vertical parallel axes. A particular dimension of an individual corresponds to one vertical axis. All the values associated with the j^{th} dimension are plotted on the j^{th} axis, $j \in 1, \dots, d$. All the points that visualize the components of the i^{th} individual are connected by a polygonal line, that is a polyline, as illustrated in Figures 1, 2, and 3.

Parallel coordinates can be very useful for visualizing the individuals of particle swarm algorithms. A structured overview of the individuals can be displayed, thus allowing us to recognize patterns, identify trends, and establish relationships among the dimensions of various individuals of a population (or experiment(s)). This is an important visualization aspect, as individuals represent potential solutions to the optimization problem and are the most important elements of a particle swarm algorithm. Using parallel coordinates we can visualize the structure of individuals at a particular time-step (such as in Fig. 1), thus providing insights into the algorithm's progress at a particular moment in time. One also gains an overall picture of all individuals during an entire experiment. The overlaid parallel coordinates plot in Figures 1, 2, and 3 are colored, where the lighter values indicate those regions that were covered initially during the search, and the darker colors represent the parallel coordinates with polylines towards the end of the simulation.

Parallel coordinates also provide information about whether multiple solutions were discovered. In the foxholes example, we have peaks of the same height. Particle swarms find all four peaks, nicely illustrated by the parallel coordinates plot in Figure 2, where the four-dimensional combinations are depicted by four polyline groups.

3.5 Range Plots: What Are the Parameter Ranges?

The parameter (dimension) range changes during the course of a PSO run can be observed via range plots. In Figure 3, we display the minimum and the maximum values (i.e., a range) for each parameter of the initial population at iteration 1. These ranges are depicted by vertical bars. One can also visualize the ranges for all individuals over a certain number of iterations as illustrated in Figures 3

and 3. In Figure 3 the vertical bars are smaller, thus indicating convergence behavior of the parameter values.

3.6 An Application Example: Soccer Kick Simulation

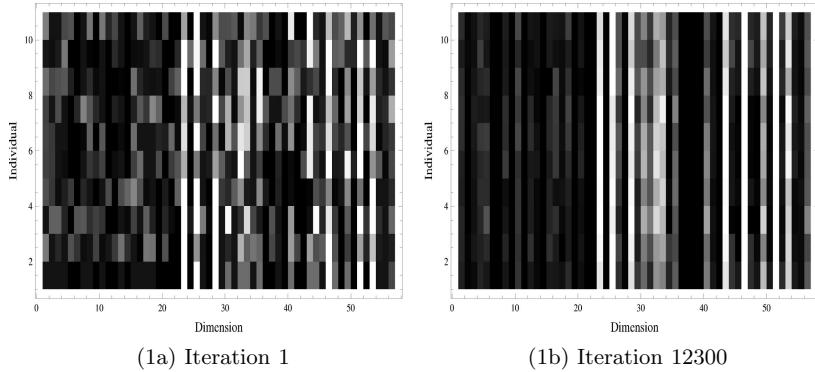
Kicking the ball is one of the most important skills required for playing soccer since the arms and hands are not allowed to touch the ball. The motion of the leg kicking the ball involves 17 different muscle groups in the foot and toes, talus, thigh, shank, etc. In a kinematic model, the 17 muscles together with the coordinates of the ball result in a 56-dimensional search problem [3]. Particle swarm optimization was used to optimize the modeled leg movement, such that when the foot touches the ball, high ball velocity is obtained. The results obtained from the PSO experiments were compared to those achieved from a simple Evolution Strategies (ES) algorithm [11]. During these experiments we particularly realized the importance of visualization tools. In the remainder of this section we discuss the visualization techniques introduced above in the context of this soccer kick simulation.

We conducted an experiment with PSO using the same initial individual as in the ES experiment. Since the population size for the PSO experiment was ten, nine new individuals were mutated from the initial ES individual. The first 23 parameters have values in a lower range than the rest, as is visible in both the density and parallel coordinates plots in Figure 4. This is observed in the vertical bars having similar grayscale shades (such as dimensions 1-20). Figure 4 represents time step 12,300, in which one observes that certain parameters are displayed as single-color columns, indicating that these parameters have been locked at particular values.

It is often difficult to understand why an algorithm is successful in generating solutions of high fitness. By looking at the parallel coordinates visualization for the soccer kick (Fig. 4.2), one can determine the successful ranges for the parameter values, i.e., the range of values on a polyline. These polylines can be considered as a means of estimating the distribution of the explored solutions relative to the entire search space. This can further help us determine the regions of the search space where the individuals of particle swarms have become stuck in local peaks. With the parallel coordinates plots we can also identify those regions where the particle swarm optimizer has found a local peak on the search space. The identified ranges of values for each dimension can further narrow down our search space by adding constraints to the fitness function. Parallel coordinates can be relatively simple and can be very useful in providing information about how the particle swarm algorithm traverses the search space, and aid in detecting trends that may suggest convergence or divergence.

Another variant of the range plots was created for the soccer kick simulation (Fig. 5), where the vertical bars represent the range for each of the 56-dimensions, over all iterations, limited to all those solutions that have a fitness above a certain threshold (τ). The range plots graphically reveal the subset of the search space and the successful combination of parameters that yields a high fitness value for the soccer kick simulation. For example, Figure 5a represents all the individuals

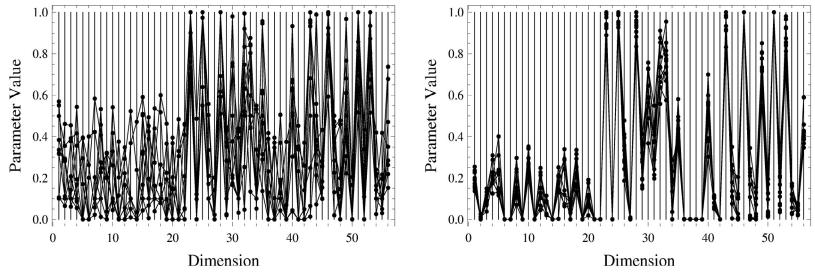
(1) Density Plots



(1a) Iteration 1

(1b) Iteration 12300

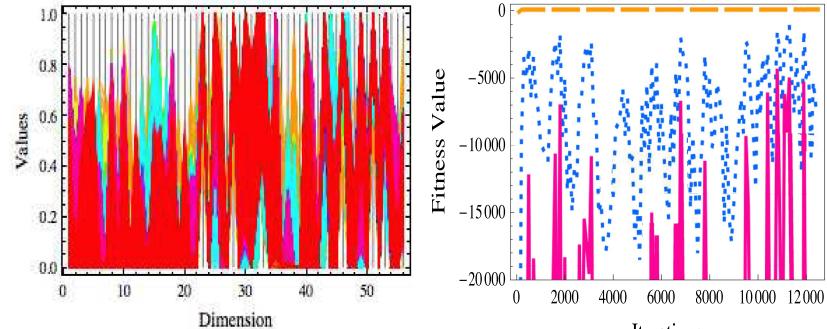
(2) Parallel Coordinates Plots



(2a) Iteration 1

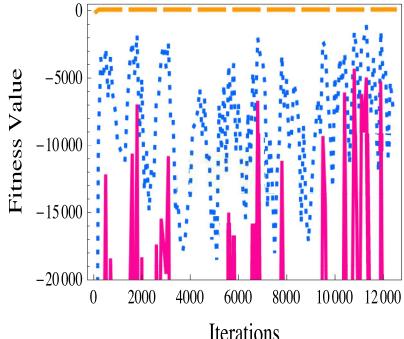
(2b) Iteration 12300

(3) Overlay Parallel Coordinates



(3) Iteration 1-12,300

(4) Fitness Plot



(4) Iteration 1-12,300

Fig. 4. Plots for the 56-dimensional soccer kick simulation. (1) Density plots at iteration 1 and 12,300. (2) Parallel coordinates plots at iteration 1 and 12,300. (3) Parallel coordinates for all generations. (4) Fitness plot (solid line-worst, dotted line-average, long dashed line-best).

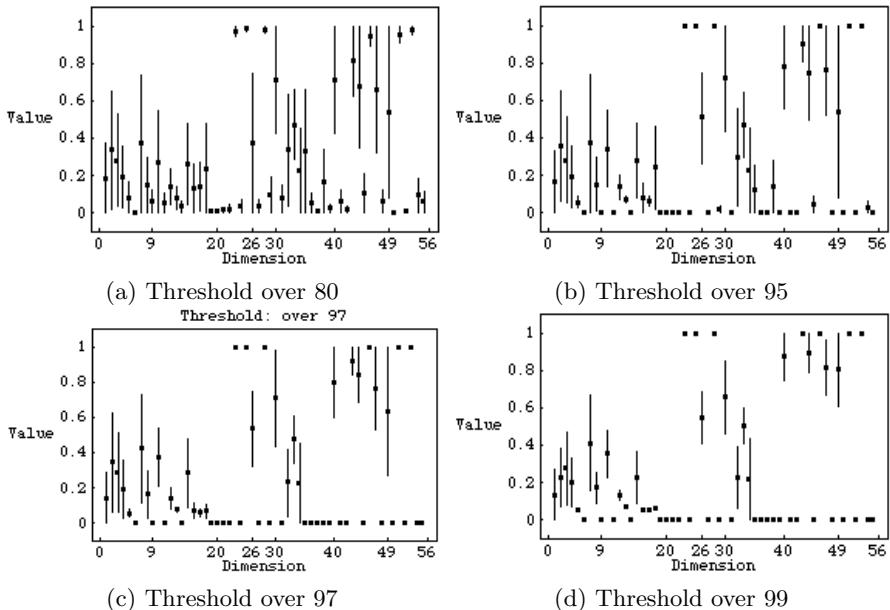


Fig. 5. Insights from the range plots for the soccer kick simulation The plots show the location intervals for individuals with fitness over a certain threshold. (a) Threshold: 80. (b) Threshold: 95. (c) Threshold: 97. (d) Threshold: 99.

that have their fitness over 80, and Figure 5d includes all individuals with a fitness value over 99 (the maximum ball speed obtained is 99.39). These plots suggest two things:

1. For successful individuals, the range of the parameters decreases over time. Consider parameter 49, for example. The value of this parameter is initially varied between 0.1 and 1. Over time, a significant reduction of the parameter space occurs, and the interval length shrinks to at least a third (0.6 and 1) for individuals having a fitness over 99. A similar pattern is observed for parameter 26.
 2. We also observe that some of the parameter values get completely locked in. As the fitness increases, more of these parameter values have a certain maintained value. For example, parameter 9 was in the range of 0 and 0.123214. For individuals with a fitness value of over 95, the value of this parameter is 0. Also the majority of the parameters for individuals with a higher fitness are locked in at either 0 or 1.

4 Conclusion

Graphically it is much easier to find patterns and visual cues that show relations among the parameters, clusters in the data, or successful combinations of parameters in the data sets generated by particle swarm optimizers. Our exploratory

and inquiry platform also lets the users visualize multi-dimensional individuals of the particle swarms and analyze the extent to which the search space is explored. We can obtain overall gross population statistics, such as the convergence or divergence of the particle swarm optimizer.

For future work, we expand our PSO visualization platform to allow the user to analyze results from a single experiment to a series of experiments. We will also work on creating online data visualization systems for particle swarms, so that interactive modification and exploration of parameter spaces is possible. For further information about our visualization tools visit <http://www.swarm-design.org/visualization>.

References

1. Card, S.K., Mackinlay, J.D., Shneiderman, B. (eds.): *Readings in information visualization: using vision to think*. Morgan Kaufmann, San Francisco (1999)
2. Kennedy, J., Eberhart, R.: Particle swarm optimization. In: *Sixth International Symposium on Micromachine and Human Science* (1995)
3. Khemka, N.: Comparing particle swarm optimization and evolution strategies: benchmarks and application. Master's thesis, University of Calgary (2005)
4. Khemka, N., Jacob, C., Cole, G.: Making soccer kicks better: a study in particle swarm optimization and evolution strategies. *IEEE Transactions on Evolutionary Computation (CEC)* (2005)
5. Pohlheim, H.: Visualization of evolutionary algorithms - set of standard techniques and multidimensional visualization. In: *Genetic and Evolutionary Computation Conference (GECCO)* (1999)
6. Pohlheim, H.: Geatbx: genetic algorithm toolbox for use with matlab (1998), <http://www.geatbx.com/index.html>
7. Hart, E., Ross, P.: Gavel - a new tool for genetic algorithm visualization. *IEEE Transactions on Evolutionary Computation (CEC)* (2001)
8. Collins, T.D.: Understanding evolutionary computing: a hands on approach. In: *IEEE Congress on Evolutionary Computation (CEC)* (1998)
9. Jacob, C.: *Illustrating Evolutionary Computation with Mathematica*. Morgan Kaufmann, San Francisco (2001)
10. Inselberg, A.: Multidimensional detective. In: *IEEE Symposium on Information Visualization (InfoVis)* (1997)
11. Cole, G., Gerritsen, K.: Influence of mass distribution in the shoe and plate stiffness on ball velocity during a soccer kick. *Adidas-Salomon AG* (2002)

A Dynamic Swarm for Visual Location Tracking

Marcel Kronfeld, Christian Weiss, and Andreas Zell

Computer Science Department, University of Tübingen, Tübingen, Germany
`{marcel.kronfeld,c.weiss,andreas.zell}@uni-tuebingen.de`

Abstract. The visual localization problem in robotics poses a dynamically changing environment due to the movement of the robot compared to a static image set serving as environmental map. We develop a particle swarm method adapted to this task and apply elements from dynamic optimization research. We show that our algorithm is able to outperform a Particle Filter, which is a standard localization approach in robotics, in a scenario of two visual outdoor datasets, being computationally more effective and delivering a better localization result.

1 Introduction

Environments with uncertainty like noise or dynamic changes are especially challenging for optimization [1]. A possible application for dynamic optimization is self-localization of mobile robots (Fig. 1), which need to know their position to interact with their environment in any useful way. Visual localization is based on an image database and therefore computationally expensive [2]. Moreover, the robot may move between any two iterations of the localization method, making the problem highly dynamic.

An up-to-date approach for localization in robotics is the so-called Particle Filter (PF), introduced in Sec. 3. A Particle Swarm Optimization method (PSO, [3]) for visual robot localization was described in [4] from the robotics point of view. This work goes into details of the PSO variant and analyzes parameter settings empirically. Closely related is the work of Vahdat et al. [5], who employed DE and PSO for global robot localization using laser sensors in an indoor environment. Their case treats a larger search space, where standard DE and PSO approaches showed to be superior to a Particle Filter. They did not, however, go into dynamics and thus handled only an initial phase of localization. From the robotics view, there have been several approaches extending Particle Filters with evolutionary concepts, e.g., [6]. Particle swarm algorithms have been extended to dynamic problems, e.g., by boosting diversity or creating sub-swarms to track optima [7].



Fig. 1. RWI outdoor robot Arthur

2 Particle Swarm Optimization

The PSO technique takes as basic idea the flocking behavior of animals. It searches the solution space by assigning velocities \mathbf{v} and a neighborhood relationship to the individuals \mathbf{x} . An individual is in each generation attracted to the best location in its history (\mathbf{p}^h) and to the best location found by its neighborhood (\mathbf{p}^n). The classical formulation is given in Eqs. 1 and 2.

$$v_i(t+1) = \omega v_i(t) + \phi_1 r_1(p_i^h - x_i) + \phi_2 r_2(p_i^n - x_i) \quad (1)$$

$$x_i(t+1) = x_i(t) + v_i(t+1) \quad (2)$$

The neighborhood type may range between small (local), randomized, and large (global), differing especially in the rate of information distribution. The parameters $\phi_{1/2}$ control the impact of the attractors, while $r_{1/2}$ are uniform random samples in $[0, 1]$ used as stochastic components. The inertness factor ω controlling the influence of former velocities is usually set < 1 for convergence.

The PSO concept seems to match problems with dynamically changing target functions, and dynamic adaptations of PSO have proven to be successful in this domain [8,7]. Yet, most work in the direction considers problems where the time between environmental changes is rather long, whereas, for mobile location tracking, the frequency of change is usually very high. When exploiting the rich visual data, the PSO method still offers a promising approach.

3 Visual Localization and SIFT

Visual localization addresses the question of how a mobile system, which has access to a visual representation of its environment (map), can find its position relative to this map and keep track of it during motion. As internal odometry, e.g., sensors measuring the speed of the wheels, is prone to errors, external information is necessary to obtain an initial position estimation and to achieve robust location tracking over time. Vision is a major source of information for humans, so it is appealing to use it for robot localization as well.

Visual localization may be divided into two stages. Firstly, for training, images are collected with position information, e.g. GPS tags, and stored in a database - the map. For later localization, a mobile robot moves through the same environment, takes pictures online, and relates them to the map. Visual data is very sensitive to changes, e.g. in light conditions, so localization requires a robust method for image comparison, which is apt to be time-consuming.

If the size of the reference image set is small, localization is possible by just comparing a new image to every image in the database and choose the best match as position estimation. This gets, of course, infeasible in larger scenarios. Using a probabilistic approach such as a Particle Filter, the set of tested images is reduced to the most probable ones.

A Particle Filter (PF, [9]) is a sequential Monte-Carlo-method for hidden state estimation, which approximates a probability density function p using a finite

set of weighted “particles” \mathbf{x}^i (not originally related to the particles in PSO). A particle can be seen as a hypothesis on the state of the system at a time, and the particle weights w^i express the *importance* of particles. The PF seeks to deduce from the collected sensor readings $s_{1:t}$ a belief in the current state \mathbf{x}_t of the robot: $p(\mathbf{x}_t | s_{1:t}) \approx \sum_{i=1}^n w_t^i \delta(\mathbf{x}_{1:t} - \mathbf{x}_{1:t}^i)$ (with Dirac δ).

In a PF iteration, a new proposal distribution is first predicted by advancing the particles using a transition model, e.g., from odometry readings. Then, the particles are reweighed incorporating new external information, judging how probable each hypothesis is. By resampling the particles using the updated weights, the PF concentrates on more promising areas of the state space. For an application using visual data, the weighing may be traced back to an image similarity measure. If the state-space is large, a PF requires many particles, e.g. $n = 300$ in [2], and much more for less distinctive sensory such as laser scanners [9,5]. PSO is, by contrast, known to be effective with relatively small swarms.

3.1 Interpretation in an Optimization Context

The aim of the PF method is, in terms of optimization: From a given sample set, produce a new set which contains samples of same or higher fitness with respect to the image similarity measure. As the system is mobile, beyond finding an optimum, we need to track it over time. Assuming that position \mathbf{x}_t is typically close to \mathbf{x}_{t-1} and associating particle velocities with the robot’s speed, we argue that PSO together with a distinctive image similarity measure is effective for localization. For such a distinctive function of two images, $m : S \times S \rightarrow \mathbb{R}$, we formulate a target fitness function for the map M :

$$f_M(x, t) = m(img_M(x), s(t)) \cdot pen(d(x, p_M(img_M(x)))), \quad (3)$$

where $img_M : X \rightarrow M \subset S$ returns the associated training image of a position \mathbf{x} , i.e., the training image closest to \mathbf{x} . $s(t)$ is the test image at iteration t , $p_M : M \rightarrow X$ delivers the known position for an image in the map. The penalty function $pen : \mathbb{R} \rightarrow \mathbb{R}$ reduces the fitness for particles far away from the training data, because localization is feasible only where there is training information. This is done similarly to [2] in terms of a Gaussian function. The problem of tracking a position now corresponds to a dynamic optimization problem: find the optimum $\hat{\mathbf{x}}$ of f_M at a time and follow it ensuring a plausible path.

A popular method to find and describe image features is Lowe’s Scale Invariant Feature Transform (SIFT) [10]. By using an image’s scale-space and assigning oriented features to the SIFT key points, they can be found relatively robust under changing views. The SIFT-match function m_{SIFT} delivers a value in $[0, 1]$ indicating image similarity by calculating the ratio of single feature matches to all possible matches and is used in the function f_M stated above.

4 A Dynamic PSO for Localization

We base our algorithm on the original PSO formulation [3] using the inertness parameter ω and a maximum velocity v_0 . Due to the dynamic tracking

requirement, the swarm is not to converge below a certain scale defined by the map. The fitness of an individual at time t is calculated using the SIFT similarity between the current test image and the training image closest to the individual's position (Eq. 3). As SIFT is relatively distinctive, we expect one strong main attractor most of the time and use the global neighborhood as swarm topology.

$$v_i(t+1) = \omega v_i(t) + \phi_0 r_0 \delta_i v_0 + \phi_2 r_2 (p_i^n(t) - x_i(t)), \quad (4)$$

$$x_i(t+1) = x_i(t) + v_i(t+1), \quad (5)$$

$$x_i^q(t+1) = p_i^n(t) + \delta_i N(0, \hat{q}_d). \quad (6)$$

In Eq. 4, we replaced the \mathbf{p}^h -component by a random term, because we expect a continuously changing environment where historically good positions lose relevance quickly. ϕ_0 is the weight of the random perturbation, while δ_i normalizes to the range of axis i . As the random perturbation is undirected, r_0 is sampled uniformly from $[-1, 1]$.

A fraction of $\hat{q}_r = 10\%$ of particles \mathbf{x}^q are treated as quantum particles [7] to ease the dynamic tracking, cf. Eq. 6. They are distributed around \mathbf{p}^n with standard deviation $\hat{q}_d = 0.15$. The inertia is usually below 1 to allow for convergence, yet it needs to be high to stress the correlation of robot motion, so we set it to 0.99. The trade-off between the ϕ -values remains important, balancing between exploration and exploitation. Random perturbation is necessary for diversity, but reduces accuracy if too dominant. For scenarios with constant velocity, preliminary tests show robustness towards settings of $\phi_0 \in (0, 2]$. We suggest $\phi_0 = 1$ as a default, while ϕ_2 is set to 0.6 where not stated otherwise. A fraction of $\hat{h}_r = 10\%$ of particles are allowed a velocity $\hat{v}_0 = 3v_0$ for quick optimum recovery. The best individual at every iteration is taken as position estimation.

4.1 Self-adaptive Parameters

We introduce two self-adaptive mechanisms. By calculating the speed v_{sw} of the swarm's center, we dynamically hold the relation $v_0 \approx 2v_{sw}$. This enables the method to react to speed changes while providing robust tracking at any speed. Also, v_{sw} gives a good estimate of the robot's speed.

SIFT features offer robust image similarity information in outdoor areas, yet situations may still be ambiguous and the real position may get lost. To handle this, we include a mechanism to adapt the swarm diversity: If the SIFT match of the best position guess is still bad, e.g., matching less than 5% of the features, it may be an ambiguous position or the swarm lost track of the position. If this happens several times in a row, we start a recovery phase and boost particle diversity by increasing v_0 , \hat{q}_r and decreasing ϕ_2 towards limit values. As soon as the particles' quality increases again, the initial values are gradually restored. Experiments show that the adaptive speed improves tracking and the recovery phase improves robustness, cf. Sec. 6.1. With increasing v_0 the method becomes more sensitive to the setting of ϕ_0 . Therefore, we tested several values for ϕ_0 in a setting where recovery phases were important (Sec. 6).

5 Experimental Setting

In the experiments in [2], an RWI ATRV-JR outdoor robot (Fig. 1) collected images in a campus environment. One 320×240 pixel gray scale image per second was taken with a Videre Design SVS camera system at a constant velocity of about 0.6 m/s. The robot is equipped with a differential GPS (DGPS) system, from which ground truth data was read. Under ideal conditions, the accuracy of the DGPS is below 0.5 m. However, due to occlusion by trees and buildings, the GPS path sometimes significantly deviated from the real position or contained gaps. As the robot moved on a smooth trajectory, some wrong GPS values were eliminated as outliers and gaps could be closed by interpolation.

Two different data sets S and C were produced, each consisting of three rounds around a building. A round is 260 m long and contains about 400 images. Three were collected under sunny conditions (S), three more on a cloudy day (C, cf. Fig. 2). The images contain buildings, streets, cars, as well as vegetation. There are also dynamic objects like cars and people passing by. The SIFT features of a round with GPS annotations make up the localization map.



Fig. 2. Example images of the data sets, sunny (left) and cloudy (right)

One experimental run is defined by a training and a test round, the training round constituting the reference map M . At each iteration, an image $img(t)$ of the test round is presented to the algorithm and interpreted as current view of the robot. Where not stated otherwise, the images are presented in the order they were taken in. At each time step, the deviation of the estimated position $\hat{x}(t)$ to the real position of the test image $img(t)$ gives the online error, the average of which makes up the allover error of the run.

For a full experiment on two rounds, we repeat the localization k times with different starting positions, so k is the number of images in the test round. The average error over these runs give the performance in the experiment. To examine some parameter settings, we experiment on two exemplary rounds, while for the final results, we additionally loop over all the rounds in the data sets.

6 Results

Table 1 shows results for different settings of ϕ_0 in a sunny vs. cloudy scenario. They indicate that a setting of $\phi_0 \approx 1$ is favorable, keeping in mind that the

Table 1. Average error (m) varying ϕ_0 with recovery

ϕ_0	0.005	0.02	0.08	0.32	0.64	0.96	1.28	1.60
Avg.err. (m)	2.64	2.67	2.61	2.50	2.43	2.41	2.44	2.59
Error variance	0.26	0.46	0.28	0.25	0.17	0.09	0.07	0.09

Table 2. Varying the number of particles for the PSO-localization

Method	PSO-30	PSO-60	PSO-80	PSO-100	PF-100	PF-300
Avg.err. (m)	3.34	2.51	2.42	2.39	3.95	3.39
Avg.comp./img.	16.87	23.68	27.36	30.60	40.8	62.4

Table 3. Comparing localization with and without recovery for sunny vs. cloudy

Recovery active / ϕ_0	+/1	-/1	+/-0.005	-/-0.005
Avg.err. (m)	2.87	3.55	2.91	3.46
Error variance	0.39	3.26	0.54	2.68

random perturbation is also proportional to the maximum speed v_0 which is increased in recovery phases. A high ϕ_0 -value increases the number of image comparisons, because the swarm diversity tends to be higher.

When comparing several swarm sizes for $\phi_0 = 1$ (Table 2), the localization performance increases with additional particles as expected. At the same time, the number of comparisons increases, and consequently the computation time. For comparison, the results for a PF with 100 and 300 particles are also shown (cf. Sec. 6.2). For robust localization, PSO uses 80 particles in further experiments.

For the sunny vs. cloudy (S vs. C) situation, the advantage of the adaptive recovery is compared to the performance without recovery in Table 3. It shows the averaged errors for 80 particles and $\phi_0 \in \{0.005, 1\}$. For a small ϕ_0 , localization tends to be more exact in simple rounds but is more likely to lose the position. For robust localization, we favor setting $\phi_0 = 1$ and adaptive v_0 with recovery.

To demonstrate the effect of the v_0 -adaptation, we run simulations with different virtual robot velocities. For $\frac{1}{4}/\frac{1}{2}$ of the original speed, we present the same test image 4/2 times in a row, while for 2/3/4 times the original speed, we present only the 2nd/3rd/4th image of the test round, resulting in the virtual speed modified by the respective factor. Fig. 3 (right) shows localization results in the S vs. C case. For the non-adaptive experiment, v_0 is set to roughly twice the original speed, which works for slower speeds but clearly fails for high speeds.

6.1 Kidnapped-Robot Scenario

For localization, a “kidnapping” of the robot, meaning that it is moved by hand without getting informed, is one of the toughest challenges. The robot’s position estimate suddenly becomes completely invalid and misleading. In our environment, we simulate kidnapping by adding $\frac{k}{2}$ to the test image index *modulo k*

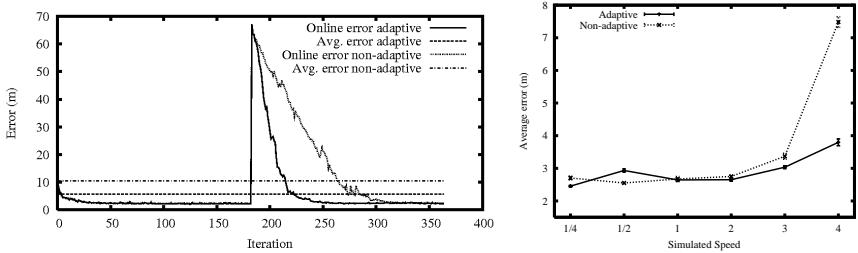


Fig. 3. Kidnapped-robot scenario (left). Varying the virtual speed of the robot (right).

after $\frac{k}{2}$ iterations. Thus, the localization method is forced to jump to the opposite side of the round after converging for half of the run. In Fig. 3 (left), the simulated kidnapping causes an abrupt error of about 68m averaged. Yet, the adaptive method quickly finds and retracks the position.

6.2 Final Comparison to a Particle Filter

For the final comparison of the PSO localization method with a Particle Filter approach, we loop over all the rounds in the two data sets, but without testing a round against itself. This means that for S vs. S and C vs. C, there are six, for S vs. C there are nine experiments averaged (Table 4).

Table 4. Comparing the PF to PSO in avg. error and SIFT comparisons per image

	PF-100		PF-300		PSO-80, $\phi_0 = 0.005$		PSO-80, $\phi_0 = 1$	
	Avg.err.(m)	#Cm	Avg.err.(m)	#Cm	Avg.err.(m)	#Cm	Avg.err.(m)	#Cm
S vs. S	3.16±0.89	40.0	2.15 ± 0.29	60.8	2.03± 0.42	21.0	2.16± 0.63	24.2
C vs. C	3.46±1.28	36.5	2.06 ± 0.56	55.3	1.45 ± 0.53	19.3	1.49± 0.35	22.7
S vs. C	3.93±0.66	40.4	3.27 ± 0.27	60.9	2.91 ± 0.73	22.8	2.87± 0.62	27.1

We compare two PSO variants with different scales of the random perturbation with a PF using 100 and 300 particles [2]. The PF-100 localization is rather inaccurate, producing errors of 3m – 4m, and it requires nearly twice as many image comparisons as the PSO approach. The PF-300 nearly reaches the accuracy of the PSO, but requires about three times as many image comparisons, so the PSO-80 variant is clearly more effective. The difference between low and high random perturbation, depending on ϕ_0 , lies mostly in robustness. Since the standard deviations in Table 4 refer to experiments with different rounds and not single runs, this is more clearly visible in Table 1.

A SIFT comparison of the considered data sets took about 0.015 s on average on our test system, a 2.4 GHz dual core AMD Opteron. An iteration of PF-300 therefore takes approx. 0.8 – 0.9 s. Compared to that, PSO reduces the necessary comparisons by more than 50% and saves nearly half a second in every iteration.

7 Conclusions

Visual outdoor localization of mobile systems requires visual data processing and is therefore time-consuming. A typical localization approach from robotics, the Particle Filter, is successful especially with highly ambiguous sensory and non-Gaussian noise. Yet, sparse visual outdoor images, which occur if large areas (e.g. whole cities) are to be mapped in a short time, are relatively distinctive. We therefore employed a PSO heuristic with modifications appropriate to the high dynamics of a mobile robotic system. Using a current method to extract and compare visual features from images, SIFT, we formulated an optimization problem relating a test image sequence to a given map of images. By adding adaptive mechanisms, the robustness of the swarm method could be increased.

Test results using two data sets recorded under different wheather conditions showed that the PSO localization method requires considerably fewer particles and thus less computation time compared to a Particle Filter approach, but is still more accurate. It is able to adapt to different speeds and solves the difficult kidnapped-robot case. We will tackle larger scenarios and incorporate odometry readings from the robot, e.g., as an additional attractor in the PSO-formula, in future work.

References

1. Jin, Y., Branke, J.: Evolutionary optimization in uncertain environments – a survey. *IEEE Transactions on Evolutionary Computation* 9, 303–317 (2005)
2. Weiss, C., Masselli, A., Tamimi, H., Zell, A.: Fast outdoor robot localization using integral invariants. In: Proc. of the 5th International Conference on Computer Vision Systems (ICVS), Bielefeld, Germany (2007)
3. Kennedy, J., Eberhart, R.: Particle swarm optimization. In: IEEE Int. Conf. on Neural Networks, Perth, Australia (1995)
4. Kronfeld, M., Weiss, C., Zell, A.: Swarm-supported outdoor localization with sparse visual data. In: 3rd Europ. Conf. on Mobile Robots, pp. 259–264 (2007)
5. Vahdat, A.R., NourAshrafoddin, N., Ghidary, S.S.: Mobile robot global localization using differential evolution and particle swarm optimization. In: Srinivasan, D., Wang, L. (eds.) 2007 IEEE Congress on Evolutionary Computation, Singapore, IEEE Computational Intelligence Society, pp. 1527–1534. IEEE Press, Los Alamitos (2007)
6. Moreno, L., Garrido, S., Muñoz, M.L.: Evolutionary filter for robust global localization. *Robotics and Autonomous Systems* 54(7), 590–600 (2006)
7. Li, X., Branke, J., Blackwell, T.: Particle swarm with speciation and adaptation in a dynamic environment. In: GECCO 2006: Proc. of the 8th annual conf. on Genetic and evolutionary computation, pp. 51–58. ACM Press, New York (2006)
8. Eberhart, R.C., Shi, Y.: Tracking and optimizing dynamic systems with particle swarms. In: Proceedings of the 2001 Congress on Evolutionary Computation, vol. 1, pp. 94–100 (2001)
9. Fox, D., Thrun, S., Burgard, W., Dellaert, F.: Particle Filters for Mobile Robot Localization. In: Sequential Monte Carlo Methods in Practice, pp. 401–428. Springer, Heidelberg (2000)
10. Lowe, D.: Distinctive image features from scale-invariant keypoints. *Int. Journal of Computer Vision* 60(2), 91–110 (2004)

A Simulation Study of Routing Performance in Realistic Urban Scenarios for MANETs

Gianni A. Di Caro, Frederick Ducatelle, and Luca M. Gambardella

Istituto Dalle Molle di Studi sull’Intelligenza Artificiale (IDSIA), Lugano, Switzerland
{gianni,frederick,luca}@idsia.ch

Abstract. We study in simulation the performance of two MANET routing algorithms in an urban environment. The two algorithms, AODV and AntHocNet, are representative of two different approaches and design methodologies. AODV is a state-of-the-art algorithm following a purely reactive approach. AntHocNet is based on Ant Colony Optimization and integrates proactive and reactive mechanisms. We investigate the usefulness of the different approaches they adopt when confronted with the peculiarities of urban environments and real-world applications. At this aim we define a detailed and realistic simulation setup in terms of radio propagation, constrained node mobility, and data traffic.

1 Introduction

Recently, the study of *mobile ad hoc networks (MANETs)* has attracted a lot of interest and a number of routing protocols have been designed. However, due to the cost and technological difficulty of setting up scalable MANET testbeds, an important part of research is carried out in simulation, as is common in telecommunications. These simulations are often based on simplified scenarios, where nodes move randomly in an open area, and rely on idealized models of radio propagation and interference. Lately, however, experiences with real world testbeds (e.g., [1]) have shown that results from such simple simulations do not reflect well the performance that can be expected in reality. There is therefore now a lot of interest in the study of simulation scenarios that reflect the more complex situations that can be found in reality in terms of applications, mobility, and radio propagation (e.g., [2]). In particular, it is well understood that the closeness between the performance measured in simulation and that obtainable in practice is strongly affected by the selection of a good model of radio propagation in relationship to the characteristics of the embedding environment [3]. Urban scenarios are particularly challenging in this respect due to the pervasive presence of buildings. Also, they are of interest from an application point of view, since MANETS and mesh-based MANETs in densely populated areas can be a cheap and flexible alternative or complement to other types of networks (e.g., GSM), and are already deployed in some major cities, (e.g., Philadelphia and Taipei).

In this paper, we single out the distinctive properties of urban scenarios in terms of *radio propagation* and *mobility patterns*, and we envisage some basic

ways of using urban MANETs in everyday life. We study how these properties affect the effectiveness of different mechanisms commonly used in routing algorithms (see also [4] for more extensive results). We consider two well-known algorithms, *AntHocNet* [5,6] and *Ad-hoc On-demand Distance Vector routing (AODV)* [7], which have different characteristics and are representative of two different approaches to routing. AODV is a state-of-the-art algorithm that adopts a *purely reactive* strategy: it sets up a route on-demand at the start of a communication session, and uses it till it breaks, after which a new route setup is initiated. AntHocNet is a swarm intelligence algorithm designed after a self-organizing behavior of ant colonies, the shortest paths discovery, and the principles of the related framework of *ant colony optimization (ACO)* [8]. AntHocNet is a *hybrid* algorithm based on the integration of a reactive and a proactive approach to set-up, maintain, and improve paths. We have also investigated the performance of *OLSR* [9] and *ANSI* [10] that are based respectively on a *purely proactive* strategy and on a *mostly reactive swarm intelligence* design; however, in the considered urban scenarios their overall performance is on average worse than AntHocNet and AODV (detailed data will be reported in a different paper). We evaluate both algorithms under different scenarios in an urban environment derived from the street organization of the Swiss town of Lugano. We model urban mobility by limiting the movements of the nodes to streets and open areas in the town, and adjusting their speed to the typical speed of people in a urban environment, be it pedestrians, cyclists or cars. We model the physical propagation of radio waves through the streets of the town using a ray-tracing approach, which accounts for interactions between radio waves and buildings, such as reflection and diffraction [11]. We also did an effort to account for different possible usages of the network, modeling different kinds of applications, including SMS and VoIP traffic. The aim of this work is to point out pro and cons of the considered approaches, both originally developed to mainly address open space situations, when dealing with the challenges of realistic urban scenarios.

The use of town maps and realistic ray propagation has been proposed in a few recent publications. In [12], the performance of AODV is evaluated for different traffic types in a London area, pointing out the need for high node density. The authors of [13] make a detailed simulation of the Munich city center, and evaluate how the performance of AODV in this scenario compares to that in open space simulations. Our work is to our knowledge the first that compares different routing strategies in such a detailed simulation of an urban environment.

2 The Simulation Setup

For the simulations we used the *QualNet* [14] discrete event simulator, to which we have made some adaptations in order to get a realistic simulation of urban conditions. QualNet provides faithful implementations of the different network protocols. At the physical and datalink layer, we used the IEEE 802.11b algorithm, running in distributed coordination function mode, and sending 2 Mbps at 2.4 GHz. At the network layer, we used the routing algorithms described in

Section 2.4. Finally UDP is used at the transport layer, as it is commonly known that TCP has difficulties to work properly in MANETs [15]. All reported data points represent averages over 10 different runs of 500 simulated seconds each.

2.1 The Urban Scenario and Node Mobility

Lugano is a relatively small old town presenting an irregular street topology common to most European cities. We focused on an area of $1561 \times 997 \text{ m}^2$, which covers most of downtown Lugano. Streets define the open spaces where nodes are free to move. Buildings are inaccessible to the nodes and basically play the role of obstacles that put constraints on node movements and shield and reflect signal propagation. Node movements were generated according to an adaptation of the *random waypoint mobility* model (RWP) [16]. Under this model, nodes iteratively choose a random destination and speed, move in a straight line to the chosen destination at the chosen speed, and then pause for a certain time. In our urban version of RWP, destinations are only chosen from among the open spaces in the town, and nodes do not move along a straight line to their destination, but instead follow the shortest path through the streets of the town.

In all our simulations, we have chosen node speeds that correspond to realistic inner city movements. In most simulations, we chose the MANET nodes to be *pedestrians*, with a maximum speed of 3 m/s. Only in the experiments with increased mobility, we allow nodes to go up to 15 m/s, which is a reasonably maximum speed for *cars* in an urban environment. The pause time of our RWP is always 30 sec. Finally, in all experiments, we keep 20% of the nodes *static*, to represent immobile network users in the town or mesh infrastructure devices.

2.2 Radio Propagation

Wireless communication in an urban environment is strongly conditioned by the way radio waves interact with the objects they encounter. The most basic effect is that waves produced at street level are blocked by buildings, so that connectivity in urban wireless networks is restricted compared to open space scenarios. Many urban simulation studies for MANETs only account for this effect, using open space propagation models along the line of sight (LoS) and blocking any non-LoS communication (see e.g. [17]). Others use different heuristic approximations, reducing signal strength for each encountered building (e.g., [2]). In the current study, we use a more detailed approach, which incorporates also other propagation effects. The most important of these effects is reflection off buildings: as radio rays bounce off building walls, they can travel around corners into side streets. Also, reflection allows a signal to travel further along the LoS through a street than it would in open space, since multiple reflected rays are tunneled in the same direction. This means that crude approximation models that do not account for reflection are too restrictive. Another important effect is diffraction, which allows rays to bend around corners to a certain extent. This further improves connectivity to side streets. Other effects include scattering, which is the reflection off small objects and uneven surfaces, and signal variations over time

due to changes in the environment, such as the passing of vehicles or people. Both of these last effects are hard to model correctly and greatly increase the computational complexity (see [18]), and were therefore not taken into account.

The modeling of radio propagation was done in preprocessing using the *WinProp* tool [19], which is a commercial software package to calculate ray propagation in cluttered environments. We started from a two-dimensional map of the center of Lugano, and assumed each building on the map to be of a height sufficient to block radio communication going over it (a height of 5 meters already makes diffraction over the building impossible [18]). Then, we took sample positions every 5 meters along the streets of the town, resulting in 6070 different positions. We placed a transmitter sending with 10 mW at 2.4 GHz in each of these positions, and calculated the resulting received signal strength in each of the other positions using WinProp. Subsequently, we adapted the radio propagation module of QualNet. The precalculated signal strength values are read into memory. During the simulation, the signal strength between a transmitter a and a receiver b is approximated by the precalculated signal strength between a transmitter in the sample point closest to a and a receiver in the point closest to b . This results in a maximal error of 2.5 meters on each side.

2.3 Traffic Patterns

In order to account for different possible application of the network by the users' community, we consider different realistic scenarios for traffic load and distribution. All nodes are seen as equal peers and the two end-nodes of a traffic session are selected at random among all nodes. Data traffic is exchanged in bi-directional way to model interactive communications. Data rate is varied, from 1 packet every 30 seconds, representing an interactive *SMS* (*short messaging service*) communication, up to 25 packets/s, which is sufficient to support good quality *voice-over-IP* (*VoIP*) applications. The packet size is set to 160 bytes which is the payload used by the G.711 PCM voice codec and can also represent a typical size of an SMS. In order to represent silent periods in the interactive communication, only 40% of all scheduled packets are sent (this corresponds to the typical proportion of send time in VoIP traffic).

2.4 The Routing Algorithms

We consider the ACO algorithm AntHocNet and AODV, a reference state-of-the-art algorithm in the field. Both algorithms have been presented in a number of papers, therefore, here we only briefly summarize their characteristics.

AODV[7] follows a reactive approach to routing, which means that nodes only gather routing information for destinations that they are actively communicating with. Nodes that start a data session with a destination that they have no information about, launch a route discovery process that, if successful, sets up a single path to route session data. During the session, the only action taken by the routing algorithm is to periodically send out beacon messages, which allows nodes along the path to control whether each link is still alive. When a link

failure is detected, either intermediate nodes try to locally rebuild the route or the source starts a new route discovery process.

AntHocNet [5,6] combines the typical path sampling behavior of ACO algorithms with a pheromone bootstrapping mechanism derived from Bellman-Ford algorithms to adaptively learn pheromone tables playing the role of routing tables. AntHocNet is a hybrid algorithm, since it combines both reactive and proactive elements. It is reactive since it gathers routing information at the start of a new session via the generation of path discovery agents called *reactive ants*, it uses periodic broadcast of messages to detect link failures, and it reacts to route failures with the generation of ants for local repair or with a new route discovery. In addition to this, while a route is being used AntHocNet also performs proactive actions to improve and extend the available paths. The proactive route improvement is based on a combination of *pheromone diffusion* and path sampling. Periodically, nodes send out messages including pheromone information about the paths that they have available. This allows to incrementally build up and revise paths according to a bootstrapping mechanism. Since these paths might be potentially unreliable due to the slow node-by-node construction, *proactive ants* are repeatedly generated to validate these paths.

3 Experimental Results

3.1 Effect of Data Send Rate and Number of Sessions

In a first set of experiments, we consider an urban scenario with 300 nodes and 10 randomly chosen parallel bi-directional sessions. We change their data send rate from 0.033 packets/s (1 packet every 30 seconds, corresponding to interactive SMS exchanges) up to 25 packets/s (corresponding to good quality VoIP communications). Figure 1 shows the results for delivery ratio and average delay. At the lowest data rate, both algorithms show low delivery and high delay. This is because both of them need to set up a route between source and destination prior to communication. When data packets are sent sporadically, previously constructed routes can hardly ever be reused, and a new route setup is needed almost every time. This is reflected in the overhead, not shown here, calculated as number of control packets forwarded per received data packet. AntHocNet scores bad for this measure at the lowest data rate due to its continuous efforts to improve the created route. As data rates increase, subsequent packets can profit from previous route setups. In AntHocNet, where routes are proactively maintained and therefore remain valid for longer, this effect is visible at lower rates than for AODV, and it is reflected both in the performance and in the overhead, that becomes comparable between the two algorithms starting from 1 packet/s. For the highest rate, both algorithms have a decrease in performance, because the high load of data packets starts to interfere with the control packets. For AntHocNet, that uses more and larger control packets, this effect is stronger.

Since VoIP requires both a delivery ratio $\geq 90\%$ and a delay ≤ 0.15 s, the results for the VoIP rate seem to indicate that none of the two algorithms can support VoIP. However, when we do tests varying the number of sessions from 1

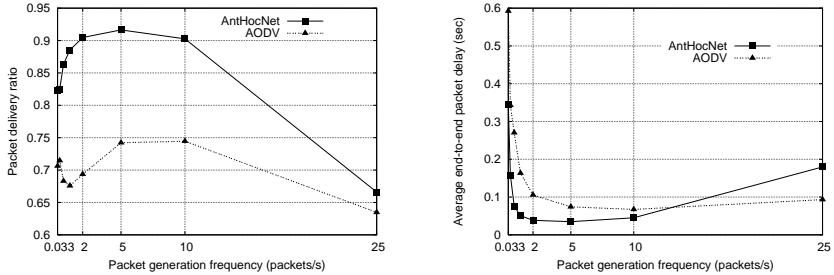


Fig. 1. Delivery and delay with increasing data send rate

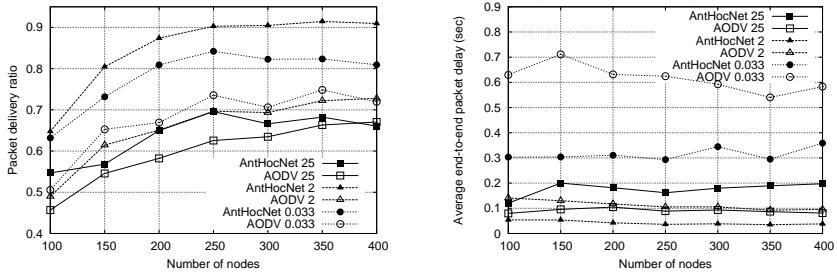


Fig. 2. Delivery and delay with increasing node density and different data send rates

to 10 and we look at individual sessions, we see that a few of them can actually deliver VoIP (results are not shown for lack of space). The average fraction of sessions that can meet VoIP requests degrades linearly from 90% for 1 session to 10% for 10 sessions for AntHocNet, while it stays almost constant for AODV, passing from 40% to 10%. For AntHocNet the total number of sessions with VoIP quality grows up on average to almost 2.5 with 4 sessions, and then remains more or less stable up to 7, when it decreases. For 10 sessions only 1 gets VoIP quality. For AODV, on average, always only 1 session can reach VoIP quality.

3.2 Effect of Node Density and Node Speed

Figure 2 shows results for delivery ratio and delay for changing node density in the case of three types of realistic data load: low (0.033 packets/s), medium (2 packets/s) and high (25 packets/s). The general pattern is similar for each of the data rates: delivery increases as density increases, while the delay stays more or less constant. In terms of delivery, AntHocNet always outperforms AODV, except for the highest data rate in the densest scenario, confirming that the proactive mechanism has its limits when interference gets too high. The same is seen in terms of delay, with AntHocNet outperforming AODV at all densities for the low and medium data rate, but suffering at the highest rate.

We also studied the effect of node speed considering the case of 300 nodes and varying the maximum speed for the same data rates as before. Results in Figure 3

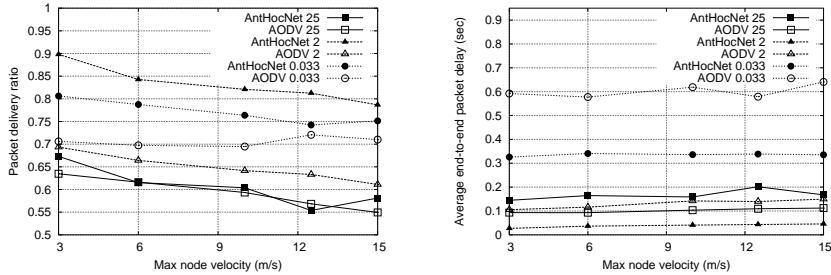


Fig. 3. Delivery and delay with increasing node speed and different data send rates

show that delivery ratio slowly goes down with increased mobility, while delay remains more or less constant. AntHocNet systematically outperforms AODV. The interesting remark is that the node speed has overall relatively little impact on the performance, especially in the limited range of speeds that can be found in a realistic city scenario. The impact of node density and data traffic load seems to be much more important, even if in many MANET simulation studies the speed parameter has often gotten relatively more attention.

4 Conclusions

We have reported the results of an extensive simulation study investigating the performance of two routing algorithms in a realistic simulation of an urban environment. The algorithms, AODV and AntHocNet, differ in their design approach. AODV is a reference state-of-the-art algorithm that uses a purely reactive strategy, while AntHocNet is based on the Ant Colony Optimization framework and combines a reactive approach to route setup with a proactive mechanism to improve and extend existing routing information. The aim of the study was to investigate the advantages of either approach in relationship to the peculiar characteristics of urban environments and to practical application models for real-world MANETS. We created urban node mobility by limiting movements to the streets and open spaces of the town, used ray tracing techniques to model the propagation of radio waves in the urban environment, and applied different types of traffic loads to reflect different kinds of utilization of the network such as exchange of SMS and VoIP communications.

In general, we can see that AntHocNet outperforms AODV both in terms of delivery ratio and delay for most of the scenarios. Thanks to the proactive mechanism, more routing information is available in the network. In other tests (not described here due to space constraints), we have found that this leads to a lower need for route setups and to more success in local route repair attempts. This advantage can lead to less overhead in terms of number of packets despite the use of extra control packets to support the proactive function. However, in situations of high node density, or high data load, the larger beacon messages start to interfere with each other or with data packets. At very low rates, both

algorithms have difficulties due to their specific approach. In urban scenarios, AntHocNet has the advantage that the local density experienced by each node (the number of neighbors) is relatively low, and grows slowly [4]. In previous work, we have noticed that also in open space, AntHocNet outperforms AODV more clearly in sparser scenarios with longer paths and less good connectivity [5]. We also found that node density has a strong impact on the delivery ratio, while the node speed seemed to have relatively lower impact.

References

1. Tschudin, C., Gunningberg, P., Lundgren, H., Nordström, E.: Lessons from experimental MANET research. *Ad Hoc Networks Journal* 3(2), 221–233 (2005)
2. Huang, E., Hu, W., Crowcroft, J., Wassell, I.: Towards commercial mobile ad hoc network applications: A radio dispatch system. In: *Proceedings of the sixth ACM Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)* (2005)
3. Liu, J., Yuan, Y., Nicol, D., Gray, R., Newport, C., Kotz, D., Perrone, L.: Empirical validation of wireless models in simulations of ad hoc routing protocols. *Simulation* 81(4), 307–323 (2005)
4. Ducatelle, F., Di Caro, G., Gambardella, L.M.: A study on the use of MANETs in urban environments. *Technical Report 01-07, IDSIA* (2007)
5. Di Caro, G.A., Ducatelle, F., Gambardella, L.: AntHocNet: an adaptive nature-inspired algorithm for routing in mobile ad hoc networks. *European Transactions on Telecommunications (ETT)* 16(5) (2005)
6. Ducatelle, F., Di Caro, G.A., Gambardella, L.: Using ant agents to combine reactive and proactive strategies for routing in mobile ad hoc networks. *International Journal of Computational Intelligence and Applications (IJCIA)* 5(2) (2005)
7. Perkins, C., Royer, E.: Ad-hoc on-demand distance vector routing. In: *Proceedings of the 2nd IEEE Workshop on Mobile Computing Sys. and Applications* (1999)
8. Dorigo, M., Di Caro, G.A., Gambardella, L.M.: Ant algorithms for distributed discrete optimization. *Artificial Life* 5(2), 137–172 (1999)
9. Clausen, T., Jacquet, P., Laouiti, A., Muhlethaler, P., Qayyum, A., Viennot, L.: Optimized link state routing protocol. In: *Proceedings of IEEE INMIC* (2001)
10. Rajagopalan, S., Shen, C.: ANSI: A swarm intelligence-based unicast routing protocol for hybrid ad hoc networks. *Journal of Systems Architecture* 52(8-9) (2006)
11. Wireless Communications. Cambridge University Press, Cambridge (2005)
12. Sridhara, V., Kim, J., Bohacek, S.: Performance of urban mesh networks. In: *Proceedings of ACM MSWiM* (2005)
13. Schmitz, A., Wenig, M.: The effect of the radio wave propagation model in mobile ad hoc networks. In: *Proceedings of ACM MSWiM* (2006)
14. Scalable Network Technologies, Inc.: QualNet Simulator, Version 3.8 (2005), <http://www.scalable-networks.com>
15. Gerla, M., Tang, K., Bagrodia, R.: TCP performance in wireless multihop networks. In: *2nd IEEE Workshop on Mobile Computing Sys. and Applications* (1999)
16. Johnson, D., Maltz, D.: Dynamic Source Routing in Ad Hoc Wireless Networks. In: *Mobile Computing*. Kluwer Academic Publishers, Dordrecht (1996)
17. Marinoni, S., Kari, H.H.: Ad hoc routing protocol performance in a realistic environment. In: *Proceedings of IEEE ICN* (2006)
18. Sridhara, V., Bohacek, S.: Realistic propagation simulation of urban mesh networks. *Computer Networks* 51(12), 3392–3412 (2007)
19. AWE Communications: WinProp software suite

ACO-Based Scheduling of Parallel Batch Processing Machines with Incompatible Job Families to Minimize Total Weighted Tardiness*

Li Li, Fei Qiao, and Qidi Wu

School of Electronics & Information Engineering, Tongji University, Shanghai, China
 {lili,fqiao}@mail.tongji.edu.cn, wuqidi@moe.edu.cn

Abstract. This research is motivated by the scheduling problem in the diffusion and oxidation areas of semiconductor wafer fabrication facilities (fabs), where the machines are modeled as Parallel Batch Processing Machines (PBPM). The objective is to minimize the Total Weighted Tardiness (TWT) on PBPM with incompatible lot families and dynamic lot arrivals, with consideration on the sequence-dependent setup times. Since the problem is NP-hard, Ant Colony Optimization (ACO) is used to achieve a satisfactory solution in a reasonable computation time. A number of experiments have been implemented to demonstrate the proposed method. It is shown by the simulation results that the proposed method is superior to the common Apparent Tardiness Cost-Batched Apparent Tardiness Cost (ATC-BATC) rule with smaller TWT and makespan, especially TWT that has been improved by 38.49% on average.

1 Motivation

There are many Batch Processing Machines (BPMs) with the ability of processing several lots together in the diffusion and oxidation areas of wafer fabs. A good BPMs scheduling decision is important to efficiently utilize their capacity while satisfy the requirements of their downstream machines to balance the workload in a fab to achieve better fab-wide operational performance.

In recent years, there were considerable researches related to the BPMs scheduling problem. [1] presented a literature review of the related 98 articles published between 1986 and 2004. The related research has been extended a lot since 2004. For example, [2] developed a genetic algorithm (GA) combined with a novel timetabling algorithm for the scheduling of the furnace process. [3] presented a mixed integer program and a simulated annealing (SA) based heuristic method for a single BPM. To be applicable to the real production environment,

* This project was supported by the National Natural Science Foundation of China (No.70531020), the National Basic Research Program of China (No.2002CB312202), the Grant from the Ph.D. Programs Foundation of Ministry of Education of China (No. 20070247007), the Program for Young Excellent Talents in Tongji University (No.2006KJ006), and the Program for New Century Excellent Talents (No.NCET-07-0622).

[4] took into account the future arrival lots during the scheduling process. [5] proved that the single BPM scheduling problem of minimizing total tardiness was NP-hard even if the machine's capacity was two jobs. As a result, the meta-heuristic searching methods (such as GA and SA), with the ability to pursue global optimization, have been gradually adopted to solve this kind of problems.

Ant colony optimization (ACO), inspired by the behavior of real ant colonies, in particular by their foraging behavior, is a population-based approach developed by [6] in 1996. It has been successfully applied to several NP-hard combinatorial optimization problems, such as TSP, QAP, VRP, JSP, FSP, etc. However, few researchers have applied ACO to solve the BPMs scheduling problem. We just find that [7] applied ACO to solve the static scheduling of PBPM with incompatible job families to minimize TWT.

In this paper, we model the diffusion and oxidation operations as PBPM with incompatible lot families and dynamic lot arrivals, and propose an ACO-based solution to minimize TWT. The rest of this paper is organized as follows. In Sect. 2, we describe the problem assumptions and notations. Then we outline the ACO algorithm considered in Sect. 3. In Sect. 4, we continue with computational experiments and results. Finally, we provide conclusions in Sect. 5.

2 Problem Assumptions and Notations

The assumptions and notations of the PBPM scheduling problem include:

1. There are M identical BPMs ($\{m|m = 1, \dots, M\}$) in PBPM, whose capacity is B lots.
2. There are I recipes ($\{i|i = 1, \dots, I\}$) on one BPM. The processing time of recipe i is denoted as P_i . The lots using the same recipe on one BPM can be processed together. However, their number cannot exceed the BPM's capacity, i.e., the maximum batch size constraint. Besides, the processing time of one batch on one BPM is independent of the number of the lots in the batch. Once processing begins on one batch, no lot can be removed from or added to the machine until the processing of the batch finishes.
3. There is sequence-dependent random setup times for changeovers between the lots from different families, and no setup times between the lots from the same family. The setup time between recipe i and h is denoted as U_{ih} .
4. There are n_i lots of family i to be scheduled during the schedule horizon, $\sum_{i=1}^I n_i = N$.
5. Lot j of family i is described as ij . The arrival time, due date, finish time and weight of ij are denoted as A_{ij} , D_{ij} , C_{ij} and w_{ij} , respectively. The tardiness of ij is represented as $T_{ij} = \max\{0, (C_{ij} - D_{ij})\}$. Then, the optimized objective can be written as $\min \left(\sum_i \sum_j w_{ij} T_{ij} \right)$.

Finally, the PBPM scheduling problem in this paper can be represented as

$$M | A_{ij}, \text{Batch}, \text{Incompatible} | \min \left(\sum_i \sum_j w_{ij} T_{ij} \right) \quad (1)$$

3 ACO-Based Solution

There are two ways to solve the PBPM scheduling problem. One is to distribute the scheduled lots to PBPM first, then, batch the lots on each BPM and determine the priorities of the batches on each BPM. The other is to batch the scheduled lots first, then, distribute the batches to PBPM and determine the priorities of the batches on each BPM. [8] pointed out that the second way is superior to the first way with better solution quality and shorter computation time. So we adopt it to solve the PBPM scheduling problem in this paper.

There are two main phases to solve the PBPM scheduling problem by using ACO, i.e., building the search space and implementing the searching process.

3.1 Building the Search Space

The first job to build the search space is to batch the scheduled lots. When batching the lots, there are two main constraints to consider. Firstly, only lots using the same recipe on one BPM can be processed together. On the other hand, the number of the lots in one batch cannot exceed the BPM's capacity. Besides, there is still one important issue to consider, i.e., the trade-off between the time-based utility and the capacity-based utility of PBPM.

There are $C_{n_i}^1 + C_{n_i}^2 + \dots + C_{n_i}^B$ batching styles for n_i lots of family i , subject to the maximum batch size constraint. When there are many lots to be scheduled (especially with a number of dynamic arrival lots), this approach is more vulnerable to the low computation efficiency, which is also helpless to obtain better solution. In this paper, we batch the scheduled lots with the time window concept proposed by [9] to increase the computation efficiency of ACO. At every point of batching decision time t , we consider a time window $\Delta t = dt \sum_i \sum_j P_{ij}/BM$ where dt is the distribution parameter of Δt ; P_{ij} is the processing time of ij equal to P_i . The set of un-batched lots of family i with arrival time less than the upper boundary of the time window interval $t + \Delta t$ is denoted as $M(j, t, \Delta t) = \{ij \mid A_{ij} \leq (t + \Delta t)\}$. Then, we batch the lots in $M(j, t, \Delta t)$ according to their arrival time in ascending order subject to the maximum batch size constraints. Repeat the above process until the batching process finishes. It is noted that the ready time of a batch equals to the latest arrival time of the lots composed of the batch. Eventually, the search space (denoted as S) is built by the combinations of the batches and BPMs in PBPM.

3.2 Implementing the Searching Process

The searching process is to determine the processing machines and processing priorities of the batches. The main issues to consider are the due dates of the scheduled lots and the sequence-dependent setup times. The number of the ants in the artificial ant colony is set as the number of the combinations of the batches and BPMs in the search space. In addition, there are two kinds of termination conditions. One is the maximum iterations (denoted as t_{\max}), while the other is the minimum difference between the continuous minimum objective values

1. Initialization of each artificial ant

Firstly, build a tabu-list L_{tabu}^k and task-list L_{task}^k for each artificial ant k , whose initial values are set as ϕ and S , respectively. Secondly, distribute the combinations in the search space randomly to the artificial ants. The distributed combination to ant k will be added to L_{tabu}^k , and deleted from L_{task}^k . Finally, the combinations with the same batch as the distributed combination will be deleted from L_{task}^k to guarantee one batch processed once.

2. Searching process

Step 1: Initialize the pheromone on each arc as a small positive number ε

$$\tau_{xy}(0) = \begin{cases} \varepsilon & x, y \in S, x \neq y \\ 0 & x, y \in S, x = y \end{cases}, \quad (2)$$

where x, y denote the combinations (i.e., the nodes) in the search space.

Step 2: Each ant selects its next combination from L_{task}^k according to (3). The combination with the largest probability will be added to L_{tabu}^k and deleted from L_{task}^k . Meanwhile, the combinations with the same batch as the selected combination will be deleted from L_{task}^k . Repeat above process until L_{task}^k becomes ϕ . Obviously, L_{tabu}^k is the solution obtained by ant k .

$$p_{c_0 c}^k = \frac{(\tau_{c_0 c}^\alpha \eta_{c_0 c}^\beta)}{\sum_c (\tau_{c_0 c}^\alpha \eta_{c_0 c}^\beta)} \quad (c \in L_{\text{task}}^k) \quad (3)$$

$$\eta_{c_0 c} = \left(1 - \frac{P_c + U_{c_0 c} + \max((A_c - f_{c_0}), 0)}{\max_c(P_c) + \max_c(U_{c_0 c}) + \max_c(\max((A_c - f_{c_0}), 0))} \right) + \frac{B_c}{B} + \Delta W_c$$

$$W_c = \begin{cases} \frac{W_c}{\max_m(W_m)} & \text{if } W_c \leq \max_m(W_m) \\ \frac{W_c}{\max_m(W_m)} - 1 & \text{if } W_c \leq \max_m(W_m) \end{cases},$$

where c is the candidate combination in L_{task}^k ; c_0 is the last selected combination by artificial ant k that uses the same machine with c ; P_c is the processing time of c ; $U_{c_0 c}$ is the setup time for the changeover between c_0 and c ; f_{c_0} is the finish time of c_0 ; A_c is the arrival time of c ; $\eta_{c_0 c}$ is the heuristic factor comprised of three parts: c 's occupation time (including processing time, setup time and waiting time) on the machine processing c_0 , the capacity utility rate of the machine processing c_0 and the relative workload among the machines in PBPM if c is selected as the successor task of c_0 ; B_c is the batch size of c ; W_c is the workload of the machine processing c_0 if c is selected as the successor task of c_0 ; W_m is the workload of machine m ; α, β are the parameters standing for the relative importance of the pheromone density and the heuristic factor.

Step 3: Compute the objective value of the solution obtained by each ant. If the difference between continuous minimum objective values is no more than a small positive value δ , stop the searching process. The tabu-list of

the ant with the minimum objective value is set as the solution. Otherwise, determine whether t_{\max} is reached. If the answer is yes, select the tabu-list with the minimum objective value as the solution; otherwise, go to Step 4. Step 4: Update the pheromone value on the arcs with the minimum objective value according to (4). Repeat Step 2 and Step 3.

$$\Delta\tau_{xy} = \begin{cases} 1 / \min_k (\sum_i \sum_j w_{ij} T_{ij}) & xy \in L_{\text{tabu}}^k | \min_k (\sum_i \sum_j w_{ij} T_{ij}) \\ 0 & \text{otherwise} \end{cases} \quad (4)$$

$$\tau_{xy}(t+1) = (1 - \rho)\tau_{xy}(t) + \Delta\tau_{xy}, \quad 0 < \rho < 1,$$

where ρ is the evaporation rate of the pheromone on the arcs.

4 Computational Experiments and Results

4.1 Simulation Model Description

We use the problem instances from the poly diffusion operations of a real wafer fab to demonstrate our proposed ACO-based solution. There are 3 machines for poly diffusion operations in the wafer fab. The capacity of each machine is 4-lot. Each machine can process 6 different recipes. Firstly, we determine Δt 's distribution parameter dt 's value by simulations (shown as Table 1).

Table 1. Problem instances for determining Δt 's distribution parameter dt

Problem Parameter	Value Used	Total Values
Number of machines	3	1
Number of lots per recipe	8	1
Capacity	4	1
Number of recipes	6	1
Setup times (min)	Uniform(20,30)	1
Recipe1 processing time (min)	Uniform(200,220)	1
Recipe2 processing time (min)	Uniform(200,220)	1
Recipe3 processing time (min)	Uniform(200,220)	1
Recipe4 processing time (min)	Uniform(230,250)	1
Recipe5 processing time (min)	Uniform(200,220)	1
Recipe6 processing time (min)	Uniform(200,220)	1
Arrival time (min)	Uniform(0, $r \cdot \sum_i \sum_j P_{ij}/(BM)$), $r = 0.25, 0.50, 0.75$	3
Due date (min)	$A_{ij} + \text{Uniform}(0, d \cdot \sum_i \sum_j P_{ij}/(BM))$, $d = 0.25, 0.50, 0.75$	3
Time window Δt	$dt \cdot \sum_i \sum_j P_{ij}/(BM)$, $dt = 0.05, 0.25, 0.5, 0.75, 1$	5
Weight per lot	Uniform(0,1)	1
	Total parameter combinations	45
	Number of problems per combination	5
	Total problems	225

The simulation results are shown in Fig. 1. From the simulation results, it can be seen that a better selection for dt 's value is set as 0.25, 0.5 and 0.5 when r is set as 0.25, 0.5 and 0.75, respectively. The main cause is that smaller dt results in the batch size of more batches less than the PBPM's capacity, i.e., less waiting time leads to capacity's loss. So it is worthy to sacrifice some waiting time to gain full utilization of PBPM's capacity in practical production environment unless there are some hot lots or a small quantity of the scheduled lots with extraordinarily loose arrival rate.

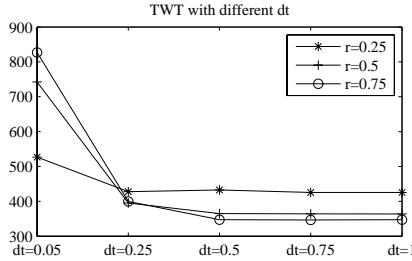


Fig. 1. Simulation results for determining dt

4.2 Comparison between ACO and ATC-BATC

A dispatching rule called ATC-BATC is used to validate the performance of the proposed ACO-based solution. The parameters of ACO-based method are set as follows. The relative importance parameters α and β of the pheromone density and the heuristic are set as 0.9; the evaporation rate of the pheromone ρ is set as 0.1; the initial value of the pheromone on the arcs ε is set as 0.01; the minimum difference between the continuous minimum objective values δ is set as 0.001; the maximum iterations t_{\max} is set as 1000. In addition, we set the maximum computation time as 300 seconds to guarantee the proposed method suitable to the practical production environments.

The problem instances for the comparison between ACO and ATC-BATC are generated in the same manner as Table 1. However, we consider different number of the scheduled lots set as 48, 60, 72, 90 and 120, respectively.

The simulation resules are shown as Fig. 2 and Fig. 3. From the simulation results, we can obtain following conclusions.

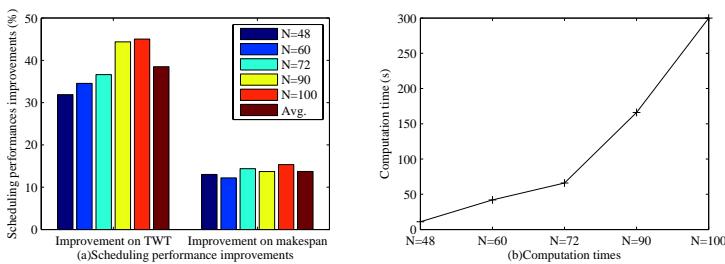


Fig. 2. Comparison between ACO and ATC-BATC with variable number of the scheduled lots

1. The number of the scheduled lots has strong impacts on ACO-based solution's improvement on TWT that is continuously enhanced with the increasing number of the scheduled lots. For example, when the number of the scheduled lots is 48, 60, 72, 90 and 120, the average improvement on TWT

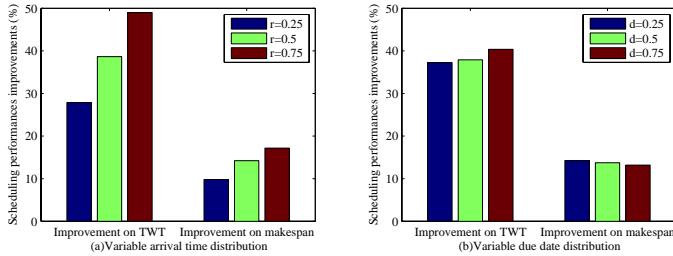


Fig. 3. Comparison between ACO and ATC-BATC with variable arrival time distribution and due date distribution

is 31.89%, 34.56%, 36.63%, 44.36% and 45.03%, respectively. However, it has fewer impacts on ACO's improvement on the makespan that are always between 12.20% and 15.36%. So ACO-based solution is always superior to ATC-BATC with smaller TWT and makespan that are improved by 38.49% and 13.73% on average, respectively (shown in Fig. 2(a)). In addition, the number of the scheduled lots has serious impacts on ACO-based solution's average computation time. For example, when the number of the scheduled lots is 48, 60, 72, 90 and 120, the average computation time is 11s, 42s, 66s, 166s and 300s, respectively (shown in Figure 2(b)).

2. ACO-based solution's improvements on the performance issues are strongly correlated to the arrival time distribution parameter r of the scheduled lots. If the scheduled lots are coming more loosely during the schedule horizon, the improvements on TWT and makespan are much better. For example, when the arrival time distribution parameter r is set as 0.25, 0.5 and 0.75, the average improvement on TWT is 27.86%, 38.64% and 48.99%, respectively, and the average improvement on the makespan is 9.80%, 14.23% and 17.17%, respectively (shown in Fig. 3(a)). However, the due date distribution parameter d has relatively less impacts on ACO solution's improvements on the performance issues. For example, when the due date distribution parameter d is set as 0.25, 0.5 and 0.75, the average improvement on TWT is 37.25%, 37.89% and 40.35%, respectively, and the average improvement on the makespan is 14.26%, 13.75% and 13.19%, respectively (shown in Fig. 3(b)).

5 Conclusions

BPMs play an important role in semiconductor wafer fabrication facilities. In this paper, an ACO-based algorithm is proposed to solve the PBPM scheduling problem with incompatible lot families and dynamic lot arrivals. The main contributions of the paper are to create a production environment applicable method concerned with dynamic lot arrivals, and apply ACO algorithm to obtain the solutions. The simulation results show that the proposed method is superior to the common ATC-BATC rule, especially for TWT performance issue. It has the potential to be used in the real fabs to achieve better operational performances.

References

1. Mathirajan, M., Sivakumar, A.I.: A literature review, classification and simple meta-analysis on scheduling of batch processors in semiconductor. *International Journal of Advanced Manufacturing Technology* 29, 990–1001 (2006)
2. Chien, C.F., Chen, C.H.: A novel timetabling algorithm for a furnace process for semiconductor fabrication with constrained waiting and frequency-based setups. *OR Spectrum* 29, 391–419 (2007)
3. Erramilli, V., Mason, S.J.: Multiple orders per job compatible batch scheduling. *IEEE Transactions on Electronics Packaging Manufacturing* 29, 285–296 (2006)
4. Mönch, L., Zimmermann, J., Otto, P.: Machine learning techniques for scheduling jobs with incompatible families and unequal ready times on parallel batch machines. *Engineering Applications of Artificial Intelligence* 19, 235–245 (2006)
5. Liu, L.L., Ng, C.T., Cheng, T.C.E.: Scheduling jobs with agreeable processing times and due dates on a single batch processing machine. *Theoretical Computer Science* 374, 159–169 (2007)
6. Dorigo, M., Maniezzo, V., Alberto, C.: The ant system: optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics Part B: Cybernetics* 26, 29–41 (1996)
7. Raghavan, N.R.S., Venkataramana, M.: Scheduling parallel batch processors with incompatible job families using ant colony optimization. In: Proceeding. of the 2006 IEEE International Conference on Automation Science and Engineering, Shanghai, China, pp. 507–512 (2006)
8. Balasubramanian, H., Mönch, L., Fowler, J.W., Pfund, M.E.: Genetic algorithm based scheduling of parallel batch machines with incompatible job families to minimize total weighted tardiness. *International Journal of Production Research* 42, 1621–1638 (2004)
9. Mönch, L., Balasubramanian, H., Fowler, J.W., Pfund, M.E.: Heuristic scheduling of jobs on parallel batch machines with incompatible job families and unequal ready times. *Computers and Operations Research* 32, 2731–2750 (2005)

Adaptive Particle Swarm Optimization*

Zhi-hui Zhan and Jun Zhang

Department of Computer Science, Sun Yat-sen University, China
junzhang@ieee.org

Abstract. This paper proposes an adaptive particle swarm optimization (APSO) with adaptive parameters and elitist learning strategy (ELS) based on the evolutionary state estimation (ESE) approach. The ESE approach develops an ‘evolutionary factor’ by using the population distribution information and relative particle fitness information in each generation, and estimates the evolutionary state through a fuzzy classification method. According to the identified state and taking into account various effects of the algorithm-controlling parameters, adaptive control strategies are developed for the inertia weight and acceleration coefficients for faster convergence speed. Further, an adaptive ‘elitist learning strategy’ (ELS) is designed for the best particle to jump out of possible local optima and/or to refine its accuracy, resulting in substantially improved quality of global solutions. The APSO algorithm is tested on 6 unimodal and multimodal functions, and the experimental results demonstrate that the APSO generally outperforms the compared PSOs, in terms of solution accuracy, convergence speed and algorithm reliability.

1 Introduction

Particle swarm optimization (PSO) is one of the swarm intelligence (SI) algorithms that was first introduced by Kennedy and Eberhart in 1995 [1], inspired by swarm behaviors such as birds flocking and fishes schooling. Since its inception in 1995, PSO has been seen rapid development and improvement, with lots of successful applications to real-world problems [2].

Attempts have been made to improve the PSO performance in recent years and a few PSO variants have been proposed. Much work focused on parameters settings of the algorithm [3,4] and on combining various techniques into the PSO [5,6,7]. However, most of these improved PSOs manipulate the control parameters or hybrid operators without considering the varying states of evolution. Hence, these operations lack a systematic treatment of evolutionary state and still sometimes suffer from deficiency in dealing with complex problems.

This paper identifies and utilizes the distribution information of the population to estimate the evolutionary states. Based on the states, the adaptive

* This work was supported by NSF of China Project No.60573066 and the Scientific Research Foundation for the Returned Overseas Chinese Scholars, State Education Ministry, P.R. China.

control parameters strategies are developed for faster convergence speed and the elitist learning strategy (ELS) is carried out in a convergence state to avoid the probability of being trapped into local optima. The PSO is thus systematically extended to adaptive PSO (APSO), so as to bring about outstanding performance when solving global optimization problems.

The rest of this paper is organized as follows. In Section 2, framework of PSO will be described. Then Section 3 presents the evolutionary state estimation (ESE) approach in details and develops the ESE enabled adaptive particle swarm optimization (APSO) through an adaptive control of PSO parameters and an adaptive elitist learning strategy (ELS). Section 4 compares this APSO algorithm against some various existing PSO algorithms using a number of test functions. Conclusions are drawn in Section 5.

2 Particle Swarm Optimization

In PSO, a swarm of particles are introduced to represent the solutions. Each particle i is associated with two vectors, the velocity vector $\mathbf{V}_i = [v_i^1, v_i^2, \dots, v_i^D]$ and the position vector $\mathbf{X}_i = [x_i^1, x_i^2, \dots, x_i^D]$. During an iteration, the fitness of particle i will first be evaluated at its current position. If the fitness is better than that of $pBest_i$, defined as the best solution that the i^{th} particle has achieved so far, then $pBest_i$ will be replaced by the current solution. Following updating all $pBest_i$, the algorithm selects the best $pBest_i$ among the entire swarm as the global best, denoted as $gBest$. Then, the velocity and position of every particle are updated as (1) and (2)

$$v_i^d = \omega \times v_i^d + c_1 \times rand_1^d \times (pBest_i^d - x_i^d) + c_2 \times rand_2^d \times (gBest^d - x_i^d). \quad (1)$$

$$x_i^d = x_i^d + v_i^d. \quad (2)$$

where ω is inertia weight linearly decreasing from 0.9 to 0.4 [3] and c_1, c_2 are acceleration coefficients that are conventionally set to a fixed value 2.0; $rand_1^d$ and $rand_2^d$ are two independently generated random numbers within the range $[0, 1]$ for the d^{th} dimension. Then the algorithm goes to iteration, until a stop condition is met.

Given its simple concept, PSO has been applied in many fields concerning optimization and many researchers have attempted to improve the performance, with variants of PSOs proposed [2].

On the concerns of parameters study, Shi and Eberhart introduced the linearly decreasing inertia weight [3]. Also, Ratnaweera et al. [4] has proposed a linearly time-varying values method for both acceleration coefficients, namely HPSO-TVAC, with a larger c_1 and a smaller c_2 at the beginning and gradually decreasing c_1 whilst increasing c_2 during the running time.

What is more, different techniques such like the selection [5], mutation [4] introduced from GAs have been merged into original PSO to improve the performance. By the inspiration of biology, some researchers introduced niche technology [6] and speciation technology [7] into PSO on the purpose of avoiding the swarm crowding too close and locating as many optimal solutions as possible.

3 Particle Swarm Optimization

3.1 Evolutionary State Estimation

The evolutionary state estimation (ESE) approach in this paper will use not only the fitness information of individuals, but also the population distribution information of the swarm. The evolutionary state in each generation is determined by a fuzzy classification method controlled by an *evolutionary factor* f . These techniques and the estimation process are detailed in the following steps.

Step 1: At current position, calculate the mean distance of particle i to all the other particles by (3)

$$d_i = \frac{1}{N-1} \sum_{j=1, j \neq i}^N \sqrt{\sum_{k=1}^D (x_i^k - x_j^k)^2}. \quad (3)$$

where N and D are the population size and dimension, respectively.

Step 2: Compare all d_i 's and determine the maximal distance d_{\max} and the minimal distance d_{\min} . Denote d_i of the global best particle by d_g . Define an *evolutionary factor* f as (4)

$$f = \frac{d_g - d_{\min}}{d_{\max} - d_{\min}} \in [0, 1]. \quad (4)$$

which is set to 1 if d_{\max} is equal to d_{\min} , and is also initialized to 1 when the algorithm starts.

Step 3: Classify the value of f through fuzzy set membership functions, as shown in Fig. 1(a), and hence determine the current evolutionary state into one of the four different states, say, convergence, exploitation, exploration and jumping-out states. These membership functions are designed empirically and are according to the intuitions that f is relative large in the exploration or jumping-out state and is relative small in the exploitation or convergence state. Since the functions are likely to overlap, the expected oscillation sequence of S_i , such as $S_3 \Rightarrow S_2 \Rightarrow S_1 \Rightarrow S_4 \Rightarrow S_3 \Rightarrow \dots$, may be further used to ascertain the classification.

3.2 Adaptive Strategies for Parameters

The inertia weight ω is used to balance the global and local search abilities, and was suggested to linearly decrease from 0.9 to 0.4 with generation [3]. However, it is not necessarily proper to decrease purely with time. Hence, in this paper, the value of the ω is adaptively adjusted by the mapping $\omega(f) : \mathfrak{R} \rightarrow \mathfrak{R}$ as (5).

$$\omega(f) = \frac{1}{1 + 1.5e^{-2.6f}} \in [0.4, 0.9], \forall f \in [0, 1]. \quad (5)$$

Note that, with the mapping function, ω now changes with f , with large value in exploration state and small value in exploitation state, but not purely with

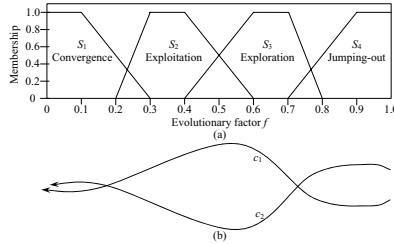


Fig. 1. (a) Fuzzy membership functions for different evolutionary states; (b) Adaptation of the acceleration coefficients according to ESE with the ideal sequence of states $S_3 \Rightarrow S_2 \Rightarrow S_1 \Rightarrow S_4 \Rightarrow S_3 \Rightarrow$

time or with the generation number. Hence, the adaptive inertia weight is expected to be changed efficiently according to the evolutionary states. Since f is initialized to 1, ω is therefore initialized to 0.9 in this paper.

Acceleration coefficients c_1 and c_2 are also important for the exploration and exploitation abilities. In [4], the values of c_1 and c_2 are dynamic changed with time, with larger c_1 and smaller c_2 at the beginning for better exploration and smaller c_1 with larger c_2 at the end for better convergence. Based on the effect of these two parameters, this paper adaptively adjusts them according to the strategies as in Table 1 for different evolutionary states.

Table 1. Strategies for tuning the values of c_1 and c_2

Strategies	States	c_1	c_2
Strategy 1	Exploration	Increase	Decrease
Strategy 2	Exploitation	Slight increase	Slight decrease
Strategy 3	Convergence	Slight increase	Slight increase
Strategy 4	Jumping-out	Decrease	Increase

These strategies share the common attempts with [4] to control the acceleration coefficients dynamic. However, the strategies in this paper are according to the evolutionary states and are expected to be more reasonable and warrantable. The values of c_1 and c_2 are initialized to 2.0 and gradually change as illustrated in Fig. 1(b). With larger c_1 in the exploration state and larger c_2 in the convergence state, the algorithm will balance the global and local search ability adaptively. What is more, a larger c_2 with a smaller c_1 in the jumping-out state can make the swarm in local optimal region separate and fly to the new better region as fast as possible.

The generational change is as (6) where δ is a uniformly generated random value in the range $[0.05, 0.1]$, as indicated by the empirical study. It should be noticed that we use 0.5δ in the strategies 2 and 3 where ‘‘Slight’’ changes are used.

$$c_i(g+1) = c_i(g) \pm \delta, i = 1, 2. \quad (6)$$

What is more, the values of c_1 and c_2 are clamped in range [1.5, 2.5] and their sum is clamped within [3.0, 4.0]. If the sum exceeds the bound, the values of c_1 and c_2 are adjusted by sliding scale.

3.3 Elitist Learning Strategy for $gBest$

The ESE enabled adaptive parameters are expected to bring faster convergence speed to the PSO algorithm. Nevertheless, when the algorithm is in a convergence state, for the $gBest$ particle, it has no other exemplars to follow. So the standard learning mechanism does not help $gBest$ escape from the current optimum if it is local. Hence, an elitist learning strategy (ELS) is developed in this paper to give momentum to the $gBest$ particle. The ELS randomly chooses one dimension of $gBest$'s historical best position, denoted by p^d , and assigns it with momentum to move around. For this, a learning strategy through Gaussian perturbation as (7)

$$p^d = p^d + (X_{\max}^d - X_{\min}^d) \times Gaussian(\mu, \sigma^2). \quad (7)$$

within the saturation limits $[X_{\min}^d, X_{\max}^d]$ can be applied. Here, $Gaussian(\mu, \sigma^2)$ represents Gaussian distribution with a mean $\mu=0$ and a time-varying standard deviation as (8)

$$\sigma = \sigma_{\max} - (\sigma_{\max} - \sigma_{\min}) \times (g/G). \quad (8)$$

where $\sigma_{\max}=1.0$ and $\sigma_{\min}=0.1$ as indicated by the empirical study. It is should be noted that, the new position will be accepted if and only if its fitness is better than the current $gBest$.

4 Experimental Tests and Comparisons

4.1 Testing Functions and Tested PSOs

Six benchmark functions listed in Table 2 are used for the experimental tests. These test functions are widely adopted in benchmarking optimization algorithms [8]. The first 3 are unimodal functions, and the second 3 are complex multimodal functions with a large number of local minima. For details of these functions, refer to [8].

Table 2. Six test functions used in comparison

Test function	n	Search Space	f_{\min}	Acceptance
$f_1 = \sum_{i=1}^n x_i^2$	30	$[-100, 100]^n$	0	0.01
$f_2 = \sum_{i=1}^n x_i + \prod_{i=1}^n x_i $	30	$[-10, 10]^n$	0	0.01
$f_3 = \sum_{i=1}^{n-1} [100(x_{i+1} - x_i)^2 + (x_i - 1)^2]$	30	$[-10, 10]^n$	0	100
$f_4 = \sum_{i=1}^n -x_i \sin(\sqrt{ x_i })$	30	$[-500, 500]^n$	-12569.5	-10000
$f_5 = \sum_{i=1}^n [x_i^2 - 10 \cos(2\pi x_i) + 10]$	30	$[-5.12, 5.12]^n$	0	50
$f_6 = -20 \exp(-0.2 \sqrt{1/n} \sum_{i=1}^n x_i^2) - \exp(1/n \sum_{i=1}^n \cos 2\pi x_i) + 20 + e$	30	$[-32, 32]^n$	0	0.01

The PSO-IW [3] and HPSO-TVAC [4] algorithms are used here for comparison because PSO-IW aims to improve the parameter inertia weight while HPSO-TVAC is improved PSO, by improving the acceleration coefficients. The parameters of these PSOs are set according to the literatures [3,4], and the parameters of APSO are as descriptions above. For a fair comparison among all the three PSOs, they are tested using the same population size of 20, and the same maximal number of 2.0×10^5 function evaluations (FEs) for each test function. Each function is simulated 30 trials independently and their mean values are used in the comparisons.

4.2 Results Comparisons and Discussions

The performance of every PSO is compared in Table 3, in terms of the mean and standard deviation of the solutions obtained by each algorithm.

Table 3. Results of variant PSOs on six test functions

f	PSO-IW	HPSO-TVAC	APSO
f_1	$1.98 \times 10^{-53} \pm 7.08 \times 10^{-53}$	$3.38 \times 10^{-41} \pm 8.50 \times 10^{-41}$	$1.45 \times 10^{-150} \pm 5.73 \times 10^{-150}$
f_2	$2.51 \times 10^{-34} \pm 5.84 \times 10^{-34}$	$6.90 \times 10^{-23} \pm 6.89 \times 10^{-23}$	$5.15 \times 10^{-83} \pm 1.44 \times 10^{-83}$
f_3	28.1 ± 24.6	13.0 ± 16.5	2.84 ± 3.27
f_4	-10090.16 ± 495	-10868.57 ± 289	$-12569.5 \pm 5.22 \times 10^{-11}$
f_5	30.7 ± 8.68	2.39 ± 3.71	$5.80 \times 10^{-15} \pm 1.01 \times 10^{-14}$
f_6	$1.15 \times 10^{-14} \pm 2.27 \times 10^{-15}$	$2.06 \times 10^{-10} \pm 9.45 \times 10^{-10}$	$1.11 \times 10^{-14} \pm 3.55 \times 10^{-15}$

The comparisons in Table 3 show that, when solving unimodal problems, APSO offers the best performance on all the test functions. The fact that the APSO can obtain better solutions on unimodal functions indicates that its adaptive nature indeed offers a faster convergence speed. What is more, APSO outperforms other PSOs on the optimization of all the complex multimodal functions f_4-f_6 as the results presented in Table 3. The advantages are more evident while solving the much more complex problems as Schwefel's function (f_4) and the Rastrigin's function (f_5). This suggests that the APSO has the ability of jumping out local optimal and achieve the global optimum efficiently.

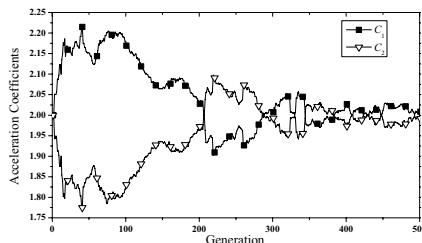


Fig. 2. Adaptive acceleration coefficients during the running time on f_5

Table 4. Mean FEs to reach acceptable solutions and successful ratio

<i>f</i>	PSO-IW	HPSO-TVAC	APSO
f_1	105695(100.0%)	30011(100.0%)	7074(100.0%)
f_2	103077(100.0%)	31371(100.0%)	7900(100.0%)
f_3	101579(100.0%)	33689(100.0%)	5334(100.0%)
f_4	90633(100.0%)	44697(100.0%)	5159(100.0%)
f_5	94379(56.7%)	7829(100.0%)	3531(100.0%)
f_6	110844(96.7%)	52516(100.0%)	40736(100.0%)
Mean Reliability	92.2%	100.0%	100.0%

In order to track the change of acceleration coefficients, Fig. 2 plots the curves of c_1 and c_2 on function f_5 for the first 500 generations. Fig. 2 shows that c_1 is increasing whilst c_2 is decreasing for a number of generations at the beginning because the population is exploring for the optimum. Then c_1 and c_2 reverse their change directions when exploiting for convergence. The jumping out state can also be detected where the value of c_2 increases, c_1 decreases. The search behavior indicates that APSO algorithm has indeed identified the evolutionary states and can adaptively adjust the parameters for better performance.

Table 4 reveals that the APSO offers a generally faster convergence speed, using a small number of function evaluations (FEs) to reach an acceptable solution. For example, tests on f_1 show that the average numbers of FEs of 105695 and 30011 are needed by the PSO-IW and HPSO-TVAC, respectively, to reach an acceptable solution. However, the APSO uses only 7074 FEs to reach the solution. Table 4 also reveals that the APSO offers a generally highest percentage of trials reaching acceptable solutions and the highest reliability averaged over all the test functions.

While the APSO uses identified evolutionary states to adaptively control the algorithm parameters for a faster convergence, it also performs elitist learning in the convergence state to avoid possible local optima. In order to quantify the significance of these two operations, the performance of the APSO without parameters adaptation or elitist learning was tested. Results of mean values on 30 independent trials are presented in Table 5.

Experimental results in Table 5 show that with elitist learning only and without adaptive control of parameters, the APSO can still deliver good solutions to multimodal functions (although with a much lower speed, such a lower

Table 5. Merits of parameter adaptation and elitist learning

<i>f</i>	APSO with Both Adaptation & Learning Parameters	APSO Without Adaptation	APSO Without Elitist Learning	PSO-IW (Standard PSO Without Either)
f_1	1.45×10^{-150}	3.60×10^{-50}	7.67×10^{-160}	1.98×10^{-53}
f_2	5.15×10^{-84}	2.41×10^{-32}	6.58×10^{-88}	2.51×10^{-34}
f_3	2.84	12.75	13.89	28.10
f_4	-12569.5	-12569.5	-7367.77	-10090.16
f_5	5.80×10^{-15}	1.78×10^{-16}	52.73	30.68
f_6	1.11×10^{-14}	1.12×10^{-14}	1.09	1.15×10^{-14}

convergence speed can be reflected by the lower accuracy in solutions to unimodal functions at the end of the search run). On the other hand, the APSO with parameters adaptation only and without an ELS can hardly jump out of local optima and hence results in poor performance on multimodal functions, but it can still solve unimodal problems well. However, both reduced APSO algorithms generally outperform a standard PSO with neither parameters adaptation nor elitist learning, but the full APSO is the most powerful and robust for any given problem. This is most evident in the test result on f_3 . These results confirm the hypothesis that adaptive control of parameters speeds up the convergence while elitist learning helps to jump out of local optima.

5 Conclusions

An adaptive particle swarm optimization (APSO) enabled by evolutionary state estimation has been developed in this paper. Experimental results show that the proposed algorithm yields outstanding performance on not only unimodal, but also multimodal function, with faster convergence speed, higher accuracy solutions, and better algorithm reliability. Future work will focus on testing the APSO on a comprehensive set of benchmarking functions and the applications to real-world optimization problems.

References

1. Kennedy, J., Eberhart, R.C.: Particle Swarm Optimization. In: Proceedings of the IEEE International Conference on Neural Networks, Perth, Australia, pp. 1942–1948 (1995)
2. Li, X.D., Engelbrecht, A.P.: Particle Swarm Optimization: an Introduction and Its Recent Developments. In: Proceedings of the 2007 Genetic Evolutionary Computation Conference, pp. 3391–3414 (2007)
3. Shi, Y., Eberhart, R.C.: A Modified Particle Swarm Optimizer. In: Proceedings of the IEEE World Congress on Computation Intelligence, pp. 69–73 (1998)
4. Ratnaweera, A., Halgamuge, S., Watson, H.: Self-organizing Hierarchical Particle Swarm Optimizer with Time-varying Acceleration Coefficients. *J. IEEE Trans. Evol. Comput.* 8, 240–255 (2004)
5. Angeline, P.J.: Using Selection to Improve Particle Swarm Optimization. In: Proceedings of the IEEE Congress on Evolutionary Computation, Anchorage, AK, pp. 84–89 (1998)
6. Brits, R., Engelbrecht, A.P., van den Bergh, F.: A Niching Particle Swarm Optimizer. In: Proceedings of the 4th Asia-Pacific Conference on Simulated Evolutionary Learning, pp. 692–696 (2002)
7. Parrott, D., Li, X.D.: Locating and Tracking Multiple Dynamic Optima by a Particle Swarm Model Using Speciation. *J. IEEE Trans. Evol. Comput.* 10, 440–458 (2006)
8. Yao, X., Liu, Y., Lin, G.M.: Evolutionary Programming Made Faster. *J. IEEE Trans. Evol. Comput.* 3, 82–102 (1999)

Ant Based Heuristics for the Capacitated Fixed Charge Location Problem

Harry Venables¹ and Alfredo Moscardini²

¹ Sunderland Business School, University of Sunderland, UK

harry.venables@sunderland.ac.uk

² School of Computing & Technology, University of Sunderland, UK

alfredo.moscardini@sunderland.ac.uk

Abstract. This paper presents two different $\mathcal{MAX} - \mathcal{MIN}$ Ant System (\mathcal{MMAS}) based algorithms for the Capacitated Fixed Charge Location Problem (CFCLP) which is a discrete facility location problem that consists of selecting a subset of facilities that must completely supply a set of customers at a minimum cost. The first algorithm is concerned with extending and improving existing work primarily by introducing a previously unconsidered local search scheme based on pheromone intensity. Whilst, the second method makes a transformation of the derived \mathcal{MMAS} algorithm into the hyper-cube framework in an attempt to improve efficiency and robustness. Computational results for a series of standard benchmark problems are presented and indicate that the proposed methods are capable of deriving optimal solutions for the CFCLP.

1 Introduction

The Capacitated Fixed Charge Location Problem (CFCLP) considers the problem of selecting a subset of facilities from a potential set that have to supply a set of customers at a minimum cost, where each customer has an associated demand to be met and each facility has a finite amount of supply available. The CFCLP has been widely studied in the literature and applied in a variety of domains, and is known to be \mathcal{NP} -hard [1,2,3].

Very successful solution techniques are often attributed to those incorporating Lagrangean relaxation combined with various local search and problem reduction strategies [4,5,6]. Various meta-heuristic techniques have also been applied to location problems with some success, such as *Simulated Annealing*, *Genetic Algorithms*, *Tabu Search* and *Very Large Scale Neighborhood Search* [7,8,9,10,11,12] and more recently *Ant Colony Optimization* (ACO) methods have been applied to a small number of facility location problems [13,14,15,16]. A common feature of these methods is an iterative selection of a feasible sub-set of facilities, as the problem then reduces to a transportation problem (TP).

The aims of this paper are to develop, extend and improve work presented in [15]. Two iterative algorithms are presented where each consists of three phases. First a construction phase which randomly selects facilities based on pheromone intensities is implemented. The second is a local search phase that is two-fold,

consisting of pheromone biased *DROP* and *SWAP* procedures. The main difference between the two algorithms is in the third phase concerned with updating the pheromone values. The whole process is repeated until a maximum number of iterations are reached. The first algorithm is an adaption of $\mathcal{MAX}-\mathcal{MIN}$ Ant System (\mathcal{MMAS}) [17,18], whilst the second relies on an implementation of the Hyper-Cube Framework (HCF) [19].

2 Mathematical Formulation of the CFCLP

In the CFCLP, n customers and m potential facility locations are given. Each customer j has demand q_j that must be completely supplied by at least one facility i . The unit transportation cost of supplying a customer j from a facility i is given as c_{ij} . Each facility that is used incurs a one-off fixed opening charge f_i and has a supply capacity of Q_i . The objective is to select a set of facilities that supply all of the customers at a minimum cost.

Let us define:

x_{ij} = the amount of demand customer j is supplied from facility i ,

and the binary decision variable associated with opening a facility i

$$y_i = \begin{cases} 1 & \text{if facility } i \text{ is opened,} \\ 0 & \text{otherwise.} \end{cases}$$

The capacitated fixed charge location problem is defined as

$$\min z = \sum_{i=1}^m \sum_{j=1}^n c_{ij} x_{ij} + \sum_{i=1}^m f_i y_i \quad (1)$$

such that

$$\sum_{i=1}^m x_{ij} = q_j \quad j = 1, \dots, n. \quad (2)$$

$$\sum_{j=1}^n x_{ij} \leq Q_i y_i \quad i = 1, \dots, m. \quad (3)$$

$$y_i \in \{0, 1\} \quad i = 1, \dots, m. \quad (4)$$

$$x_{ij} \geq 0 \quad i = 1, \dots, m, \quad j = 1, \dots, n. \quad (5)$$

Where (1) is the objective function used to minimize the total fixed and transportation costs associated with facility and allocation variables y_i and x_{ij} . Constraint (2) ensures that the demand q_j of each customer j is satisfied, (3) ensures that an open facility i does not supply more than its capacity Q_i , (4) is the binary variable, concerned with a facility i being selected as opened or closed and finally (5) refers to amount of demand supplied from facility i to customer j .

3 CFCLP: Ant Visibility and Local Search

In this section we describe the main modifications and extention to the *MMAS* presented in [15]. First, we consider how to obtain the visibility or heuristic information η_i associated with an ant visiting a facility i . Initially this involves calculating a new transportation unit cost matrix, with elements $C_{ij} = c_{ij} + f_i/q_j$. Those facilities with low C_{ij} values are deemed most likely to be in solutions with low objective function values. We previously used a “total opportunity-cost” method to define the visibility of a facility i as $\eta_i = 1/\sum_{j=1}^n T_{ij}$, where T_{ij} is a penalty matrix derived from the C_{ij} values; due to space limitations the reader is directed to [15] for a detailed description. Alternatively the T_{ij} elements could be set to the C_{ij} values. Unfortunately, cases may arise when unit transportation costs of customer(s) to a selection of facilities are very expensive, which may result in potentially good facilities with plenty of nearby customers being given poor visibility values. In an attempt to overcome this issue penalties are summed up to and including the median penalty. We define our new visibility as

$$\eta_i = \frac{1}{T_i} \quad i = 1, \dots, m;$$

where

$$T_i = \sum_{k=1}^{\lceil \frac{n+1}{2} \rceil} T_{ik} \quad i = 1, \dots, m$$

and T_{ik} are the ordered T_{ij} penalties, as defined by [15], in ascending order up to the median position.

Local Search Phases

During the construction phase some facilities that are fixed open early on may later only play a minor role in accommodating customer demand and thus, improvements may be made locally by closing one or more facilities in the current solution. We previously addressed this by using a best-improvement *DROP* heuristic and reported final solution errors between 0.10% and 12.45%, with some best solution times of just over two minutes. A two-phase local search procedure is presented where both procedures employ a technique that initially relies on the pheromone intensity τ_i at each facility i to help identify those that are most likely to be dropped or swapped in a current solution. Our rationale for this is that some poor facilities may be repeatedly occurring in the best solution to date and consequently acquire too much pheromone. However, this procedure involves obtaining solution to many TPs which can be computationally expensive. Consequently, we only apply these to the best ant solution found at the current iteration.

***DROP* Heuristic.** Firstly, facilities are sorted into ascending order of pheromone intensity. Then, starting with the highest pheromone intensity, facilities are

sequentially closed and tested for any overall cost improvements. If an improvement occurs then that facility is closed and the current solution is updated, otherwise it remains open. The advantage of this is that once the open facilities are sorted, each facility is only considered once during the process whereas a best-improvement method requires repeated searches over the set of open facilities.

SWAP Heuristic. We further attempt to improve the *DROP* solution by using a *SWAP* heuristic in a similar manner to those used in Lagrangean relaxation [4,5]. Initially, the current iterative solution is sorted into sets of opened and closed facilities based on increasing pheromone intensity, $F = \{i|y_i = 1\}$ and $\bar{F} = \{i|y_i = 0\}$. A restricted number of *SWAP*-candidates are then selected by their pheromone levels such that those opened facilities with high intensities are considered for swapping with closed facilities having low levels. The idea is to encourage the interchange of opened facilities with ones that were previously overlooked. The candidate search space is restricted in size by $\max(15, 0.1|S|)$ where $|S|$ is the size of the set of opened or closed facilities being considered. We adopt a first-improvement local search policy that seeks the first swap to give a solution improvement for an opened candidate facility. The technique is then repeated for all remaining open candidates.

4 **MMAS** for the CFCLP

Solutions are constructed for each ant in the colony by making probabilistic moves based upon pheromone intensity and visibility. We consider the CFCLP to be a fully connected graph consisting of nodes that represent facilities and ant movements determine whether facilities should be opened or closed. Initially all facilities are closed and the constraints placed upon this phase are such that a feasible number of facilities are opened and each facility is visited only once, where its status is determined and then fixed. A facility is fixed open if an improvement in the objective function is observed.

Moves are based upon a pseudo-random proportional rule as described by [18]. The scheme selects the next available facility i to be tested for inclusion into the current solution as the facility l with the largest $[\tau_l]^\alpha [\eta_l]^\beta$ in the neighborhood L with probability q_0 , otherwise facility i is chosen with probability $p_i = [\tau_i]^\alpha [\eta_i]^\beta / \sum_{l \in L} [\tau_l]^\alpha [\eta_l]^\beta$. Parameters α and β correspond to the influential roles of pheromone intensity τ_i and visibility information η_i .

After all the ants have finished their tours and the local search phase is completed the pheromones are updated, which includes some evaporation and deposit of pheromone at each facility. Our pheromone update rule is

$$\tau_i \leftarrow (1 - \rho)\tau_i + \Delta\tau_i^{best} \quad i = 1, \dots, m.$$

Where $\Delta\tau_i^{best} = 1/z^{best}$ and z^{best} is the overall cost of the best solution to date. Upper and lower limits τ_{max} and τ_{min} are placed on the pheromones in an attempt to avoid convergence to a local optimum. These are set as $\tau_{max} = 1/\rho z^{best}$ and $\tau_{min} = \tau_{max}/a$ where a is a parameter. Also, τ_{max} is updated

whenever an improvement is made in the best overall cost z^{best} . If the procedure begins to stagnate then the pheromones are reset to the current value of τ_{max} along with ρ and q_0 being discounted by 10% . This is an attempt to encourage a new exploratory search away from the region of stagnation.

5 Hyper-Cube Framework for the MMAS CFCLP

Two main features of the hyper-cube framework (HCF) are that the pheromone levels are restricted to the interval [0, 1] and as the algorithm iterates their intensities tend towards a binary vector [19]. This is certainly a desirable feature for the CFCLP as we use ACO to derive which facilities should be opened to give an optimal solution. During experiments carried out on the OR-Library test data [20] we observed that the algorithm often stagnated at an early stage and frequently thereafter; which is why we introduced discounting of ρ and q_0 in Section 4. The upper pheromone limit is inversely related to the size of the best objective function and the lower limit is scaled factor of this. Consequently, if the objective function is large then their difference is small and thus may cause early stagnation which is an issue raised in [19]. Fortunately, scaling within HCF allows the setting of the lower and upper limits to be fixed at zero and one, so premature stagnation is usually avoided. By applying the HCF to our algorithm we expect a more reliable method of solution than in previous efforts.

To implement this method we need to consider changing the pheromone update phase, setting the initial pheromone levels $\tau_0 = 0.5$ and the upper and lower pheromone limits τ_{max} and τ_{min} . The new pheromone update phase becomes

$$\tau_i \leftarrow (1 - \rho)\tau_i + \rho k \Delta\tau_i^{best} \quad i = 1, \dots, m,$$

and we set

$$\Delta\tau_i^{best} = \frac{1/z^{best}}{\sum_{h=1}^k (1/z^h)}.$$

Where z^{best} is the overall cost of the best solution to-date and k is the number of ants. If the procedure begins to stagnate then the pheromones are reset to the initial value τ_0 along with ρ and q_0 being discounted as previously by 10% .

6 Computational Experience

Experiments for a series of benchmark capacitated location problems, whose optimum solutions are known, were used and are available from the OR-Library (<http://people.brunel.ac.uk/~mastjjb/jeb/info.html>). The algorithms were coded in C++ and experiments were executed on a Pentium 4 3.0GHz Linux PC with 2Gb of RAM. Source codes were compiled using the GNU g++ compiler using the -O option. All TPs were solved exactly using a simplex dual network algorithm from the COIN-OR distribution [21].

Initially, experiments were carried out on thirty seven test problems of various sizes consisting of five trials per problem. Single ant experiments were considered to assess the performance of the *DROP* and *SWAP* procedures for our proposed *MMAS* algorithms. Each experiment was limited to two hundred iterations with the best solution time being recorded. The parameter setting were: $\alpha = 2.5$, $\beta = 0.8$, $\rho = 0.06$, and $q_0 = 0.5$. Initial pheromone levels $\tau_0 = 1/\rho z_0$, where z_0 is an initial feasible solution obtained using a sufficient number of best visibility facilities. They are reset to the current value of τ_{max} should there be no overall improvement after fifty iterations. The average error obtained from [15] was 3.16%, whilst the new method found optimal solutions for all of the test problems across most of the trials. We then considered the effects of efficiency and reliability when using a small colony of five ants. In both sets of experiments sub-optimal trial instances typically gave solutions within 0.1% of their goal. Table 1 gives summary run-time statistics for the experiments conducted. These results indicate that as problem size increases then the two-phase local search plays a more important role in reducing the computational time whilst marginally improving reliability. Interestingly, the simpler $DROP_1$ procedure performs slightly better than $DROP/SWAP_1$ for smaller problem instances. The respective coefficients of variation for the single ant $DROP/SWAP_1$ and small colony of ants for the $DROP/SWAP_5$ procedures across all problem sizes are 1.48 and 1.34, which suggests that the use of a small colony is more reliable even though it may take a little longer to obtain an optimal solution.

Recently a cross-entropy (CE) method was presented [22] and claimed to solve these test problems within two seconds. CE was recognized as being equivalent to the HCF, which uses a smoothing parameter and an elitist strategy during its update phase that are the same as ρ and $\Delta\tau$ in ACO [23]. We adopted the same parameters into our HCF algorithm: $\alpha = 1.0$, $\beta = 0.0$, $\rho = 0.9$ and $\tau_0 = 0.5$. Experiments were then carried out on the same test problems for the same number of trials and iteration limit. HCF₅ results are presented in the final row of Table 1. The coefficient of variation across all problem sizes is 0.95 with an average of 0.62 seconds which is superior to that of our *MMAS* $DROP-SWAP_5$ algorithm.

Our next set of experiments were concerned with determining the ability of using our *MMAS* and HCF techniques to solve the twelve large problems available in the OR-Library; one hundred facilities by one thousand customers. The number of experimental trials was restricted to a maximum of five using a max-

Table 1. Algorithmic run-time summary statistics for the OR-Library test problems of size $m \times n$ results using parameter settings: $\alpha = 2.5$, $\beta = 0.8$ and $\rho = 0.06$. Data displays average CPU time in seconds \bar{t} , CPU time standard deviation in seconds σ_t and coefficient of variation cv .

Algorithm	16 × 50			25 × 50			50 × 50			All Instances		
	\bar{t}	σ_t	cv	\bar{t}	σ_t	cv	\bar{t}	σ_t	cv	\bar{t}	σ_t	cv
$DROP_1$	0.08	0.07	0.89	0.81	0.78	0.96	5.99	2.42	0.40	2.22	2.99	1.35
$DROP/SWAP_1$	0.11	0.04	0.36	0.66	0.40	0.61	3.59	2.49	0.69	1.42	2.10	1.48
$DROP/SWAP_5$	0.15	0.07	0.47	0.74	0.41	0.55	3.60	2.18	0.61	1.46	1.96	1.34
HCF ₅	0.11	0.03	0.27	0.40	0.15	0.38	1.49	0.51	0.34	0.62	0.59	0.95

imum of twenty iterations and ceased when an optimal solution was observed. If an optimum was not found then the best solution and its relative error were recorded. Experiments were conducted using the previous parameters and a small colony of five ants. The HCF found all of the optimal solutions whereas *MMAS DROP-SWAP* algorithm found all but one with an error of 0.2%, performance times were marginally in favor of HCF at just over 3.5 minutes.

7 Conclusions

In this paper we present two ACO based algorithms to solve the CFCLP. The first method is based on *MMAS* and the second on the HCF. When using a small colony of ants, both methods are capable of deriving optimal solutions to a series of OR-Library benchmark problems that outperform recently presented works on CE and *Tabu Search* [10]. We observed via experimentation that the CE algorithm often gave sub-optimal solutions for those problems with fifty facilities, but obtained short run-times for any optimal solutions found. The proposed algorithms are capable of deriving near optimal solutions to a set of larger-size instances available in the library, but suffer from larger run-times than those of CE and more sophisticated Lagrangean relaxation based methods reported in the literature. It may be beneficial to use a swarm of ants in the construction phase to assist with various aspects of exploitation and exploration of solution sample spaces to improve efficiency. Finally, future research should also concentrate on improving the local search phases, perhaps by considering a k -flip technique as opposed to a restricted large neighborhood technique that was employed by the authors.

References

1. Daskin, M.: Network and Discrete Location: Models, Algorithms and Applications. John Wiley and Sons, Inc., New York (1995)
2. Drezner, Z. (ed.): Facility Location. A Survey of Applications and Methods. Springer, New York (1995)
3. Klose, A., Drexl, A.: Facility location models for distribution system design. European Journal of Operational Research (2004)
4. Agar, M., Sahli, S.: Lagrangean heuristics applied to variety of large capacitated plant location problems. J. Opl. Res. Soc. 49(10), 1072–1084 (1998)
5. Beasley, J.: Lagrangean heuristics for location problems. Eur. J. Opl. Res. 65, 383–399 (1993)
6. Bornstein, C., Campêlo, M.: An add/drop procedure for the capacitated plant location problem. Pesquisa Operacional 24(1), 151–162 (2004)
7. Bornstein, C., Azlan, H.: The use of reduction tests and simulated annealing for the capacitated plant location problem. Loc. Sci. 6, 67–81 (1998)
8. Jaramillo, J., Bhadur, J., Batta, R.: On the use of genetic algorithms to solve location problems. Computers and Operations Research 29, 761–779 (2002)
9. Filho, V., Galvão, R.: A tabu search heuristic for the concentrator location problem. Location Science 6, 189–209 (1998)

10. Sørensen, K.: Investigation of practical, robust and flexible decisions for facility location problems using tabu search and simulation. *J. Opl. Res. Soc.* 59(5), 624–636 (2008)
11. Ahuja, R., Orlin, J., Pallottino, S., Scaparra, M., Scutellà, M.: A multi-exchange heuristic for the single-source capacitated facility location problem. *Mgmt. Sci.* 50(6), 749–760 (2004)
12. Bischoff, M., Dächert, K.: Allocation search methods for a generalized class of location-allocation problems. *European Journal of Operational Research* (2007)
13. Olivetti, F., Zuben, F.V., de Castro, L.N.: MAX-MIN ant system and capacitated p-medians: Extensions and improved solutions. *Informatica* 29, 163–171 (2005)
14. Levanova, T., Loresh, M.: Ant colony optimization algorithm for the capacitated plant location problem. In: 12th IFAC Symposium on Information Control Problems in Manufacturing - INCOM 2006, vol. 3, pp. 423–428 (2006)
15. Venables, H., Moscardini, A.: An adaptive search heuristic for the capacitated fixed charge facility location problem. In: Dorigo, M., Gambardella, L., Birattari, M., Martinoli, A., Poli, R., Stützle, T. (eds.) ANTS 2006. LNCS, vol. 4150, pp. 348–355. Springer, Heidelberg (2006)
16. Chen, C., Ting, C.: Combining lagrangian heuristic and ant colony system to solve the single source capacitated facility location problem. *Transportation Research Part E* (2007)
17. Stützle, T., Hoos, H.: The MAX-MIN ant system. *Fut. Gen. Com. Sys.* 16(8), 889–914 (2000)
18. Dorigo, M., Stützle, T.: *Ant Colony Optimization*. MIT Press, Cambridge (2004)
19. Blum, C., Dorigo, M.: The hyper-cube framework for ant colony optimization. *IEEE Transactions on Systems, Man, and Cybernetics - Part B* 34(2), 1161–1172 (2004)
20. Beasley, J.: Or-library: Distributing test problems by electronic mail. In: Operations Research Proceedings, vol. 41, pp. 1069–1079. Springer, Heidelberg (1990)
21. Lougee-Heimer, R.: The common optimization interface for operations research. *IBM Journal of Research and Development* 47(1), 57–66 (2003)
22. Caserta, M., Quiñonez Rico, E.: k A cross entropy-based metaheuristic algorithm for large scale facility location problems. In: MIC - VII Metaheuristic International Conference (June 2007)
23. Dorigo, M., Zlochin, M., Meuleau, N., Birattari, M.: Updating ACO pheromones using stochastic gradient ascent and cross-entropy methods. In: Cagnoni, S., Gottlieb, J., Hart, E., Middendorf, M., Raidl, G.R. (eds.) *EvoIASP 2002, EvoWorkshops 2002, EvoSTIM 2002, EvoCOP 2002, and EvoPlan 2002*. LNCS, vol. 2279, pp. 21–30. Springer, Heidelberg (2002)

Ant Colony Optimization and the Single Round Robin Maximum Value Problem

David C. Uthus¹, Patricia J. Riddle¹, and Hans W. Guesgen²

¹ Department of Computer Science, University of Auckland, Auckland, New Zealand
`{dave,pat}@cs.auckland.ac.nz`

² School of Engineering and Advanced Technology, Massey University,
Palmerston North, New Zealand
`h.w.guesgen@massey.ac.nz`

Abstract. In this paper, we apply the ant colony optimization metaheuristic to the Single Round Robin Maximum Value Problem, a problem from sports scheduling. This problem contains both feasibility constraints and an optimization goal. We approach this problem using a combination of the metaheuristic with backtracking search. We show how using constraint satisfaction techniques can improve the hybrid's performance. We also show that our approach performs comparably to integer programming and better than tabu search when applied to the Single Round Robin Maximum Value Problem.

1 Introduction

It is often difficult to apply the ant colony optimization (ACO) metaheuristic[2] to problems that contain both feasibility constraints and an optimization goal. Previous approaches have tried to either treat the hard constraints as soft constraints[6,9] or combine ACO with constraint satisfaction algorithms like constraint programming[6], stochastic ranking[5], and backtracking search[1]. With the exception of backtracking search, the approaches used here could not guarantee that a feasible solution would be found, even if most solutions in the search space are feasible.

While using backtracking search with ACO guarantees feasible solutions, it can be slow in finding these solutions. This was seen when the approach was used with the Traveling Tournament Problem (TTP)[1]. The time needed for ACO to construct a solution grew at such a rate that it affected overall performance. This increase in time was due to the excessive amount of backtracking required to produce a single feasible solution.

This research looks at improving the performance of using backtracking search with ACO. We will be working with the Single Round Robin Maximum Value Problem (SRRMVP)[12], which has simpler constraints than the TTP and is therefore easier to study. To improve the performance, we use two search strategies used for constraint satisfaction problems, backjumping and the minimum remaining values (MRV) heuristic, along with a problem-specific heuristic inspired from forward checking[8].

Furthermore, we will compare our approach with integer programming (IP) [12] and tabu search (TS)[4] when applied to the SRRMVP. These tests show that combining backtracking search with ACO is a viable approach for both hard-constrained problems in general and sports scheduling problems in particular.

2 Single Round Robin Maximum Value Problem

A Single Round Robin (SRR) Tournament[7,12] is a problem from sports scheduling where one is creating a round robin schedule with each team playing every other team once. A problem instance consists of n teams, with n being even. The teams play across r rounds where $r = n - 1$. Within each round, each team must play one other team, and no team may play more than once within a round.

An extension of this problem is the SRRMVP, which was introduced by Trick[12]. SRR was extended by assigning random values for team i playing team j during round k . The range of these random values are $1\dots n^2$. The objective is to maximize the sum of these values while maintaining a valid solution. By including these random values and trying to maximize them, the problem is given elements of both feasibility and optimality. The motivation behind this was to simulate real-life sports scheduling where one has both feasibility constraints and an optimization goal such as maximizing game attendance.

3 Ant Colony Optimization

ACO is a metaheuristic that has been used on many optimization problems. It is based on the idea of ants being able to find a shortest path from a food source to their nest using a substance called pheromone. Within ACO, artificial pheromone is used to help construct new solutions using information gained from previously constructed solutions, slowly leading to optimal or near-optimal solutions. This separates it from local search algorithms such as tabu search and simulated annealing, which repeatedly improve on a single candidate solution[8].

3.1 Applying to the Single Round Robin Maximum Value Problem

We chose to use $\mathcal{MAX} - \mathcal{MIN}$ Ant System (\mathcal{MMAS})[10,11] as the basis for our approach. This is because it is one of the best performing ACO algorithms for the Quadratic Assignment Problem (QAP) compared to other ACO algorithms[11], and the QAP shares some similarities with the SRRMVP.

When applying ACO to the SRRMVP, we have to work with hard constraints. For this approach, we construct a solution using backtracking search[8], similar to Crauwels' and Van Oudheusden's application of ACO to the TTP[1]. The pseudocode of constructing a solution can be seen in Fig. 1. We use the same number of ants as there are teams. Each ant constructs a solution one round at a time. Within each round, the i^{th} ant begins to pair up the teams by picking the first available team in numerical order starting with the i^{th} team. It then creates a list of available, feasible teams it can play, T . If T is not empty, it chooses $team2$

```

(1) procedure constructSolution(currentRound)
(2)   team1 := first unassigned team for currentRound starting
        with the team corresponding to ant_number
(3)   T := list of unassigned, feasible teams team1 can play in
        currentRound
(4)   repeat
(5)     choose team2 from T using the random_proportional_rule
(6)     if schedule is complete then return true
(7)     else if all teams assigned for currentRound then
            check := constructSolution(currentRound+1)
(8)     else check := constructSolution(currentRound)
(9)     if check = true then return true
(10)    else T := T \ {team2}
(11)   until T is empty
(12)   return false
(13)end.

```

Fig. 1. Code for constructing a solution when combining ACO with backtracking search. The variable currentRound is set to 0 when code first begins.

from T using the random proportional rule. For this problem, the heuristic values used for the random proportional rule is defined as $\eta_{ijk} = d_{ijk}$, with d_{ijk} being the random value assigned for team i playing team j during round k . As there is no home or away games, $d_{ijk} = d_{jik}$. The pheromone values, τ_{ijk} , are then defined as the desirability of teams i and j playing during round k .

Should T become empty while constructing a solution, the ant will then have to backtrack. When constructing a solution for the SRRMVP, there are two cases of backtracking we have to consider. The first case is backtracking within a round, which is the more common of the two cases. The second case we have to consider is backtracking to a previous round or multiple previous rounds. This is less frequent due to the random nature of the algorithm, but can cause excessive backtracking in the worst case.

3.2 Enhanced Backtracking Search

We improve the performance of combining ACO with backtracking search by using various search strategies commonly found when working with constraint satisfaction problems (CSPs). These search strategies help reduce the amount of backtracking required, thus decreasing the amount of time needed to construct a solution. We use two classic search strategies, backjumping and MRV, along with a problem-specific technique we call future round checking (FRC). Backjumping is used when we have to backtrack, MRV is used for picking the first team, and FRC is used to make sure a pairing will not cause future backtracking.

Backjumping. Backjumping[8] allows us to jump back to a conflicting pairing instead of the last assignment when we cannot go any further in constructing a solution. Unlike classic backjumping that keeps track of a conflict list for every

variable, we instead only check for conflicts when backjumping. This helps reduce the overhead associated with keeping a conflict list updated.

With the SRRMVP, after we have come across a team that has no available teams it can be paired with in a given round, we mark that team as the *conflictTeam* and begin to backtrack. We stop backtracking when we come across a pairing where *team1* or *team2* is either the *conflictTeam* or at least one of teams *conflictTeam* still has to play and, in either case, $T \setminus \{team2\} \neq \emptyset$ for that round. This checking would be done at line 9 in Fig. 1.

Minimum Remaining Values Heuristic. MRV[8] is used when working with CSPs to pick the next variable to assign a value to that has the least number of remaining available values left to be assigned. For this problem, it is being used when picking the first team of the pairings, which is line 2 in Fig. 1. Ties are broken by picking a team in numerical order within the subset of tied teams starting with the i^{th} team corresponding to the i^{th} ant. Two important things that we discovered which help MRV perform better when applying ACO to the SRRMVP are using metadata and using MRV only during rounds where there is a lot of backtracking taking place instead of uniformly.

The metadata that we use for MRV is a short list for each team of the teams that it still needs to play. This information can be found out from an ant's tabu list when constructing a solution, but it is slower than keeping a short list for each team and updating this list every time we pair up two teams. We keep the metadata updated during the rounds where we are using MRV.

Using MRV throughout the solution construction process can help reduce the amount of backtracking in total. But with the SRRMVP, most of the backtracking is taking place during the second half of the solution construction process. It is better to use MRV only during the later rounds due to its overhead. As can be seen in Fig. 2, we found that MRV worked best when applied only during the last fourth of the rounds. This shows the importance of using these techniques only in locations where most of the backtracking takes place.

Future Round Checking. We developed a special technique, FRC, for the SRRMVP. The purpose of FRC is to look one round ahead and make sure the selection of the current round does not cause conflict for the future round. We apply FRC during round $r - 2$. When constructing a solution during this round, we look to see if there is an odd number subset of teams left in round $r - 1$ which have to play each other exclusively. If it is an odd number of teams, then there will be no possible way to construct a solution. For example, assume we have three teams, and each team has to play the other two. Since there are two rounds left, it will be impossible for them to play each other during these two last rounds. The same can be said of any odd number of teams. In terms of Fig. 1, this would be line 5, where we would check the pairing of *team1* and *team2* do not cause this conflict. We do this check after every pairing, even though the check might not fail until the whole subset has been assigned for that round.

We created this technique after noticing that most of the backtracking to a previous round was being caused by this problem. This check is fast and efficient

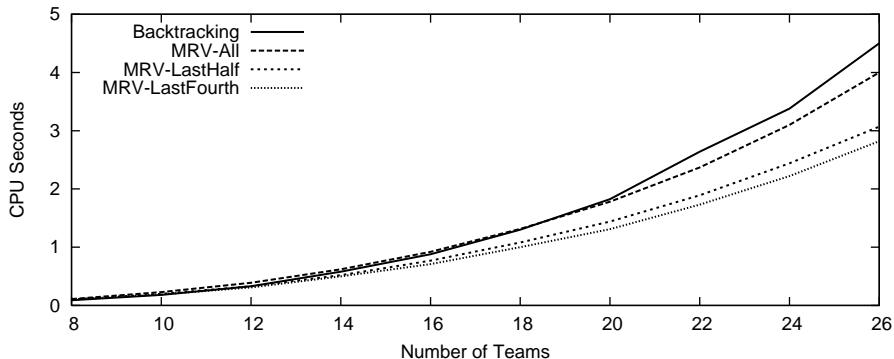


Fig. 2. Comparison of using MRV with metadata for different number of rounds. Time is the medium length of time for an ant to construct 10,000 solutions without pheromone updates over 500 runs.

since we are only checking one round in advance. In addition, we can improve its performance using the same metadata that is used for MRV. If we use FRC and MRV together, FRC's performance is further improved since MRV forces a subset of teams to be assigned together sooner within the round.

Technique Comparisons. In this section, each test for every approach consists of 500 trials of an ant constructing 10,000 solutions and no pheromone updates being used. These tests were done on a 2.13Ghz Intel processor. As can be seen in Fig. 3, each technique by itself helped improve the performance of combining ACO with backtracking search. Backjumping had the largest impact, improving the performance for all problem sizes. This is because it greatly reduced the severity of excessive backtracking and also had the smallest overhead. Using all three combined even further improved the overall performance when $n > 10$.

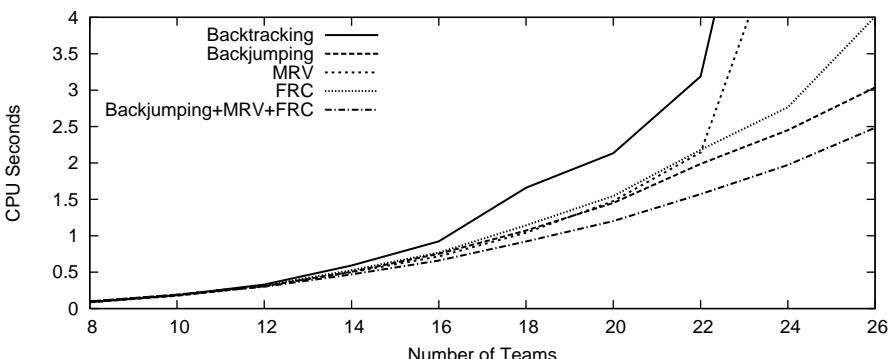


Fig. 3. Comparison of using only backtracking, backjumping, MRV with metadata during the last fourth of the rounds, FRC with metadata, and all three combined. Time is the average length of time for one ant to construct 10,000 solutions without pheromone updates over 500 runs.

We make note of the fact for this problem, backjumping would have been sufficient to solve this problem within a reasonable time. But we worked with the other two techniques to show that using them can further improve performance, which is important when working with more highly-constrained problems.

4 Comparisons and Results

We want to compare our approach to integer programming (IP)[12] because it gives guaranteed optimal results for sports scheduling problems. We also compare our approach with a tabu search (TS) approach that was designed for the TTP[4]. TS has been shown to produce equivalent average solutions with a much faster running time than other approaches to the TTP. The neighborhood we use when applying TS to the SRRMVP is composed of a union of Di Gaspero's and Schaerf's SwapMatches and SwapMatchRound neighborhoods, or $N_4 \cup N_5$. We do not use their more specialized neighborhoods such as CN_3 or CN_4 since they are optimized for the TTP and will not be applicable to the SRRMVP.

When we run our ACO approach by itself, we set $\alpha = 1.0$, $\beta = 2.0$, and $\rho = .02$, which are commonly used values for $\mathcal{M}\text{MAS}$ [2]. We also run our approach in a hybrid form (ACO+TS), pairing up with the TS approach mentioned earlier. Creating a hybrid algorithm of ACO with a local search algorithm is common practice in the field[2]. When running ACO+TS, we use a similar approach as when $\mathcal{M}\text{MAS}$ with TS was used for the QAP[10,11]. We do not use heuristic information and $\rho = .2$. There are two differences between our approach and what was used for the QAP. Instead of using only 5 ants, we use the same number of ants as there are teams, but only the top 5 ants' solutions are improved with TS. Second, we run TS for $4 \cdot \frac{n}{2} \cdot (n - 1)$ cycles instead of $4 \cdot n$ with n being the number of teams for SRRMVP and variables for the QAP. This is because the QAP has a problem size of n while for the SRRMVP it is of size $\frac{n}{2} \cdot (n - 1)$.

We ran ACO, ACO+TS, and TS for the same number of seconds that it took IP to find optimal solutions for team sets from $n = 8$ up to $n = 16$. IP was run on a 1.6Ghz Intel Processor, so we made sure our running times were the same in comparison to processor speed. We ran each approach for 30 trials for each team set. Since we have both optimal maximum and minimum values for each problem set along with their timings, we ran the three approaches twice, once to find the maximum value and once for the minimum.

Table 1 shows the average values found for each algorithm. The times listed in the table are the seconds used by IP. As can be seen when looking for either maximum or minimum values, ACO by itself performs the worse. Yet when we combine it with TS, it performs much better than ACO and has a better average solution quality and smaller standard deviation than TS by itself.

When comparing ACO+TS to IP, ACO+TS did only slightly worse than IP, with the average solutions being within 4% of the optimal solution for all instances. But as can be seen, the time IP required to find the optimal solutions grew exponentially. The advantage of using ACO+TS is that we can run it with less time and still find good solutions. When we run it with an eighth of the

Table 1. Results for the Single Round Robin Maximum Value Problem. The maximum values and minimum values were found separately. The values under IP are the optimal values, while the values under ACO, ACO+TS, and TS are the average values over 30 runs, with standard deviations listed in the brackets.

Maximum Value					
<i>n</i>	Time	IP	ACO	ACO + TS	TS
8	0.06	1393	1380.93(16.75)	1393(0)	1393(0)
10	0.17	3529	3453.03(33.79)	3529(0)	3521.93(12.1)
12	1.55	7635	7213.83(85.43)	7621.3(13.36)	7597.77(21.29)
14	59.26	14824	13799.97(185.29)	14775.4(15.16)	14626.6(55.8)
16	2581	26137	23937.87(206.02)	25871.93(54)	25408.97(89.67)

Minimum Value					
<i>n</i>	Time	IP	ACO	ACO + TS	TS
8	0.06	499	517.43(16.27)	499(0)	499(0)
10	0.18	1061	1208.47(43.33)	1061(0)	1064.9(21)
12	3.83	2092	2555.43(75.55)	2094.83(4.1)	2135.7(19.05)
14	80.58	3055	4220.2(75.17)	3102.93(23.85)	3310.03(50.86)
16	3010	4576	6740.7(110.65)	4774.7(42.73)	5329.17(93.74)

time used by IP, we obtain average maximum values of 1393, 3512.83, 7585.33, 14726.43, and 25774 for the five team sets respectively. We would expect IP to be unable to find solutions for larger team sets or more difficult problems in a reasonable amount of time, as was seen with the TTP[3]. It is with these types of problems where using a metaheuristic like ACO is required[3,7].

5 Conclusion

This research looks at improving the performance of backtracking search when applying ACO to problems that contain both feasibility constraints and an optimization goal. We show that when using classic constraint satisfaction techniques along with a problem-specific heuristic, we can greatly decrease the amount of time required for an ant to construct a solution.

We also show that ACO, when combined with TS as a hybrid algorithm, performs comparably to IP and better than TS by itself on the SRRMVP. This is an indication that ACO has strong potential for sports scheduling problems, an area which ACO has had little application to so far. In the future, we want to look at applying our approach to the TTP. From the performance we have seen with the SRRMVP, we would expect to see a hybrid ACO algorithm with a local search algorithm like TS perform strongly for this more difficult problem.

Acknowledgments. We would like to thank Professor Michael A. Trick of Carnegie Mellon University for his advice on this research and for finding the optimal maximum and minimum values for the SRRMVP.

References

1. Crauwels, H., Van Oudheusden, D.: Ant Colony Optimization and Local Improvement. In: Workshop of Real-Life Applications of Metaheuristics, Antwerp (2003)
2. Dorigo, M., Stützle, T.: Ant Colony Optimization. MIT Press, Cambridge (2004)
3. Easton, K., Nemhauser, G., Trick, M.: Solving the Travelling Tournament Problem: A Combined Integer Programming and Constraint Programming Approach. In: Burke, E.K., De Causmaecker, P. (eds.) PATAT 2002. LNCS, vol. 2740, pp. 100–109. Springer, Heidelberg (2003)
4. Di Gaspero, L., Schaerf, A.: A Composite-Neighborhood Tabu Search Approach to the Traveling Tournament Problem. *J. Heuristics* 13, 189–207 (2007)
5. Meyer, B.: Constraint Handling and Stochastic Ranking in ACO. The 2005 IEEE Congress on Evolutionary Computation 3, 2683–2690 (2005)
6. Meyer, B., Ernst, A.: Integrating ACO and Constraint Propagation. In: Dorigo, M., Birattari, M., Blum, C., Gambardella, L.M., Mondada, F., Stützle, T. (eds.) ANTS 2004. LNCS, vol. 3172, pp. 166–177. Springer, Heidelberg (2004)
7. Rasmussen, R.V., Trick, M.A.: Round Robin Scheduling - A Survey. *European Journal of Operations Research* 188, 617–636 (2008)
8. Russell, S., Norvig, P.: Artificial Intelligence: A Modern Approach, 2nd edn. Prentice Hall, New Jersey (2003)
9. Socha, K., Sampels, M., Manfrin, M.: Ant Algorithms for the University Course Timetabling Problem with Regard to the State-of-the-Art. In: Raidl, G.R., Cagnoni, S., Cardalda, J.J.R., Corne, D.W., Gottlieb, J., Guillot, A., Hart, E., Johnson, C.G., Marchiori, E., Meyer, J.-A., Middendorf, M. (eds.) EvoIASP 2003, EvoWorkshops 2003, EvoSTIM 2003, EvoROB/EvoRobot 2003, EvoCOP 2003, EvoBIO 2003, and EvoMUSART 2003. LNCS, vol. 2611, pp. 334–345. Springer, Heidelberg (2003)
10. Stützle, T.: MAX-MIN Ant System for Quadratic Assignment Problems. Technical report AIDA-97-4, FG Intellektik, FB Informatik, TU Darmstadt, Germany (1997)
11. Stützle, T., Dorigo, M.: ACO Algorithms for the Quadratic Assignment Problem. In: Corne, D., Dorigo, M., Glover, F. (eds.) New Ideas in Optimization, pp. 33–50. McGraw-Hill, London (1999)
12. Trick, M.A.: Integer and Constraint Programming Approaches for Round Robin Tournament Scheduling. In: Burke, E.K., De Causmaecker, P. (eds.) PATAT 2002. LNCS, vol. 2740, pp. 63–77. Springer, Heidelberg (2003)

Artificial Ants to Extract Leaf Outlines and Primary Venation Patterns

Robert J. Mullen, Dorothy Monekosso, Sarah Barman,
Paolo Remagnino¹, and Paul Wilkin²

¹ Digital Image Research Center, Kingston University, London, UK

{r.mullen,n.monekosso,s.barman,p.remagnino}@kingston.ac.uk

² Royal Botanic Gardens KEW, London, UK

p.wilkin@kew.org

Abstract. This paper presents preliminary results on an investigation into using artificial swarms to extract and quantify features in digital images. An ant algorithm has been developed to automatically extract the outlines and primary venation patterns from digital images of living leaf specimens via an edge detection method. A qualitative and quantitative analysis of the results is carried out herein. The artificial swarms are shown to converge onto the edges within the leaf images and statistical accuracy, as measured against ground truth images, is shown to increase in accordance with the swarm convergence. Visual results are promising, however limitations due to background noise need to be addressed for the given application. The findings in this study present potential for increased robustness in using swarm based methods, by exploiting their stigmergic behaviour to reduce the need for parameter fine-tuning with respect to individual image characteristics.

1 Introduction

The use of artificial swarms to solve computational problems has received significant attention in recent years, and the scope of the types of problems and applications to which these techniques are successfully being applied is widening. Swarm intelligence [1] techniques have been shown to perform particularly well at solving certain types of problems such as shortest route problems [2], scheduling problems [3] and assignment problems [4]. Their application is however not limited to these types of problems, and in this paper we address a problem of quite different characteristics, in the field of image processing.

This paper presents an investigation into the application of swarm intelligence to image feature extraction via boundary detection. An ant algorithm is developed and implemented to extract the outline and primary venation pattern from digital images of living leaves.

Identification of leaf types from their venation pattern and outline is common practice for botanist, however traditional plant taxonomy often involves hand drawing such features and making subjective identification by eye [5]. With vastly increasing data sets of digitised herbarium specimen, the scope for data

mining such sources of data has wide reaching importance in applications ranging from vegetation inventory to medicinal plant use to evolutionary links with environmental change. Quantification of leaf features such as venation patterns and outline structures would add a whole new dimension to such data sets, extending the comparative search possibilities of plant characteristics. Due to the very large numbers of specimens involved in these data sets, a robust automated method is required that can efficiently quantify specimen characteristics from a wide range of images. Since specimens are in varying conditions when digitised, robustness of any method used is of key importance.

The motivation behind this work is two-fold; to further research in swarm intelligence, and in doing so, to develop new image processing and classification algorithms with improved robustness. The main aim here is to develop an artificial swarm based automated feature extraction and classification system, with an emphasis on improving robustness, which is key to the success of the chosen application in this paper. This paper presents the first stage in this work.

2 Related Work

Image segmentation is already a well posed problem to which there exists an array of solutions. The measure of success of these solutions is however often based on subjective measurements, and the majority of these methods are based on similar techniques.

More ‘conventional’ methods are often based on filtering operations, such as gradient based methods like the Canny edge detector [6] and Laplacian of Gaussian (LOG) filter [7]. These methods assume edges to be regions in the image that exhibit a large rate of change in gray-level intensity.

The relatively recent approach of using swarm intelligence based methods in image processing has produced some promising results and led to increased interest and further development in this area. Swarm intelligence methods, and more specifically ant algorithms, have been used for such tasks as image segmentation via clustering methods [8][9][10] and via boundary detection methods [11][12][13], as well as image thresholding [14].

In this paper we present an ant algorithm for image segmentation via boundary detection methods with similar characteristics to the algorithm in [13], and perform a quantitative analysis based on a pixel-wise comparison of the results with ground truth images.

3 The Algorithm

The basic framework of this algorithm is based around the workings of the original Ant System (AS) [2], with a modified, application specific pheromone update rule and heuristic information.

The algorithm employs artificial ants as simple computational agents. The algorithm is initialised with N ants occupying ‘random’ pixels within the image, where *pixels* in the *image* are equivalent to *states* in the search *environment*.

At each time step t , each of the N ants moves a distance of 1 pixel to one of the eight surrounding pixels. Each ant's transition from state to state is guided by two main factors: *heuristic information*, and *artificial pheromone trials*.

The heuristic information is defined here as the *visibility*, η_{ij} , which is a measure of the local directional image gradient around the ant's current pixel location. This problem specific heuristic information aims to guide the ant agents to follow along edges and high contrast boundary regions within the image.

The pheromone concentration at any given pixel is given by τ_{ij} . Pheromone deposition by the ants happens at the end of each time step, along with a constant evaporation of the entire pheromone field. These processes are governed by the following pheromone update rule:

$$\tau_{ij}(t+1) = (1 - \rho) \cdot \tau_{ij}(t) + \sum_{k=1}^N \Delta\tau_{ij}^k, \quad (1)$$

where $\rho \in (0, 1]$ is the evaporation rate and $\Delta\tau_{ij}^k$ is the quantity of pheromone deposited at pixel location (i, j) by the k^{th} ant and is given by:

$$\Delta\tau_{ij}^k = \begin{cases} \eta_{ij}/255 & \text{if ant } k \text{ in pixel } (i, j) \text{ and } \eta_{ij} > T, \\ 0 & \text{otherwise,} \end{cases} \quad (2)$$

where T is a user defined threshold value that can be set to only allow pheromone deposition by ants following edges or boundaries above a certain ‘strength.’ In addition there is also a *daemon action* implemented that terminates any ant agent with $\eta_{ij} < T$ for more than Z consecutive time steps. This terminated agent is immediately replaced by a new ant agent at a new ‘random’ location. This step is implemented to reduce the amount of ant agents ‘lost’ searching large background areas of the image and to speed up the rate of convergence of the agents onto the desired regions of the image search space.

Each ant then chooses its next pixel location by applying a probabilistic state transition rule, such that the probability of the k^{th} ant moving from state i to state j , at time step t , is given by

$$p_{ij}^k(t) = \begin{cases} \frac{[\tau_{ij}(t)]^\alpha \cdot [\eta_{ij}]^\beta}{\sum_{k \in \text{allowed}_k} [\tau_{ik}(t)]^\alpha \cdot [\eta_{ik}]^\beta} & \text{if } j \in \text{allowed}_k \\ 0 & \text{otherwise,} \end{cases} \quad (3)$$

where allowed_k is the eight pixels surrounding the k^{th} ant, excluding any pixels in tabu_k , and α and β control the relative importance of the pheromone trail and visibility respectively. tabu_k is a list containing the last n pixel locations of the k^{th} ant, where $n = \text{tabu_max}$ gives the number of time steps into the past for which ants cannot re-visit previously visited pixel locations.

The input to the algorithm is a leaf image, and the output is the emerged pheromone field. This is then thresholded to a binary image to serve as the output feature image.

4 Results

The algorithm was tested on both real and artificial leaf images. The real leaf images are of living samples collected and scanned at the Royal Botanic Gardens KEW (RBG KEW), London, England. Image sizes range from approx. 240px by 600px to approx. 500px by 600px and all images are converted to grayscale before processing.

For each image, unless stated otherwise, the algorithm was run for $t = 500$ time steps, with the following parameters: $N = 5000$, $\alpha = 1.0$, $\beta = 7.0$, $\rho = 0.0001$, $T = 20$, $Z = 5$, and $tabu_max = 500$.

These values were determined, by trial and error, to produce good results over the range of images used in this study.

4.1 Ground Truth Images

Ground truth images for the real leaf images were created by manually tracing the leaf outlines and primary venation patterns via a touch-screen tablet PC device.

Quantitative analysis of the algorithm performance was measured by a pixel wise comparison of the algorithm output against the ground truth images, measuring the True Positive (TP), True Negative (TN), False Positive (FP) and False Negative (FN) pixel classifications, allowing us to compute the statistical measures of *accuracy*, *sensitivity* and *specificity*.

The quality of the manual ground truth method was assessed by carrying out this analysis method on a set of artificial leaf images (to which the true boundary locations are known). The accuracy, sensitivity and specificity results were all in excess of 0.93, and this method was deemed acceptable for this study.

4.2 Qualitative Analysis

As the algorithm runs the ant agents converge onto the boundary regions of the image, resulting in an emergent pheromone field as shown in Figure 1, where the

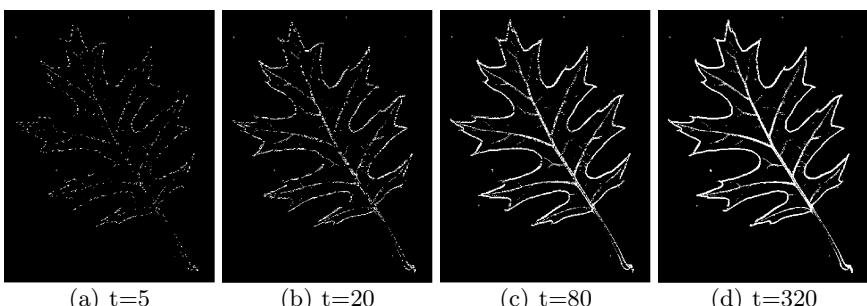


Fig. 1. Emerging pheromone map at different time-steps in the algorithm run. Brighter pixels equal higher pheromone concentration at that point.

brighter pixels correspond to higher pheromone intensity at that pixel location within the image search space.

Figure 2 shows example results of the final pheromone field next to the corresponding ground truth image and original image for both a real and artificial leaf image. As we can see the resultant pheromone field maps out the boundaries within the image, showing clearly the leaf outline and primary venation pattern, where large amounts of pheromone have built up.

Closer visual inspection reveals some inevitable limitations. The effects of noise and non-uniform lighting result in the finer venation detail towards the leaf edges not being fully represented in the algorithm output (see Figure 2(c)). Such missing information could cause problems for leaf type classification, as closely related venation patterns might not be picked up, and unrelated patterns might be wrongly grouped together [15].

The quality of the specimen can vary significantly between samples, with some specimen exhibiting much more defined venation patterns than others, and aside from the effects of noise, lighting issues inherent in the scanning process also affect the specimen image quality.

The images shown in Figures 1 and 2 are typical of the results seen in all the leaf images used in this study.

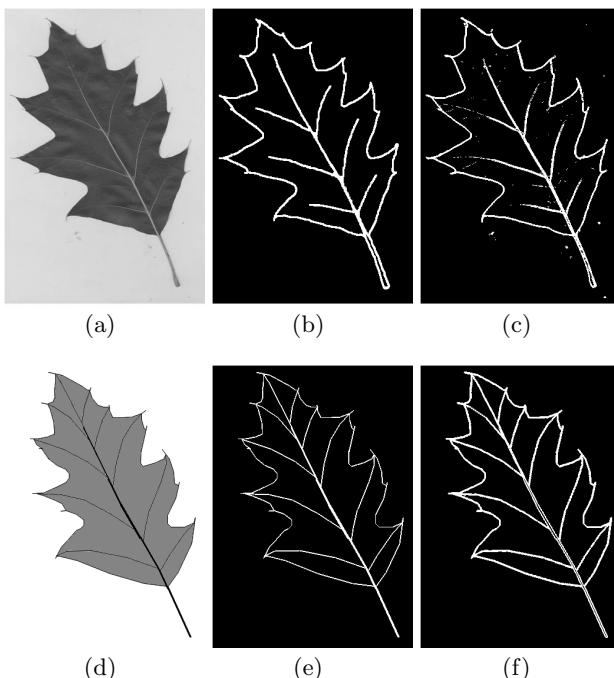


Fig. 2. Example results: original images, (a,d), corresponding ground truth images, (b,e), and final pheromone fields, (c,f). Original image (a) is a real leaf image, and original image (d) is an artificial leaf image.

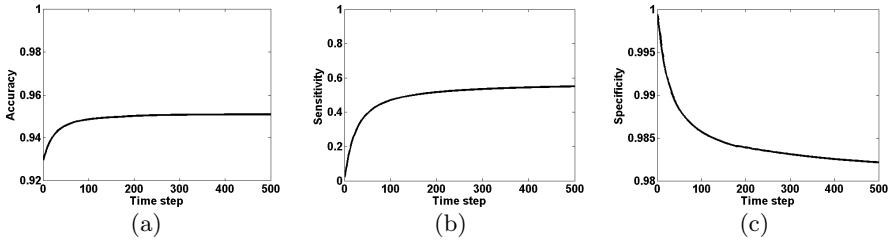


Fig. 3. Plots of average results from 20 real leaf images. Plots show average accuracy, (a), sensitivity, (b), and specificity, (c), over 500 time-steps, as measured from comparing the algorithm output with the ground truth images. Note: Scales are not the same for each image.

4.3 Quantitative Analysis

The quantitative performance analysis of the algorithm is carried out by computing the sensitivity, specificity and accuracy of the algorithm output when compared to the ground truth images, as described in Section 4.1.

The results in Figure 3 are averages from the results of twenty real leaf images. The results show how the accuracy, sensitivity and specificity vary as the algorithm runs. As can be seen, the overall accuracy increases over time, as the ant agents converge on the edges and the pheromone concentration here increases such that more and more pixels are detected as edge pixels. This results in an increase in the number of TP classifications and a decrease in FN, which is also reflected in the increase in sensitivity (Figure 3(b)). The specificity (Figure 3(c)) decreases over time as the number of TN counts decreases and the FP increases, as pheromone concentration builds up across the entire pheromone field, including areas outside of the ‘true’ edge regions. The specificity does not however decrease by any large amount and remains at a high value due to the fact that the TN count is always much greater than the other counts because the majority of the pixels within all of the images are in fact not edge or boundary pixels (i.e. they are background pixels).

5 Discussion

One of the main motivations behind the use of artificial swarms in the area of image processing is to improve on robustness and automation. Many traditional edge detection methods, such as the previously mentioned Canny edge detector [6], operate on the entire image in a linear fashion, and require a number of parameters to be tuned for a given image in order to produce satisfactory results. The ant algorithm approach works in a different way, relying on the phenomena known as *stigmergy* [1] to produce self emergent behaviour amongst the artificial swarm in response to the given environment, which in this case is a digital image. When programmed with a search preference towards high image gradient change,

the swarm converges onto the stronger edges of the image and the resulting pheromone field produced by the swarm maps out these edges. This convergence is reflected in Figure 3(a) with the rapid increase in accuracy over time during the early time-steps as the swarm is converging onto the image edges. After approximately 400 time-steps the gradient of the accuracy curve has almost dropped off to zero, as convergence is achieved.

A potential advantage of the swarm intelligence approach is that it is possible to remove the need to set a sensitivity threshold. In many traditional methods choosing an appropriate threshold value is of most importance in obtaining the best quality results. Future work will therefore aim to remove the threshold value T , which was used in this work as an application specific parameter to facilitate faster convergence and eliminate as much as possible the effects of noise in the resultant pheromone field. A careful choice of all other parameters (in particular the control parameters α and β) to optimise for image edge detection in general (i.e. not just leaf images) could provide a robust enough platform so as to not require any threshold value, and moreover, no parameter tuning at all, for different images.

6 Concluding Remarks

In this paper we have presented a new algorithm for edge detection in image processing using an ant algorithm approach. This has been implemented for the specific task of automated feature extraction of leaf outlines and primary venation patterns in digital leaf images.

The limitation of noise susceptibility addressed in this paper is not specific to our approach, and is in fact a typical problem amongst edge detection algorithms. Ways to combat this problem will be part of the next stage of development of this approach. This will focus on optimising the trade-off between exploration and exploitation, and positive and negative feedback within the system. By performing a thorough analysis of the effects of the parameters used in this method, we aim to increase the robustness of the algorithm and improve on its performance.

The next stage in this work will involve developing ways to quantify the extracted leaf outline and venation pattern data and perform automated leaf classification based on this information.

Robustness being the focal point, additional future work will involve developing hybrid ant algorithms that employ other machine learning techniques to achieve *online* parameter adjustment with an aim to focusing on dynamic problems including the application to feature tracking in near real time imagery.

References

1. Bonabeau, E., Dorigo, M., Theraulaz, G.: *Swarm Intelligence: From Natural to Artificial Systems*. Oxford University Press, Oxford (1999)
2. Dorigo, M., Gambardella, L.M.: Ant Colony System: A cooperating learning approach to the travelling salesman problem. *IEEE Transactions on Evolutionary Computation* 1(1), 53–66 (1997)

3. Colomi, A., Dorigo, M., Maniezzo, V., Trubian, M.: Ant System for job-shop scheduling. *JORBEL - Belgian Journal of Operations Research, Statistics and Computer Science* 34(1), 39–53 (1994)
4. Maniezzo, V., Colomi, A., Dorigo, M.: The Ant System applied to the quadratic assignment problem. Technical Report IRIDIA/94-28, Universite Libre de Bruxelles, Belgium (1994)
5. Hickey, L.J.: Classification of the architecture of dicotyledonous leaves. *American Journal of Botany* 60(1), 17–33 (1973)
6. Canny, J.: A computational approach to edge detection. *IEEE Trans. Pattern Analysis and Machine Intelligence* 8, 679–698 (1986)
7. Gonzalez, R.C., Woods, R.W.: *Digital image processing*, 2nd edn. Prentice Hill (2001)
8. Ouadfel, S., Batouche, M.: Unsupervised image segmentation using a colony of cooperating ants. In: Bültmann, H.H., Lee, S.-W., Poggio, T.A., Wallraven, C. (eds.) *BMCV 2002. LNCS*, vol. 2525, pp. 109–116. Springer, Heidelberg (2002)
9. Ouadfel, S., Batouche, M.: An efficient ant algorithm for swarm-based image clustering. *Journal of Computer Science* 3(3), 162–167 (2007)
10. Channa, A.H., Rajpoot, N.M., Rajpoot, K.M.: Texture segmentation using ant tree clustering. In: *2006 IEEE International Conference on Engineering of Intelligent Systems*, pp. 1–6 (2006)
11. Ramos, V., Almeida, F.: Artificial ant colonies in digital image habitats - a mass behaviour effect study on pattern recognition. In: Bosma, W. (ed.) *ANTS 2000. LNCS*, vol. 1838, pp. 113–116. Springer, Heidelberg (2000)
12. Fernandes, C., Ramos, V., Rosa, A.C.: Self-regulated artificial ant colonies on digital image habitats. *Int. Journal of Lateral Computing* 2(1), 1–8 (2005)
13. Nezamabadi-pour, H., Saryazdi, S., Rashedi, E.: Edge detection using ant algorithms. *Soft Computing* 10, 623–628 (2006)
14. Malisia, A.R., Tizhoosh, H.R.: Image thresholding using ant colony optimization. In: *CRV 2006: Proceedings of the 3rd Canadian Conference on Computer and Robot Vision (CRV 2006)*, p. 26. IEEE Computer Society, Los Alamitos (2006)
15. Wilkin, P.: personal communication, Royal Botanic Gardens, Kew, London, England (February 2008)

Autonomous Reconfiguration in a Self-assembling Multi-robot System

Rehan O'Grady¹, Anders Lyhne Christensen², and Marco Dorigo¹

¹ IRIDIA, CoDE, Université Libre de Bruxelles, Brussels, Belgium

{rogrady,mdorigo}@ulb.ac.be

² DCTI-ISCTE, Lisbon, Portugal

anders.christensen@iscte.pt

Abstract. Self-assembling multi-robot systems can, in theory, overcome the physical limitations of individual robots by connecting to each other to form particular physical structures (morphologies) relevant to specific tasks. Here, we show for the first time how robots in a real-world multi-robot system can autonomously self-assemble into and reconfigure between arbitrary morphologies. We use a distributed control paradigm. The robots are individually autonomous and homogeneous - they all independently execute the same control program. Inter-robot communication is visual and strictly local. We demonstrate our technique on real robots.

1 Introduction

In multi-robot systems, the individual robots can carry out different tasks in parallel. When necessary, they can also cooperate. Self-assembly is a mechanism that allows teams of cooperating robots to overcome the physical limitations of the individual team members by connecting to each other to form physically larger composite robotic entities. In order to maximize the utility of self-assembly, the morphology of the resulting robotic entity must be appropriate to the task. In this paper, we demonstrate how a group of robots can autonomously self-assemble into and reconfigure between different specific morphologies.

This paper's contributions are as follows. 1) We demonstrate autonomous distributed reconfiguration in a real world system of self-assembling robots. 2) We build on our previous work to implement a coordination mechanism that allows a group of connected, independently controlled, self-assembled robots to coordinate the distributed reconfiguration process. 3) We show reconfiguration with six real robots from a star morphology to a line morphology to a square morphology and back to a star. 4) We demonstrate more complex reconfiguration in simulation.

2 Related Work

There is a large body of existing research on self-reconfiguring robotic systems. However, successful demonstrations of reconfiguration on real world robotic

platforms have been restricted to systems in which the individual modules were simple and incapable of independently executing meaningful tasks. Other studies have explored the reconfiguration possibilities of systems made up of autonomous self-assembling robots, but only in simulation.

In self-reconfiguring modular systems, morphological flexibility is explored through the use of relatively simple connected units. Examples include CEBOT [1], PolyBot [2], M-TRAN [3], ATRON [4] and SuperBot [5]. For detailed overviews see [6,7]. In a few self-reconfigurable systems such as the Super-Mechano Colony [8] and the Swarm-bot platform (which we use in this study), the individual modules are capable of carrying out meaningful tasks on their own.

Some proposed control algorithms for self-assembly and/or reconfiguration are centralized, see for instance [9]. Other approaches give each unit a unique ID and a predefined position in the final structure, see for instance [10]. More distributed approaches include [11], [12], and [13]. In simulation, Støy and R. Nagpal [14,15] have demonstrated algorithms for self-reconfiguration and directed growth of cubic units based on gradients and cellular automata. Bojinov *et al.* [16] have shown how a simulated modular robot (Proteo) can self-reconfigure into useful and emergent morphologies when the individual modules use local sensing and local control rules.

3 Hardware Platform and Control Methodology

For our experiments we use the Swarm-bot robotic platform [17], see Fig. 1 (left). This platform was designed and built by Francesco Mondada’s group at the Laboratoire de Système Robotiques (LSRO) of EPFL. The Swarm-bot platform consists of a number of robots called *s-bots*. Each s-bot is self-propelled using a differential drive system composed of combined tracks and wheels (*treels*). Physical connections between s-bots are established by a gripper-based connection mechanism as shown in Fig. 1. An s-bot is surrounded by a transparent ring that can be grasped by other s-bots. Eight sets of RGB-colored LEDs are distributed around the inside of the transparent ring. Individual LEDs can be independently illuminated. The s-bot camera can perceive the illuminated LEDs of other s-bots at a range of up to approximately 50 cm depending on light conditions. The camera records the panoramic images reflected in a spherical mirror mounted above the s-bot in a perspex turret.

Our control paradigm is distributed. The robots are homogeneous and individually autonomous — each robot independently executes the same control program. Because of the limited sensing range of the robots, no individual robot can perceive the global shape of a morphology composed of physically connected s-bots. This means that an unattached robot cannot deduce where it should connect in order to extend or reconfigure a morphology appropriately. Instead, robots that are already part of a morphology indicate how new robots should connect. This is done using the *directional self-assembly* mechanism, which allows a robot to light up a specific set of LEDs in order to invite a connection at a given point on its body with a corresponding specified orientation for the connecting robot (see Fig. 1 right). This mechanism is described in more detail in [18].

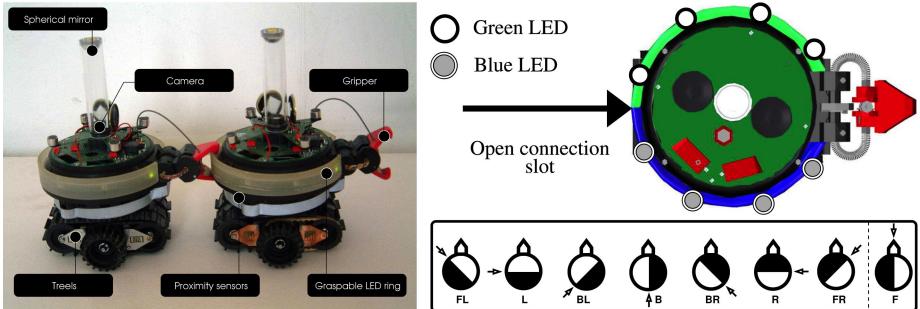


Fig. 1. Left: Two physically connected s-bots. Right top: An s-bot with an open connection slot to its rear (it has illuminated its left green LEDs and its right blue LEDs). Right bottom: Representation of the eight possible connection slots that an s-bot can open. The eighth connection slot (F) is only used for signalling, as it is occupied by the gripper of the s-bot displaying the connection slot.

When a new connection is formed, the two connected robots communicate. Using a low bandwidth visual communication protocol, instructions are typically passed from the robot that is already part of the morphology to the newly connected robot. Similar communication between connected robots occurs during the reconfiguration process. Simple algorithmic rule sets describe when the robots choose to communicate and how to interpret and act upon received communication. Different rule sets allow self-assembly into and reconfiguration between arbitrary morphologies. These rule sets are expressed in a high level descriptive language — SWARMORPH-script. This language is described in more detail in [19].

4 Reconfiguration

In our previous work [19], we showed how homogeneous independently operating robots executing SWARMORPH-script instructions can generate arbitrary morphologies. In this study, we use the same principles to generate autonomous reconfiguration in a distributed self-assembling system.

The key new challenge we have solved in this study is that of coordination between independently controlled robots once they are self-assembled. Coordination is an essential component for any distributed reconfiguration system. Imagine a group of connected robots crossing a hole. If the first robots to cross the hole detect a new obstacle which triggers self-reconfiguration and these robots start trying to reconfigure while some robots are still suspended over the hole, the consequences could be disastrous. In this example, we would want each individual robot to wait before reconfiguring until the whole morphology is ready to reconfigure. What renders this type of coordination difficult is the distributed nature of the system—each robot is controlled independently, and at the same time communication is local and can only occur between directly connected robots.

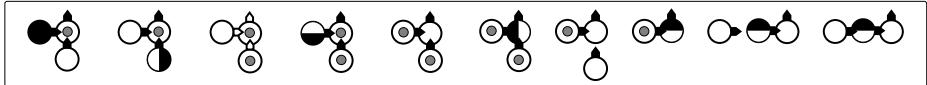


Fig. 2. Coordinated reconfiguration example. A small grey circle in the center of an s-bot indicates that the s-bot is waiting for a signal. For details see text.

On our hardware platform, each robot has a single gripper. This means that any morphology formed, whatever its spatial configuration (shape), has a tree-like connection topology, where a parent node can have many child nodes, but a child node can only have a single parent node. To enable coordination in a connected morphology, we allow for two types of communication. Information can be passed up the tree, and instructions can be passed down the tree.

An example of reconfiguration using these two types of communication is shown in Fig. 2. In this simple example, a three s-bot arrow morphology (one root node, two child nodes) reconfigures into a line morphology. The SWARMORPH-script to generate this behavior is shown in Algorithm 1 (the script includes instructions for the self-assembly of the initial arrow morphology which is not shown in the figure). The control is homogeneous for the three robots—the root node is the first robot to encounter a hole, the other two robots see an open connection slot, attach, and become the child nodes.

In Fig. 2, a small grey circle in the center of an s-bot indicates that the s-bot is waiting for a signal. In Fig. 2 step 1, the arrow morphology has already formed and the seed is waiting for both children to signal that they are ready to reconfigure. In steps 2-4, the child nodes independently wait for a given timeout before they are ‘ready’. Once the child nodes are ready to reconfigure, they both independently signal their readiness to the root node.¹ Once they have signaled their readiness to the root node (information passing up the tree), the child nodes wait for a signal from the root node (instructions passing down the tree) before proceeding with any further reconfiguration steps. In step 5, the root node has received signals from both of its children, and thus knows that the whole morphology is ready to reconfigure. In steps 6-8, the root node signals to both of the child nodes in turn that they can proceed with the reconfiguration. In steps 9-10, having received the relevant instruction, each child node carries out its subsequent reconfiguration instructions. The result is the line morphology.

In the more general case, morphologies can be of arbitrary depth and have arbitrary numbers of branches. Information is passed up the tree from child nodes to parent nodes. Information continues to ascend the tree in this manner until one node takes responsibility for collating the information and issuing instructions to nodes beneath it in the tree. Starting from this collating node, instructions to start the reconfiguration process are then passed down the tree from parent nodes to child nodes.

¹ In this example, the child node signals do not overlap temporally. The algorithm would be unaffected, however, if both child nodes signalled their readiness at the same time—the root node would acknowledge receipt of each of the two signals in turn.

Algorithm 1. Reconfiguration script to generate the reconfiguration sequence in Fig. 2. The script is independently executed on each of the robots.

```

RandomWalk();           // until hole detected OR connection slot detected
if hole-detected then
    // I am the root node since I detected the hole before I saw a connection slot
    OpenConnSlot(back);      // Attract a child robot and...
    SendRuleID(1);          // instruct it to follow rule 1
    OpenConnSlot(left);     // Attract another child robot and...
    SendRuleID(2);          // instruct it to follow rule 2
    WaitForASignal();       // One child is ready
    WaitForASignal();       // Other child is ready
    SendSignal(back);       // Instruct one child to start reconfiguration
    SendSignal(left);       // Instruct other child to start reconfiguration
end
else if connection-slot-detected then
    // I am a child node since I saw a connection slot before I detected the hole
    SearchForAndAttachToConnectionSlot();
    ReceiveRuleID();
    if receivedruleid = 1 then
        Timeout();           // Ready after timeout
        SendSignal(front);   // Inform parent I am ready
        WaitForASignal();    // Wait to receive reconfigure instruction
        Disconnect();
        SearchForConnSlot();
    end
    if receivedruleid = 2 then
        Timeout();           // Ready after timeout
        SendSignal(front);   // Inform parent I am ready
        WaitForASignal();    // Wait to receive reconfigure instruction
        OpenConnSlot(back);
    end
end
StopExecution();

```

In the experiments we perform in this study, the node that takes responsibility for collating information is always the root node (that is, the node that has no parents). However, in other cases, a purely local reconfiguration might be more efficient—this can occur if a node further down the tree takes responsibility for collating information and issuing reconfiguration instructions. Imagine, for example, a self-assembled entity in which a single constituent robot develops a fault. The local structure could reconfigure to eject the faulty robot, or otherwise compensate for the fault. The rest of the assembled robots need not be involved in or even be aware of the local reconfiguration. Note that local configuration still requires coordination—it is just that the coordination is restricted to the subset of assembled robots that are reconfiguring.

5 Results

We performed an experiment with real robots in which six s-bots form the sequence of morphologies: star-line-square-star. We implemented a simple coordination strategy to ensure that each individual morphology is complete before reconfiguration into the next morphology begins: each sub-branch of the

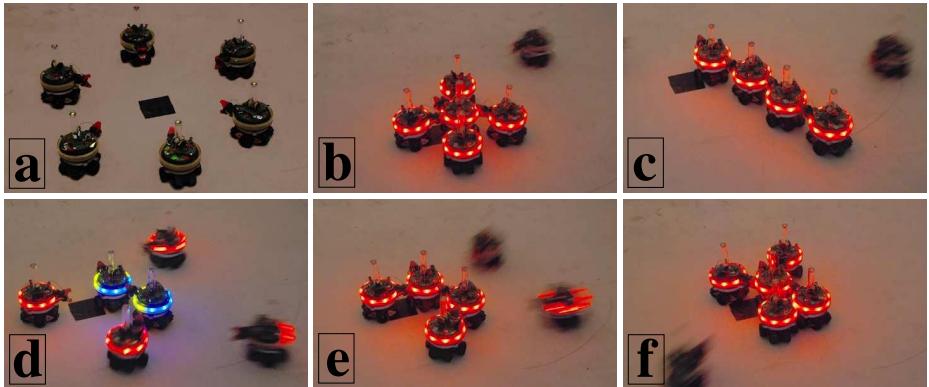


Fig. 3. Photos of different stages in a self-reconfiguration experiment with six real s-bots. a) The start configuration. b) The star morphology. c) The line morphology. d) The line morphology reconfiguring into the square. e) The square morphology. f) The star morphology.

morphology reports local completion up the tree (information passing up the tree). The root node collates the information, and once it is sure that all sub-branches have completed, it sends the instruction to reconfigure down through each branch (instructions passing down the tree). As individual nodes receive the reconfiguration instruction, they execute their subsequent reconfiguration control logic that results in the formation of the next morphology.

Photographic snapshots of this experiment with the real robots are shown in Fig. 3. Videos of the experiments described in this section and other explanatory material, including full SWARMORPH-script reconfiguration algorithms, can be found on the web at <http://iridia.ulb.ac.be/supp/IridiaSupp2008-004>.

We conducted several more complex experiments in simulation. An example is shown in Fig. 4. Here, by executing the relevant SWARMORPH-script program, the robots first assemble into a 9-robot square formation and then reconfigure into three 3-robot arrow morphologies. During the reconfiguration, two connected pairs of s-bots (four s-bots in total) remain connected. Using a

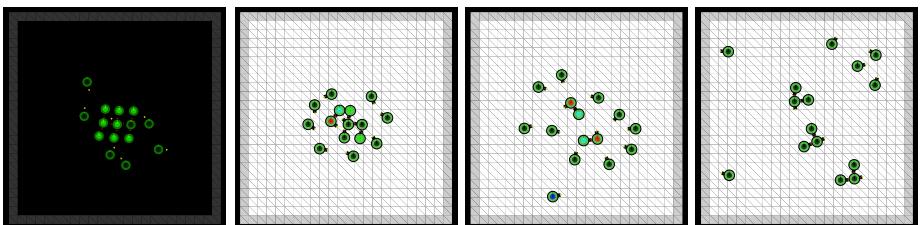


Fig. 4. Snapshots of different stages in a self-reconfiguration experiment with fifteen simulated s-bots (reconfiguration from single 9 s-bot square morphology to three 3 s-bot arrow morphologies)

SWARMORPH-script primitive for coordinated motion, these pairs travel away from the site of the original morphology in opposite directions to give themselves the space required to form subsequent morphologies. These pairs form the basis for two of the three new arrow morphologies. The s-bot that seeded the square morphology does not move, and becomes the seed for the third new arrow morphology. All other s-bots detatch and try to join growing morphologies by attaching to displayed connection slots.

6 Conclusions and Future Work

In this study, we showed how a group of real robots can autonomously reconfigure using self-assembly and local communication between connected robots. Our approach relies on a completely distributed control paradigm — the robots are all independently autonomous and rely only on local communication to cooperate. One advantage of distributed control is scalability. With large numbers of robots, different subsets of robots could perform different tasks in parallel. Since our approach is decentralized, multiple different morphologies can be formed and undergo reconfiguration at the same time.

In our ongoing research we are trying to give the robots the capability to identify different types of obstacle and to reconfigure adaptively into appropriate morphologies based on the environments they encounter.

Acknowledgments. This work was made possible by the innovative robotic hardware developed by Mondada's group at the Laboratoire de Système Robotiques (LSRO) of EPFL. This work was supported by the *SWARMANOID* project, funded by the Future and Emerging Technologies programme (IST-FET) of the European Commission, under grant IST-022888 and by the *VIRTUAL SWARMANOID* project funded by the F.R.S.-FNRS. The information provided is the sole responsibility of the authors and does not reflect the European Commission's opinion. The European Commission is not responsible for any use that might be made of data appearing in this publication. Marco Dorigo acknowledges support from the F.R.S.-FNRS, of which he is a Research Director.

References

1. Fukuda, T., Buss, M., Hosokai, H., Kawauchi, Y.: Cell structured robotic system CEBOT: control, planning and communication methods. *Robotics and Autonomous Systems* 7(2-3), 239–248 (1991)
2. Yim, M., Roufas, K., Duff, D., Zhang, Y., Eldershaw, C., Homans, S.B.: Modular reconfigurable robots in space applications. *Autonomous Robots* 14(2-3), 225–237 (2003)
3. Murata, S., Yoshida, E., Kamimura, A., Kurokawa, H., Tomita, K., Kokaji, S.: M-tran: Self-reconfigurable modular robotic system. *IEEE-ASME Transactions on Mechatronics* 7(4), 431–441 (2002)
4. Østergaard, E.H., Kassow, K., Beck, R., Lund, H.H.: Design of the ATRON lattice-based self-reconfigurable robot. *Autonomous Robots* 21(2), 165–183 (2006)

5. Shen, W., Krivokon, M., Chiu, H., Everist, J., Rubenstein, M., Venkatesh, J.: Multimode locomotion for reconfigurable robots. *Autonomous Robots* 20(2), 165–177 (2006)
6. Yim, M., Shen, W.M., Salemi, B., Rus, D., Moll, M., Lipson, H., Klavins, E., Chirikjian, G.S.: Modular self-reconfigurable robot systems. *IEEE Robotics & Automation Magazine* 14(1), 43–52 (2007)
7. Groß, R., Bonani, M., Mondada, F., Dorigo, M.: Autonomous self-assembly in swarm-bots. *IEEE Transactions on Robotics* 22(6), 1115–1130 (2006)
8. Damoto, R., Kawakami, A., Hirose, S.: Study of super-mechano colony: concept and basic experimental set-up. *Advanced Robotics* 15(4), 391–408 (2001)
9. Rus, D., Vona, M.: Crystalline robots: Self-reconfiguration with compressible unit modules. *Autonomous Robots* 10(1), 107–124 (2001)
10. White, P., Zykov, V., Bongard, J., Lipson, H.: Three dimensional stochastic reconfiguration of modular robots. In: *Proceedings of Robotics Science and Systems*, pp. 161–168. MIT Press, Cambridge (2005)
11. Jones, C., Matarić, M.J.: From local to global behavior in intelligent self-assembly. In: *Proceedings of the 2003 IEEE International Conference on Robotics and Automation, ICRA 2003*, vol. 1, pp. 721–726. IEEE Computer Society Press, Los Alamitos (2003)
12. Butler, Z., Kotay, K., Rus, D., Tomita, K.: Generic decentralized control for lattice-based self-reconfigurable robots. *International Journal of Robotics Research* 23(9), 919–937 (2004)
13. Shen, W.M., Will, P., Galstyan, A., Chuong, C.M.: Hormone-inspired self-organization and distributed control of robotic swarms. *Autonomous Robots* 17(1), 93–105 (2004)
14. Støy, K., Nagpal, R.: Self-reconfiguration using directed growth. In: *Proceedings of the Internation Conference on Distributed Autonomous Robot Systems (DARS-2004)*, pp. 1–10. Springer, Berlin (2004)
15. Støy, K.: Using cellular automata and gradients to control self-reconfiguration. *Robotics and Autonomous Systems* 54(2), 135–141 (2006)
16. Bojinov, H., Casal, A., Hogg, T.: Emergent structures in modular self-reconfigurable robots. In: *Proceedings of the IEEE International Conference on Robotics & Automation*, vol. 2, pp. 1734–1741. IEEE Computer Society Press, Los Alamitos (2000)
17. Mondada, F., Pettinari, G.C., Guignard, A., Kwee, I.V., Floreano, D., Deneubourg, J.L., Nolfi, S., Gambardella, L.M., Dorigo, M.: SWARM-BOT: A new distributed robotic concept. *Autonomous Robots* 17(2–3), 193–221 (2004)
18. O'Grady, R., Christensen, A.L., Dorigo, M.: SWARMORPH: Multi-robot morphogenesis using directional self-assembly. Technical Report IRIDIA/2008-001, IRIDIA, Université Libre de Bruxelles (2008)
19. Christensen, A.L., O'Grady, R., Dorigo, M.: SWARMORPH-script: A language for arbitrary morphology generation in self-assembling robots. *Swarm Intelligence* 2 (in press, 2008)

Beanbag Robotics: Robotic Swarms with 1-DoF Units

David M.M. Kriesel¹, Eugene Cheung², Metin Sitti², and Hod Lipson¹

¹ Computational Synthesis Laboratory, Mechanical and Aerospace Engineering
Cornell University, Ithaca, NY, USA

mail@dkriesel.com, hod.lipson@cornell.edu

² Robotics Institute, Department of Mechanical Engineering
Carnegie Mellon University, Pittsburgh, PA, USA
{eccheung,msitti}@andrew.cmu.edu

Abstract. Robotic swarm behavior is usually demonstrated using groups of robots, in which each robot in the swarm must possess full mobile capabilities, including the ability to control both forward and reverse motion as well as directional steering. Such requirements place severe constraints on the cost and size of the individual robots (*swarmers*), limiting the number of units and constraining the overall minimal size of a swarm. Here we show that similarly-complex swarm behavior can be achieved using much simpler individual swarmers. These possess significantly fewer controllable degrees of freedom, namely the ability to move forward at different velocities. We demonstrate how the interaction between different units then causes the entire swarm to obtain maneuverability unavailable at the individual level. These results may open the door to fabrication of simpler and smaller units for swarms allowing significantly larger numbers of units and smaller overall swarm footprints.

1 Introduction

Social insects, schools of fish and flocking birds often exhibit cooperative swarming behavior that enables complex tasks that individuals (*swarmers*) cannot manage separately. The system as a whole is said to accomplish more than the sum of its parts and to be scalable, robust and fault-redundant [1]. These observations have inspired studies in the field of swarm intelligence that seek to demonstrate similar properties in synthetic swarms [2,3]. Most reported demonstrations of robotic swarms use groups of individual swarmers already in possession of high degrees of locomotive freedom. This complexity leads to several consequences including higher costs, more possible points of failure and fewer redundancies and places severe constraints on the minimal size of individual swarmers and consequently on the size of the swarm. We suggest that complete navigational control can be attained using much simpler units, that have only the ability to control the speed of their noisy forward locomotion, constrained in a

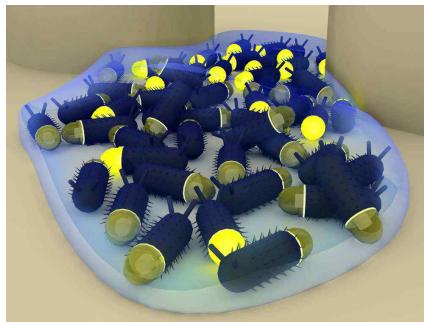


Fig. 1. Concept of a robotic system built of swimmers and a passive membrane, changing its shape to get through the gap in the background

flexible membrane (Fig. 1). We use evolutionary computation methods to demonstrate how a swarm of these simple devices can accomplish behaviors equivalent to those demonstrated by swarms composed of fully controllable individuals.

Outline. In this paper, we will first present a concept of simple swimmers (Sec. 2). We outline our principles of swimmer simulation (Sec. 3) and demonstrate first evolved behaviors (Sec. 5) in two standard experiments (Sec. 4). Following, we call attention to problems concerning the swarm and present a passive membrane as solution. We will then present the results of the same experiments performed with the membrane added (Sec. 6), and present reliability statistics. Finally, we present two physical swimmer implementations (Sec. 7) with the potential ability to be built at a very small scale.

2 Swimmer Design

Morphology, Restricted Locomotion and Steering Options. The swimmer morphology is kept very simple in our experiments: Every swimmer's body is elliptical in shape (Fig. 2). In every experiment, the sensor area is triangular, beginning at a swimmer's front. The sensor height is about $20 \cdot x$, the sensor width about $10 \cdot x$, where x is a swimmer's length (arbitrary units).

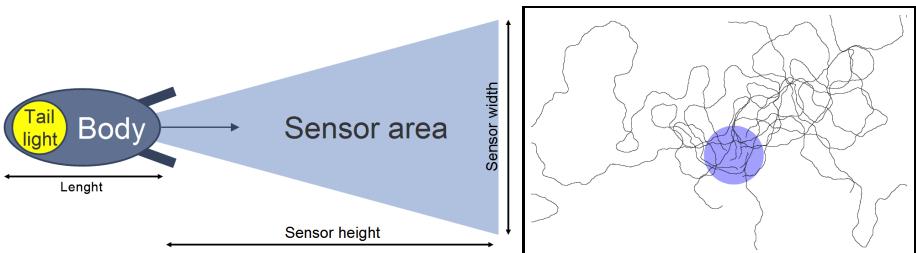


Fig. 2. Left: Individual swimmer architecture. The arrow represents the movement direction. **Right:** 10 sample trajectories by swimmers starting from the blue area.

A swarmer is equipped with a tail-light that may be switched on or off, with other swarmers able to sense its state. It only has the ability to locomote forward and is unable to control its orientation. When it moves forward, some random noise up to 30 degrees is added to its orientation, leading to stochastic movement trajectories.

Synthesized Controllers. In the experiments below, we synthesize the controllers using evolutionary algorithms [4,5]. We evolve [6] the weights of a recurrent neural net [7,8] that controls locomotion speed and the tail-light in response to input from the frontal sensor(s). All swarmers have identically evolved controllers. Fitness was determined as a function of an overall swarm behavior with regards to reaching a specific goal.

3 Simulation Principle

Simulations consist of a number of simulation steps carried out iteratively. In one step, every swarmer moves a distance defined by its speed towards the direction in which it is currently oriented. In addition to the swarmers, the world also contains two types of immobile objects. The first type constitutes obstacles that are impenetrable to the swarmers, while the second is penetrable to the swarmers, as if drawn underneath them (examples: light sources, nest objects). Objects that swarmers can sense must be visible to them, implying that they must not be entirely covered by another object or swarmer. Some experiments contained interaction with an additional, elastic passive membrane, which will be introduced later on. Being made of a closed chain of 50 links, the membrane simulation uses kinematic relaxation methods [9]. The swarmers can apply forces to every single link. In return, the entire membrane applies reaction forces to swarmers while contracting after elastic expansion. As a result, the membrane can move, expand and change its shape.

4 Experiment Setups

During all experiments, the swarmers were given one of the following two tasks: Reaching a light source, which makes it necessary for the swarm to navigate and avoid obstacles, or food foraging, which adds the challenge of collective decision making, because there are two possible locomotion goals: the food, and the nest.

Light Search. Our first goal was to create swarmers capable of reaching a light source. A swarmer is equipped with a single brightness sensor on its front. A swarmer's controller has four inner neurons, one input neuron receiving the brightness value, and two output neurons controlling the forward movement speed and tail-light. The brightness stimulus given to the input neuron increases quadratic: The nearer, the higher gets the stimulus. We assumed that a swarmer needs to gather energy from light sources to keep up an internal energy level: If it does not see the light source soon after starting, the energy level will gradually

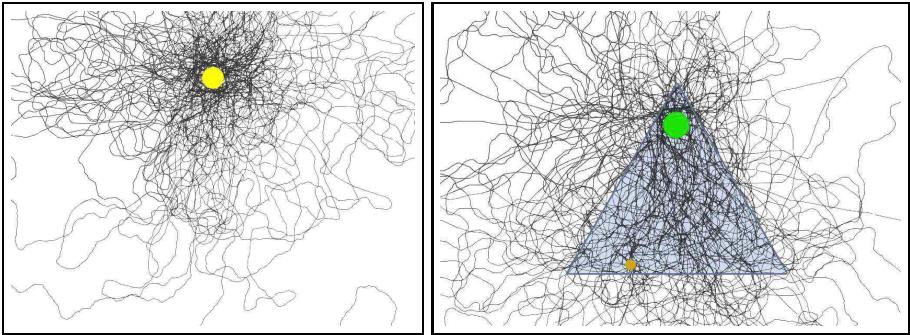


Fig. 3. Left: Traces of swarmers evolved to stay close to a light source (yellow). **Right:** Traces of swarmers evolved to collect food and bring it back to their nest (green). The food (orange) re-appears at bottom of the triangular area marked blue.

decrease to zero, so it will stop. If it sees the light source, it can gather energy from it – the closer, the more. In our fitness function f_1 (eqn. 1), e represents the gathered energy, d the distance to the light source (each measured per swarmer simulation end) and n represents the number of individuals in the swarm.

$$f_1 = \sum_{\text{swarm}} \left(\frac{e - d}{n} \right) \quad (1)$$

Food Foraging. In this experiment, food is collected by touching, and delivered to the nest by touching the nest afterwards. When a food item is collected by a swarmer, a new one appears at another (random) location, so that the number of food items is not limited by the environment. In the fitness function f_2 for this experiment (eqn. 2), f represents the number of food items delivered to the nest by an individual swarmer, measured at the end of the simulation.

$$f_2 = \sum_{\text{swarm}} f \quad (2)$$

The controller was slightly modified: It contains four input neurons for sensing swarmer tail-lights, food, the nest and whether the individual swarmer had food loaded. The stimuli for the input neurons for food, nest and tail-lights were generated analogously to the light stimuli in the first experiment. Furthermore, we increased the number of inner neurons to 5.

5 Experiments and Problems with Free Swarmers

In the light source experiment, the swarmers evolved a behavior that made them speed up when a light source was visible and slow down otherwise. The overall behavior of the swarmers was an oscillating locomotion around the light source (Fig. 3, left part). In the food foraging, too, the swarmers evolved behavior to

solve that problem (Fig. 3, right part) by accelerating and getting slower at appropriate points in time. So in general, the simple swarmers were capable of completing their tasks in both cases – but there remains a problem that needs to be addressed: Individual swarmers can easily get lost if they move too far away from the destination to sense it, or if it is occluded by obstacles.

6 Experiments with Swarmers in a Passive Membrane

A solution to the problem mentioned above is to create a passive, elastic membrane around a group of swarmers. The swarmers can apply forces to the membrane, and the resulting reaction forces are then applied to the swarmers in turn. The overall swarm movement direction is then the direction in which the majority of swarmers apply a force. In subsequent sections, we present a comparison of the light search (this time with obstacles) and the food search – both with swarmers in a membrane, and free swarmers.

Light Search. The swarmers had to get through a narrow gap to reach the target light source, which was in addition partly occluded by another obstacle. The swarm without the membrane did not evolve any successful behavior for this task (see evolution statistics in fig. 5). Although each swarmer is as simple as described, the swarm with membrane evolved a behavior which allowed it to perform directed locomotion, avoid the obstacle, morph to get through the gap and then reach and stay on the light source (Fig. 4). The controller of the light-seeking swarm enclosed in a membrane did not need to evolve dynamics, even though it had the opportunity. The more light a single swarmer sees, the faster it moves, and if the sensed brightness surpasses a certain threshold, a swarmer switches on its tail-light. This simple behavior implies that swarmers that are able to see the goal accelerate in the direction of the goal and propagate the signal to attract other swarmers that are not positioned to directly see the goal.

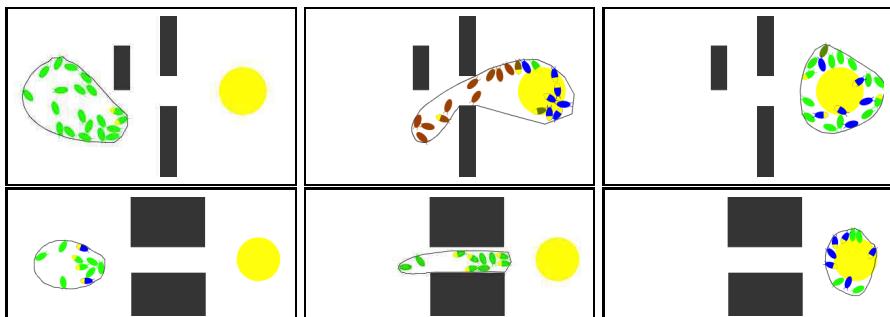


Fig. 4. Frames of the light searching swarm in a membrane with different obstacle layouts. The swarmer colors represent the energy state of the swarmers: Blue means that they are fully charged, black means that they have no energy, and their color gradually changes from green to red while discharging. A yellow tail indicates a tail-light turned on. The yellow object represents the light source, dark objects are obstacles.

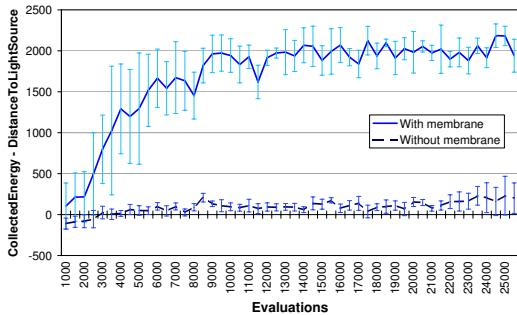


Fig. 5. Light search fitness plotted by number of evaluations. The error bars represent the standard deviation.

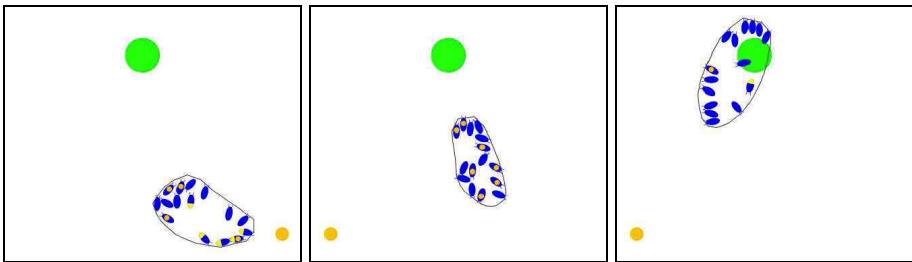


Fig. 6. Frames of the food collecting swarm in a membrane. The green object is the nest, the smaller orange objects are food objects. The food on the back of the swarmers is shown if they carry any. A yellow tail marks a tail-light turned on.

Food Foraging. This experiment was also repeated using the membrane-enclosed swarm. However, when looking at the evolution statistics (Fig. 7, left pane) the swarm without membrane seemed to be more efficient. The reasons for this effect are the following: When not contained in a membrane, individual swarmers move much faster, so that when all swarmers are in the right region, they can collect more food per time than the swarmers in the membrane. The food foraging behavior of the membrane-enclosed swarm was to collect several food items at a time and then return to the nest to deposit all of them simultaneously (Fig. 6), which is slower. But as free swarmers get lost over time, the swarm without membrane gradually loses its efficiency, while the swarm with membrane showed more robustness and reliability. The behavior of the evolved food foraging swarmer seems to follow the following rules: Whenever a swarmer sees a food object, a nest or swarmer tail-light, it accelerates. Whenever a swarmer sees a food object or another swarmer's tail-light, it switches on its own tail-light. However, seeing a nest does not cause a swarmer to switch on its tail-light. The result is an oscillating locomotion between the nest and food objects, where the swarmers are more likely to go to the food because of their tail-lights, which they only switch on when they see food in order to attract others.

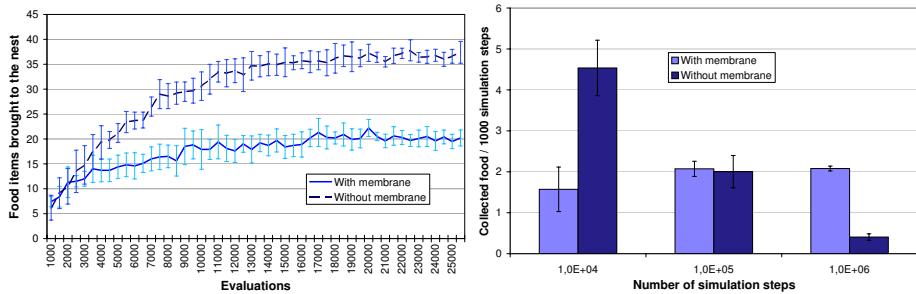


Fig. 7. **Left:** Food search fitness plotted by number of evaluations. **Right:** Performance of the evolved swarm with and without a membrane over longer periods of simulation. Error bars represent the standard deviation.

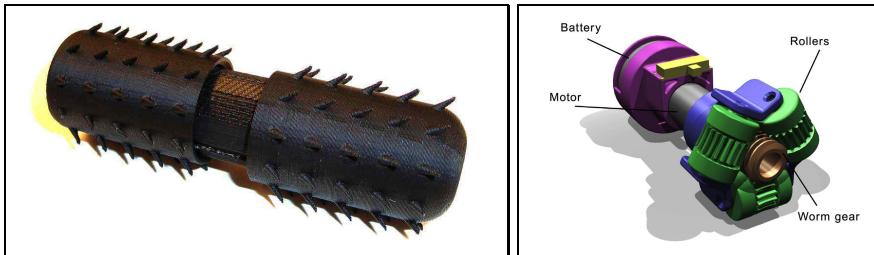


Fig. 8. Left: Implementation of the directional friction approach. The thorns that cover the surface cause directed friction, while one of the two parts vibrates along the axis of a swarmer. **Right:** A more traditional approach. A pager motor turns a worm gear that drives three rollers.

In all evolutionary runs (regardless of the specific goal) the swarmers, if put into a membrane, evolved controllers which showed the capability to locomote in an arbitrary direction as a collective behavior. In addition, no swarm ever got lost during our experiments. Without membrane, both light search and food foraging experiments were either impossible or unreliable over long time.

7 Physical Implementation

The simplified requirement of only forward, unsteerable motion, allows the consideration of much simpler robot designs. For example, simple vibratory actuation combined with non-symmetrical (directional) friction, can provide such noisy forward motion. A physical prototype imitating this behavior can be seen in fig. 8, where thorns create the directed friction and a pager motor provides vibration. We suggest that this simplified behavior is more appropriate for micro-fabrication than traditional robot designs. Another prototype capable of simple forward motion [10] can also be seen in fig. 8.

8 Conclusions and Future Work

We presented a simple swarmer that is only capable of noisy forward locomotion. Using a passive elastic membrane, we showed that this simple behavior can be channeled to achieve a variety of complex behaviors that are traditionally accomplished using significantly more complex individual swarmers. We showed a robotic system that is *highly redundant*. It is also *morphable* and capable of *robust*, relatively *complex* behavior. Furthermore, it is *variable in size* by two factors (the size of every swarmer and the number of swarmers in the membrane), and capable of *directed locomotion* even though the individuals are not. The agility of locomotion achieved by an artificial robotic swarm demonstrated in this paper, opens the door to a concept of "beanbag" robotics: Simple swarmers in a membrane. Ultimately, a robotic system consisting of large numbers of very small swarmers in a membrane could behave almost like a liquid, able to freely change its overall shape while moving in any desired direction.

References

1. Camazine, S.: Self-Organization in Biological Systems. Princeton University Press, Princeton (2003)
2. Dorigo, M., Trianni, V., Sahin, E., Groß, R., Labella, T., Baldassarre, G., Nolfi, S., Deneubourg, J., Mondada, F., Floreano, D., et al.: Evolving Self-Organizing Behaviors for a Swarm-Bot. *Autonomous Robots* 17(2), 223–245 (2004)
3. Nouyan, S., Dorigo, M.: Path formation in a robot swarm. Technical Report TR/IRIDIA/2007-002, Brussels, Belgium (February 2007)
4. Rechenberg, I.: Cybernetic Solution Path of an Experimental Problem. Farnborough Hants: Ministry of Aviation, Royal Aircraft Establishment (1965)
5. Mitchell, M.: An Introduction to Genetic Algorithms. MIT Press, Cambridge (1996)
6. Nolfi, S., Floreano, D.: Evolutionary Robotics: The Biology, Intelligence, and Technology of Self-organizing Machines. MIT Press, Cambridge (2000)
7. Rosenblatt, F.: The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological Review* 65, 386–408 (1958)
8. Kriesel, D.: A brief introduction on neural networks (2007)
9. Lipson, H.: A Relaxation Method for Simulating the Kinematics of Compound Nonlinear Mechanisms. *ASME Journal of Mechanical Design* 128, 719 (2006)
10. Sitti, M.: Microscale and nanoscale robotics systems – Characteristics, state of the art, and grand challenges. *IEEE Robotics and Automation Magazine* 14(1), 53–60 (2007)

BlåtAnt: Bounding Networks’ Diameter with a Collaborative Distributed Algorithm*

Amos Brocco, Fulvio Frapolli, and Béat Hirsbrunner

Department of Informatics, University of Fribourg, Switzerland
{amos.brocco,fulvio.frapolli,beat.hirsbrunner}@unifr.ch

Abstract. In this paper we describe BlåtAnt, a new algorithm to create overlay networks with small diameters. BlåtAnt is a fully distributed and adaptive algorithm inspired by Ant Colony Optimization (ACO), which targets dynamic and evolving networks without requiring a global knowledge. Simulation results show that our approach results in networks with a bounded diameter. This algorithm, implemented and empirically tested, will be the foundation of a fully decentralized resource discovery mechanism optimized for networks with small diameters.

1 Introduction

The “small-world phenomenon”, first observed during social studies by Stanley Milgram in the 1960’s [1], reveals that people are linked by short chains of acquaintances; that observation gave birth to the myth of “six degrees of separation”. Although not all experiments done by Milgram were successful, the interest on the topic increased, and through further research many other real-world networks (such as plane routes, power grids, etc.) were found to be instances of small-world networks. In mathematics, small-world problems concern graph theory and the distance (number of edges) between vertices in a graph. Research focuses on modeling such graphs, and reproduce their main characteristics, such as short diameters¹. In computer science, being able to reach any node following a short path is particularly interesting in distributed systems such as P2P networks or grids, even without implementing all features of small-worlds. Resource discovery is a good example of a problem that benefits from short paths.

This paper presents BlåtAnt, a distributed algorithm that augments an existing network with a minimal number of new logical links in order to minimize its diameter. The algorithm will be applied to a grid system to support efficient monitoring and resource discovering; in particular, by lowering the network diameter, we aim at being able to query virtually every peer on the network in a minimal number of hops. Our contribution is based on the collaboration between different types of mobile agents inspired by ants. In contrast to some existing algorithms, the one we detail in this paper does not enforce a fixed topology,

* Research supported by the Swiss Hasler Foundation (“ManCom Initiative”, project Nr. 2122).

¹ The diameter of the graph being the maximum of all shortest paths’ lengths.

is completely distributed and decentralized, and does not require any kind of global knowledge of the network.

The rest of this paper is organized as follows: section 2 presents research in the topic of graph augmentation algorithms, and resource discovery in small-diameter networks. Section 3 presents a description of the underlying rules used in the algorithm. Section 4 details the BlåtAnt algorithm. Section 5 provides an empirical evaluation of the distributed algorithm in static and dynamic scenarios. Finally, section 6 presents conclusions on the work done and some insights on future research directions.

2 Related Works

In our vision, the BlåtAnt² algorithm is the base for a fully decentralized resource discovery protocol that exploits the small-diameter of the network.

Although there exist various models for generating small-worlds or networks with simply a bounded diameter, see for example [2], there are only very few distributed algorithms to achieve what is commonly called small-worldization of a generic network. One of these few examples is [3], which proposes a distributed algorithm for the construction of networks with small diameter by adding a single additional link to each node. Even though this algorithm is not directly comparable to the work presented in this paper, it shows that decentralized construction of small-diameter networks based only on local information of the original topology is indeed possible.

By turning the focus to research done on P2P and grid networks, it is possible to find many distributed systems [4,5] that try to keep short distances between nodes in overlay networks even without referring them explicitly as small-worlds. These solutions are generally geared toward the problem of resource discovery, and the efficient routing of queries.

Current distributed solutions are commonly based on two techniques: distributed hash-tables (DHTs) and flooding. Whereas resource discovery using DHTs [4,5] forces structured networks in order to optimally partition the resource space and intelligently route queries to nodes that are likely to store the desired information, flooding is used mostly in unstructured networks and from a simplistic point of view, it involves querying as many nodes as possible.

In a more general way, the problem of decentralized search in small-world networks has been analyzed in [6,7], and some examples of construction of overlay networks with small-world properties are reported. Other approaches [8,9,10], let the small-world phenomenon emerge by clustering peers with similar information, whereas [11] constructs a small-world overlay network to improve the availability of resource under heavy loading. As pointed out in [12], hierarchical solutions work well for static content, but typically suffer from the mutable and heterogeneous nature of resources shared in a grid. Thus, resource discovery in unstructured and dynamically evolving networks is usually performed using

² From Harald Blåtand, the king thought to have reunited Denmark, Norway, and Sweden under a unique kingdom.

flooding algorithms. The detailed analysis on flooding mechanisms in [13] reveals that a requirement to avoid large network overheads is to limit the search space and prevent forwarding multiple copies of the same query. An *a priori* knowledge of the diameter may be used to restrict the maximum distance traveled by queries, thus limiting one of the problems of flooding.

In the same spirit, the BlåtAnt algorithm constructs and maintains an overlay network with short diameter, to provide a foundation for an optimized flooding-based resource discovery mechanism. In contrast to similar approaches, the construction step is completely separated from the resource discovery task, and it is independent from the underlying topology, the distribution of the resources, and their type. This way, beside resource discovery, other kind of distributed algorithms benefiting from the small diameter can be implemented.

3 General Idea

The BlåtAnt algorithm is meant to be executed in a distributed way on a network which will be represented, without loss of generality, as a finite graph G . In this section the general idea behind the proposed algorithm is introduced.

The goal is to augment an existing network (either physical or logical) with a minimal number of additional logical links in order to bound its diameter into a certain interval determined by a parameter D . For the rest of this paper we will refer to this process simply as *rewiring*. By applying these rewiring rules in any order, for any undirected finite graph G , with a global knowledge and in a finite number of steps, it is possible to produce a graph with a diameter less than $2D - 1$; a formal proof is provided in [14].

3.1 Connection and Disconnection Rules

Our algorithm rewrites the network according to two simple rules. The first rule is used to create an edge if the distance between two nodes is greater than a fixed threshold. The second rule is used to remove those edges that do not contribute to the solution. These rules only depend on a single integer parameter $D > 0$.

Rule 1 (Connection Rule). Let n_i and n_j be two non-connected nodes in the network graph G , and $d_G(n_i, n_j)$ the minimal distance from n_i to n_j in G . We connect n_i to n_j if:

$$d_G(n_i, n_j) \geq 2D - 1 \quad (1)$$

Rule 2 (Disconnection Rule). Let n_i and n_j be two connected nodes in the network graph G , $i \neq j$. Let $G' \leftarrow G \setminus \{n_i\}$, and N_i be the set of all nodes adjacent to n_i . Node n_i is disconnected from $n_j \in N_i$ if:

$$\exists n_k \in N_i, k \neq j : d_{G'}(n_j, n_k) + 1 \leq D \quad (2)$$

From the definition of Rule 2 it is clear that, for $D > 2$, the resulting graph has a clustering coefficient equal to zero. In other words, graphs created with our algorithm will not have full small-world characteristics.

4 BlåtAnt Algorithm Description

This section provides a description of the BlåtAnt data structures and algorithm. The idea is to globally optimize the network through successive local optimizations done by single nodes. Each node in the network executes independently by creating new ants and evaluating the creation or deletion of links; its actions are based only on local information about other nodes, which is updated by mean of ants wandering across the network. A detailed description of the algorithm is provided in [14].

4.1 Node Data Structures

The algorithm requires each node n_i to maintain some data structures containing local information and data produced during the execution.

Alpha Table α_i . Each node n_i has a α_i table which is used to store local information about the network. This table is constantly updated by mean of the information gathered through the activity of ants. As the table has a fixed maximum size, old information is purged from the table depending on the time it was last updated. Each entry contains information about another node $n_j, j \neq i$, such as the estimated distance, neighbors, local time, and remote time.

Beta and Gamma Pheromone Trails. Ants coming from n_j increase the pheromone concentration $\beta_i[n_j]$ on n_i . If this pheromone evaporates completely, n_j is assumed to be dead, and a disconnection procedure is initiated. Conversely, when an ant moves from n_i to a neighbor n_j , trail $\gamma_i[n_j]$ is reinforced; trails with high concentration becomes less desirable, thus a complete coverage of the network is ensured.

4.2 Discovery, Link and Unlink Ants

We distinguish three species of ants: Discovery Ants, used to collect and spread information about the network, Link Ants, used to perform connections between two nodes, and Unlink Ant, used to disconnect nodes.

At regular intervals, with some probability each node generates a new Discovery ant. This, combined with a maximum ant lifetime (number of wandering steps), regulates the ant population and prevents complete extinction in the event of node or network crashes.

Regardless of their species, all ants can only access local pheromone trails and information, and remember the details of the node n_j where they come from (i.e. neighbors N_j , and timestamp t_j). Information about the last visited node is handed out to the current node and used to update the Alpha table. Ants have a limited lifespan that is determined by their mission.

4.3 Frozen Connections

As methods to recover from network disconnection have not yet been implemented in the algorithm, to avoid accidentally disconnecting the network, we

freeze user-created links (including links existing before the execution of the algorithm), i.e. we do not allow the algorithm to remove them.

4.4 Timing and Pheromone Reinforcement and Evaporation

Each node n_i maintains a logical time t_i proportional to the number of incoming and outgoing ants. Using such a logical time instead of real time regulates pheromone evaporation according to the traffic, and prevents nodes with limited ant flows from clearing their information too quickly.

4.5 Algorithm Phases

The algorithm is divided in four phases: inform, evaluation, connection, and disconnection. The inform phase is executed continuously by the discovery ants while moving across the network. At regular intervals, each node evaluates if new connections need to be made and if existing connections are redundant, and can be removed. For the rest of this section, we describe the algorithm from the perspective of a node n_i .

Inform Phase. During the inform phase, discovery ants collect information and pass it to each visited node n_i , updating α_i .

Evaluating a Connection. To determine if new connections are necessary, a node has to evaluate its distance to other nodes, and check if Rule 1 applies. Since this process is based only on local information, the first step is to construct a graph \tilde{G} using the information available in the α_i table and the neighbor set N_i . Then, the distance $d_{\tilde{G}}(n_i, n_j) \forall n_j \in \tilde{G} \setminus \{n_i\}$ is computed. For each node n_j satisfying condition (1), a connection procedure is initiated by sending a Link Ant from node n_i to n_j .

Evaluating a Disconnection. The process of evaluating a disconnection is similar to the one used for a connection procedure, but it depends on Rule 2. A graph \tilde{G} based on α_i and N_i is constructed, but because frozen connections cannot be removed an additional restricted neighbor set N'_i is used beside N_i . N'_i is defined as $N'_i \leftarrow N_i \cap \Lambda \setminus \{n_j \in N_i \mid \text{link from } n_i \text{ to } n_j \text{ is frozen}\}$, where Λ contains all valid keys found in the alpha table. Thus, N'_i is the set of all neighbors with a non-null entry in the alpha table, and whose connection with n_i is not frozen. For each node in $n_j \in N'_i$, the distance $d_{\tilde{G}}(n_j, n_k) \forall n_k \in N_i \setminus \{n_j\}$, is computed, and condition (2) is checked. Eventually, disconnection of a node is achieved by sending an Unlink Ant.

Connection and Disconnection Procedures. A node n_i connects to another node n_j by first adding n_j to N_i and then updating the information in the α_i table. Conversely, disconnection from a node n_j is performed by first removing n_j from the local neighbor set N_i , thus preventing Discovery ants from migrating to n_j .

5 Evaluation

To evaluate the BlåtAnt algorithm, we conducted simulations on different topologies (including dynamic ones). We have tested our algorithm with the following topologies: a path graph of 1024 nodes, a 2D grid of size 32x32, a hypercube of 1024 nodes, and a LAN of 1281 nodes³. For each scenario, a simulation run consisted of 1280 iterations, where an iteration corresponds to a complete migration of the entire population of ants⁴. Each run was executed 42 times; details on the parameter values used during all tests, as well as additional results can be found in [14]. We present both maximum standard deviation σ_{max} of all topologies at the 1280th iteration, and the maximum mean standard deviation σ'_{max} over all iterations.

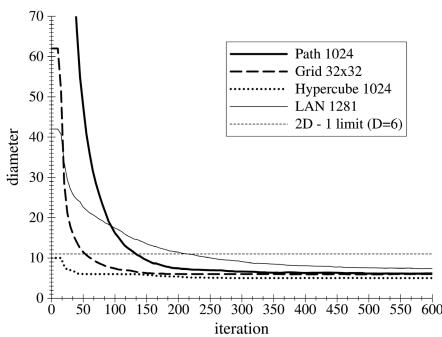


Fig. 1. Convergence of the diameter

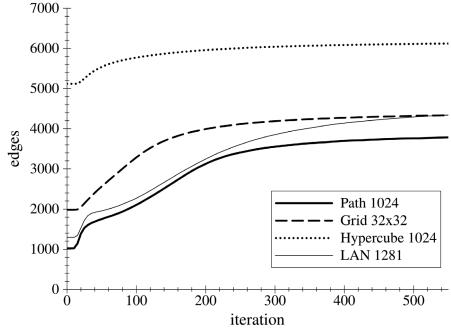


Fig. 2. Number of edges

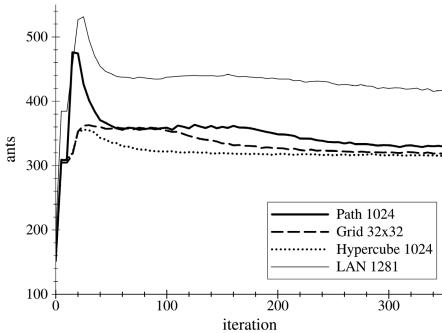
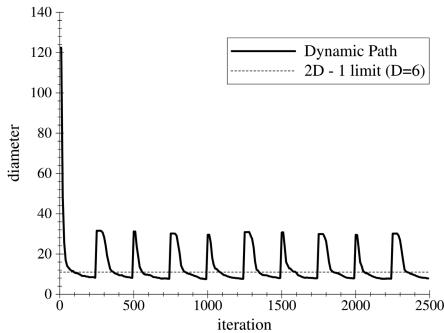
Convergence. Figure 1 shows the evolution of the network diameter in the four considered scenarios ($\sigma_{max} = 0.32$, $\sigma'_{max} = 1.16$). The diameter converges exponentially under the upper bound $2D - 1 = 11$, to a value close to $D = 6$. The *LAN 1281* topology takes more iterations because of its lower degree of connectivity, forcing ants to a longer exploration before nodes with a sufficient distance are found.

Graph Complexity. A desired property of networks generated by our algorithm is not only small diameters but also a minimal number of edges. We evaluate the minimality of the solution by computing the number of edges during the execution. Results are shown in Figure 2 ($\sigma_{max} = 51.85$, $\sigma'_{max} = 91.24$). By comparing this result with Figure 1 it is possible to notice that the number of edges stabilizes as soon as the diameter reaches its minimum.

Network Load. Figure 3 shows the number of ants created by the algorithm during its execution, which can be used to estimate the network load

³ <https://networkx.lanl.gov/browser/networkx/trunk/doc/examples/lanl.edges>

⁴ Typical execution time per iteration is 200 ms for a population of 500 ants in a 1024 nodes topology on a dual-core 2 GHz.

**Fig. 3.** Ant population size**Fig. 4.** Dynamic scenario

($\sigma_{max} = 21.97$, $\sigma'_{max} = 21.62$). Each topology starts with roughly the same population of Discovery ants, which can be estimated to 15% of the total number of nodes. When the rewiring process begins, multiple Link and Unlink ants are instantiated. As soon as an optimal diameter is found, the population starts to decrease.

Dynamic Networks. Although the BlåtAnt algorithm does not implement any mean of preventing network partitioning when a node disconnects or crashes, we propose an initial test of its behavior in dynamic networks. For that purpose we used an initial path topology consisting of 100 nodes: every 250 iterations a chain of 25 nodes is added to a random node in the graph, and every 50 iterations a node is removed. Figure 4 shows the evolution of the diameter: a minimal diameter is restored in about 100 iterations after a chain was added.

6 Conclusion and Future Works

In this paper we presented BlåtAnt, a collaborative and distributed algorithm inspired by ACO, to bound the diameter of a network without requiring a global knowledge. The algorithm uses different species of ants with different tasks in order to collect and propagate information across the network, and to create and remove links. Pheromone trails are used to direct ants toward underexploited paths, and to detect the departure of adjacent nodes. Simulations on different topologies validated the behavior of the algorithm in a fully distributed environment. Furthermore, when applied to an evolving network, the adaptive nature of the algorithm illustrated its ability to rapidly control the diameter.

There are several issues that are worth further investigation. First, a thoughtful analysis of the algorithm in large scale networks would allow us to validate its scalability. Additionally, although the algorithm performed well in the simulated dynamic network, full fault-tolerance is still lacking.

In conclusion, we believe BlåtAnt can be a foundation for a wide range of distributed algorithms that will exploit the network's shallowness, for example, a flooding based resource discovery, or an optimized network monitoring.

References

1. Milgram, S.: The small world problem. *Psychology Today* 2, 60–67 (1967)
2. Kleinberg, J.: The small-world phenomenon: An algorithmic perspective. In: Proceedings of the 32nd ACM Symposium on Theory of Computing (2000)
3. Duchon, P., Hanusse, N., Lebhar, E., Schabanel, N.: Towards small world emergence. In: SPAA 2006: Proceedings of the 18th annual ACM symposium on Parallelism in algorithms and architectures, pp. 225–232. ACM, New York (2006)
4. Ratnasamy, S., Francis, P., Handley, M., Karp, R., Schenker, S.: A scalable content-addressable network. *SIGCOMM Comp. Com. Rev.* 31(4), 161–172 (2001)
5. Stoica, I., Morris, R., Karger, D., Kaashoek, F., Balakrishnan, H.: Chord: A scalable peer-to-peer lookup service for internet applications. In: Proceedings of the 2001 ACM SIGCOMM Conference, pp. 149–160 (2001)
6. Kleinberg, J.: Complex networks and decentralized search algorithms. In: Proceedings of the International Congress of Mathematicians (ICM) (2006)
7. Sandberg, O.: Searching a small world. Master's thesis, Chalmers University (2005)
8. Zhang, H., Goel, A., Govindan, R.: Using the small-world model to improve freenet performance. In: INFOCOM 2002. 21st Annual Joint Conference of the IEEE Computer and Communications Societies, vol. 3, pp. 1228–1237 (2002)
9. Akavipat, R., Wu, L.S., Menczer, F.: Small world peer networks in distributed web search. In: WWW Alt. 2004: Proceedings of the 13th international World Wide Web conference on Alternate track papers & posters, pp. 396–397. ACM, New York (2004)
10. Clarke, I., Sandberg, O., Wiley, B., Hong, T.W.: Freenet: A distributed anonymous information storage and retrieval system. In: Proceedings of Designing Privacy Enhancing Technologies: Workshop on Design Issues in Anonymity and Unobservability, pp. 46–66 (July 2000)
11. Hui, K.Y.K., Lui, J.C.S., Yau, D.K.Y.: Small-world overlay p2p networks: construction, management and handling of dynamic flash crowds. *Comput. Netw.* 50(15), 2727–2746 (2006)
12. Iamnitchi, A., Foster, I.T.: On fully decentralized resource discovery in grid environments. In: Lee, C.A.(ed.). LNCS, Vol. 2242, pp. 51–62. Springer, Heidelberg (2001)
13. Dimakopoulos, V.V., Pitoura, E.: On the performance of flooding-based resource discovery. *IEEE Trans. Parallel Distrib. Syst.* 17(11), 1242–1252 (2006)
14. Brocco, A., Frapolli, F., Hirsbrunner, B.: Shrinking the network: The blatant algorithm. Technical Report 08-04, Department of Informatics, University of Fribourg, Fribourg, Switzerland (April 2008), <http://diuf.unifr.ch/pai>

Dependency by Concentration of Pheromone Trail for Multiple Robots

Ryusuke Fujisawa¹, Shigeto Dobata², Daisuke Kubota¹,
Hikaru Imamura¹, and Fumitoshi Matsuno¹

¹ The University of Electro-Communications, Tokyo, Japan
`{fujisawa,kubota,hikaru0420,matsuno}@hi.mce.uec.ac.jp`

² The University of Tokyo, Tokyo, Japan
`dobatan@dolphin.c.u-tokyo.ac.jp`

Abstract. In this paper, we discuss the concentration dependency of pheromone communication in swarm robotics. Instead of a pheromone trail and the insect antenna, we used ethanol and an alcohol sensor. This experimental system has a trade-off problem; high concentrations of the pheromone yield high signal strength but the signal duration is short, while low pheromone concentrations yield low signal strength but a long signal duration. We examined the optimal pheromone concentration for a swarm of robots. For this purpose, we developed a swarm behaviour algorithm and swarm robots that communicate using a pheromone trail. In addition, we discuss the effects of the pheromone concentration.

1 Introduction

A pheromone is any chemical or set of chemicals produced by a living organism that transmits a message to other members of the same species [1]. There are two types of pheromone: releaser pheromone (sexual, alarm, trail and aggregation pheromones) and primer pheromones (queen substances) [2]. In this study, we focus on “trail pheromones” for communication. A swarm is constructed by a large number of agents, and can search for and collect multiple objects. The ants use the trail pheromone to forage for prey. This communication method is chemical, local, indirect and plastic. The trail pheromone is a volatile substance, and evaporates over time.

Several studies using real or virtual pheromones have been reported previously. Sugawara *et al.* [3] and Garnier *et al.* [4] achieved the foraging behaviour of ants using a swarm of robots and a virtual pheromone (with a projector and screen). These studies represented a well-conceived measurement system. Pheromone diffusion is an important factor in real pheromone studies, and is a very difficult problem. To adjust the duration of the pheromone signal, the concentration of the pheromone should be changed or it should be mixed some other substance(s). In addition, there are few advantageous chemical sensors. The use of a virtual pheromone solves these problems. Shimoyama *et al.* [5] achieved pheromone tracking behaviour using the real insect antenna and pheromone,

but they did not consider swarm behaviour, and the use of biomaterials is difficult for swarm robotics. Purnamadjaja *et al.* [6] studied swarm robots that communicate using two real chemical substances. The latter regulates a gas sensor in a refined way. However, only one robot secretes the pheromone, so this system involves only one-sided communication.

In this study, we focused on the concentration of the pheromone when robots search for objects and attract other robots. We constructed a swarm behaviour algorithm and developed a swarm of robots that communicate using the pheromone trail. In addition, the effect of the pheromone concentration is also discussed.

2 Swarm Behaviour Algorithm

We simplified the algorithm of Kurumatani [7] to attract another agent. In this paper, the experimental field was limited, and included only agents, prey and nest, similarly to the study of Kurumatani [7]. All agents can detect the direction of the nest, which we feel is a reasonable assumption. For example, fire ants (*S. invicta*) have been shown to detect the location of the nest based on the direction of the sun [8].

The algorithm is described by the deterministic finite automaton as shown in Fig. 1. To design this algorithm, we defined 3 states $S_i(i = 1, 2, 3)$, 4 perceptual cues (stimuli) $P_i(i = 1, \dots, 4)$ and 3 effector cues (actions) $E_i(i = 1, 2, 3)$. We assume that there are many agents in the field, and that all agents can detect the location of the nest as in the case of ants. As shown in Fig. 1, the agent whose state is S_i selects the action E_i . If the agent in state S_i detects the perceptual cue P_j , the state of the agent is transited to S_k . The details of the internal states of the robot, perceptual cues and effector cues are as follows. S_i : S_1 , Search (Agent does not have any information of the prey); S_2 , Attraction (Agent has the location information of the prey); S_3 , Tracking (Agent has only the direction information of the location of the prey). P_i : P_1 , Contact with prey; P_2 , Nest arrival; P_3 , Presence of pheromone; P_4 , Timeout occurs. E_i : E_1 , Random walk; E_2 , Secrete pheromone along the nest direction; E_3 , Follow the pheromone path toward the prey.

In [7], Kurumatani proposed a global-level algorithm of the attraction behavior, so it is difficult to consider collision of robots. Hence, we should model the collision phenomena of robots. We propose action rules when the robot detects the collision as shown in Table 1. The traffic jam problem on the pheromone trail is solved by following algorithm.

Agents have different actions after the collision. The internal state of the agent is detected by itself. The purpose of the swarm is “attracting other agents using the pheromone”. So we introduce priority of the action of agents by its internal state. We define the order of high priority as following; 1) S_2 (stopping

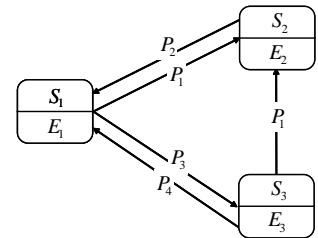


Fig. 1. State transition rule for swarm behaviour

Table 1. The action rules when an agent detects collision

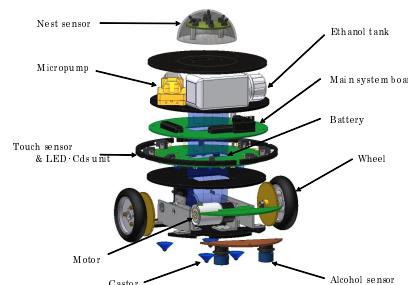
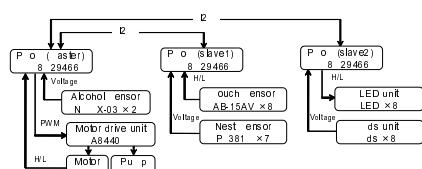
State of agent	Collision object	Action of agent after collision
S_1	agent or wall	disengage from collision point and turn around
	prey	lay down the pheromone trail
S_2	agent	temporary stop
	prey or wall	(out of the model)
S_3	agent or wall	disengage from collision point
	prey	lay down the pheromone trail

and waiting for avoidance of other agent.), 2) S_3 (acting by the rule in Table 1 and tracks the pheromone trail.), 3) S_1 (avoiding other agent.). After collision, a high priority agent makes the action based on the rule in Table 1, and a low priority agent stops for a given time or avoids the high priority agent and then makes the action.

3 Experimental Robot

To demonstrate the validity of the proposed swarm behaviour algorithm and the effects of the pheromone concentration, we developed a robot designated ARGOS-01. Figure 2 shows a photograph of the robot and its construction of robot. This is a fully autonomous robot comprised of 3 microcomputers and sensors as shown in Fig. 3.

We used PSoC (Programmable System-on-Chip) supplied by Cypress Semiconductor Corp. These microcomputers are connected with each other by I²C (Inter-Integrated Circuit). Two alcohol sensors are used as pheromone sensors to trace the pheromone trail. Eight LEDs and 8 Cds are used to find the neighbouring prey. Eight switches detect collisions with other robots, the prey or the wall as touch sensors. Seven infrared phototransistors detect the direction of the nest consisting of infrared lamps.

**Fig. 2.** Construction of ARGOS-01**Fig. 3.** Outline of the system of ARGOS-01

The robot has 2 active wheels and 4 castors. Its active wheels can be controlled independently, so the robot can move freely on a flat plane. The specifications of ARGOS-01 are as follows: body diameter, 150[mm]; height, 195[mm]; weight, 1.26[kg]; and maximum speed, 0.1[m/s].

The robot detects 4 perceptual cues (P_1, \dots, P_4). The robot has 4 sensors: Touch sensor (P_1), Nest sensor (P_2), Alcohol sensor (P_3) and Internal timer (P_4). The robot detects contact between with other objects (other robots, prey and wall) via the touch sensor. In addition, the robot detects the direction and distance to the nest with the nest sensor. The robot detects its absolute position. In this study, we used an electric infrared lamp as the nest. In addition, we used ethanol (C_2H_5OH) as the pheromone. The ethanol volatilises at normal temperature, and we expected that the chemical, plastic, indirect and local effects would be similar to those of a real pheromone.

The robot should perform 3 actions (E_1, E_2 , and E_3). To drip the ethanol and lay down the pheromone trail, we used a micropump, and the robot had an alcohol tank in place of the secretory glands of an insect. Alcohol sensors were used to detect the ethanol, and the robot traced the pheromone trails by a mechanism emulating the behaviour of ants. The robot communicates with the other robots indirectly via the pheromone trails. In the actual robotic system, ethanol is used as a substitute for the pheromone. The robot ejects the pheromone from the bottom of the body when the robot is in the attraction state (S_2). The ethanol is perceived as a pheromonal perceptual cue by the other robots in state S_1 via the ethanol sensors, which are used to detect and trace the pheromone trail. The tracing mechanism is analogous to that of ants, which detect pheromone trails using two right and left antennae [9]. When the right (left) sensor detects the alcohol, the left (right) wheel is driven, and the robot moves to the right (left) as shown in Fig. 4. To imitate this action, two ethanol sensors were installed on the bottom of the body of the robot. The attached angles are $\pi/6$ from the direction of forward movement as shown in Fig. 4.

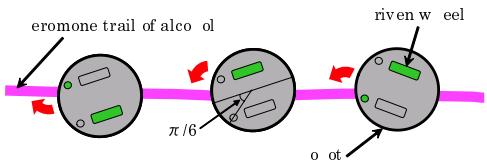


Fig. 4. Behavior on the pheromone trail

4 Experiment

The proposed behaviour algorithm has been implemented in a swarm of robots. Experiments were performed to verify the effects of the pheromone concentration. The number of robots in the swarm was 10, and the size of the field was 1800[mm] \times 1800[mm]. A nest was set at one corner, with the prey set at the opposite corner as shown in Fig. 5.

We monitored the state of the environment for 20 [min]. The ethanol concentration was set as 0, 25, 50, 75 and 100%, and the experiments were carried out 10 times for each setting. The robot found the prey, changed its behaviour

and then went to the nest while laying down a pheromone trail. When the robots detected the prey, they dripped the alcohol as a pheromonal signal, and the temperature of the dripped part was lower than the initial temperature.

Figure 6 shows a photograph of the normal camera and the temperature distributions of the field determined by thermography. At $t = 0[\text{min}]$ (Fig. 6-A, A'), 10 robots were put in the experimental field. At $t = 10[\text{min}]$ (Fig. 6-B, B'), multiple robots laid down and reinforced the pheromone trail.

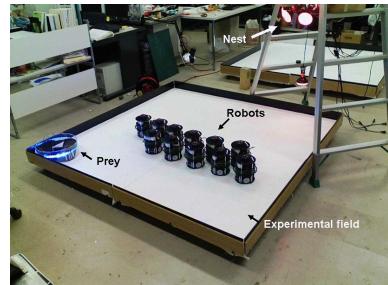


Fig. 5. Experimental field for swarm of robots

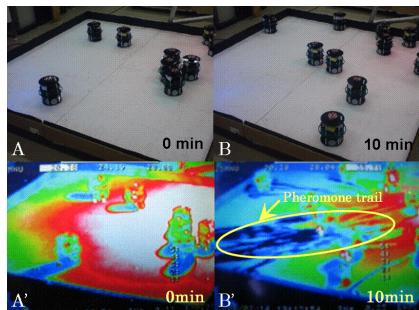
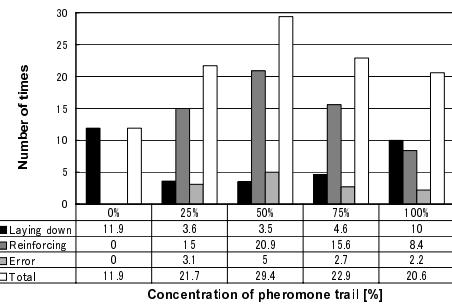


Fig. 6. Snapshots of experimental results

Fig. 7. Dependency by concentration of pheromone trail

Figure 7 shows the relationship between the pheromone concentration and the performance of the swarm. These data are the averages of 10 experiments. The pheromone concentration is shown on the horizontal axis, while the number of times that agents made contact with the prey is shown on the vertical axis. “Laying down” is when the internal state of the agent transits to S_2 (Attraction) from S_1 (Search), while “Reinforcing” is the action in which the internal state of the agent transits to S_2 from S_3 (Tracking). “Error” means that “Laying down” or “Reinforcing” failed. In addition, “Total” is the summation of “Laying down”, “Reinforcing” and “Error”. As shown in Fig. 7, the number of times that agents made contact with the prey was lower at a pheromone concentration of 0% than at the other concentrations examined. At 0% pheromone, the robots generated the pheromone trail a total of 11.9 times, and at 25%, 50%, 75% and 100% pheromone, the robots generated the pheromone trail 21.7, 29.4, 22.9 and 20.6 times, respectively. These observations indicated that the pheromone communication was an effective method for this search task.



5 Discussion

We used ethanol instead of a pheromone in this system. The ratio of the mixture of ethanol and water (H_2O) had a trade-off problem for the robots; a high concentration of the pheromone (ethanol) yielded high signal strength but the signals were of short duration, while a low concentration of the pheromone yielded low signal strength but the signals were of long duration. This trade-off is a very interesting problem for the swarm. When a behavioural field is a narrow space, a short signal duration is needed from the viewpoint of the dynamics problem. If the signal duration is too long, the pheromone trail loses plasticity, and the function for the dynamic problem is deteriorated. This example indicates that a suitable concentration of the pheromone is needed.

When 100% pheromone was used, it was vaporised easily and the information therefore did not remain in the experimental field for a long time and it was difficult for robots to detect the pheromone. Thus, “Laying down” and “Reinforcing” became about the same. At 50% pheromone, the robots showed the highest performance. These observations indicated that the performance of the swarm of robots is influenced by the pheromone concentration.

In the system developed here, the pheromone trails constantly evaporate and diffuse into the atmosphere, and are updated by the robots. This feature excels in finding a solution to a dynamic problem. There are two possible methods of increasing the performance of the swarm of robots: 1) increasing the amount of the pheromone, or 2) increasing sensor sensitivity. However, when we used these methods without sufficient thought, the ability of the swarm of robots related to the dynamic problem was reduced as described previously.

The system presented here mimics the recruitment behaviour of social insects - a form of communication that brings nestmate workers to the same place where collective action (foraging or nest site movement) is required [10]. A typical example has been reported in foraging recruitment behaviour of fire ants (*Solenopsis*) [11]. During recruitment and subsequent collective actions, trail following or recruitment pheromones (“trail pheromones”) are of great importance in mass communication.

Our results indicated that there is an optimal moderate ratio of trail pheromones when secreted with the non-pheromone component in mixed solution. In ants, the trail pheromones are mainly secreted from the poison gland, Dufour’s gland or hindgut, which are also known to be sources of a number of chemicals [12]. These chemicals are sometimes highly specialised pheromones in some social insects with complex communication systems. Therefore, trail pheromones are often secreted along with chemicals that do not have trail pheromone activity, which may affect the volatility of the trail pheromones. Generally, the volatility of chemicals in mixed solution is proportional to the number of molecules of the chemicals on the surface of the solution. Thus, the volatility of trail pheromones becomes lower than that in pure form. Although our system employed a water-ethanol mixture, and the details of volatility are affected by molecular interactions among the components, the global trend must hold in any

pheromone system using mixed solutions. The physicochemical properties of real trail pheromone mixtures should be investigated in future studies.

Previous studies that identified trail pheromones assayed only the activity of the chemicals artificially dissolved in organic solvents (e.g., acetone or hexane, which volatilise very quickly), and did not consider volatility in varying concentrations in “bio-solvents”. Some studies indicated that intermediate quantities of trail pheromone molecules are optimal for trail following [13] [14]. These results are discussed in the context of sensitivity of the insect antennae to the pheromone: a larger quantity results in too broad a pheromone field to follow a discrete trail, and smaller quantities result in failure of trail detection. Our results provided insight into the novel aspect of pheromone concentration, and suggest that social insects evolved the optimal intermediate ratio of trail pheromones in mixed solutions: higher concentrations have problems with regard to volatility, while the lower concentrations are problematic for sensitivity. This holds even for equal quantities of the molecules themselves.

Our results have several further implications. First, our system may have flexibility in changing environments. The recruitment and subsequent trail following systems should be robust against fluctuations in the external microenvironment, such as temperature, humidity, and surface substrates. In species in which trail pheromones are single active substances, pheromone concentration in mixed solutions may provide a way to deal with such changing environments by altering the concentration of the trail pheromone. Real biological systems should be explored along with investigation of the flexibility of our robot systems.

Second, in some social insects, trail pheromones are known to be multicomponent chemicals. This has been discussed in the context of guaranteeing species specificity or synergism [12]. In some species, the optimal trail following activity is achieved with some intermediate mixture ratio of the components [15] [16] [17]. Although each component has trail following activity in pure form in the above examples, our system considering the volatility in mixtures may partially explain this phenomenon.

Third, our results may explain why several ant species, which vary in life history strategy, can use the same chemical as the trail pheromone. In the ant subfamily Myrmicinae, alkylpyrazines secreted from the poison gland are often the source of trail pheromones [18]; 3-ethyl-2,5-dimethylpyrazine is used frequently. Some phylogenetic constraints may explain the commonality, but these species seem to be well-differentiated in life history strategies, such as foraging and nest relocation, requiring fine-tuning of the trail pheromone activity. These species may use different concentrations of the substances in mixed solution. Thus, further comparative studies of pheromone concentration and life history strategies are required.

References

1. Karlson, P., Butenandt, A.: Pheromones (ectohormones) in insects. Annual Review of Entomology 4, 39–58 (1959)
2. Matsuoka, M., Kitano, H., Matsumoto, T., Oono, M., Gokan, N.: Biology of Insect. Tamagawa university publishing (1992) (in Japanese)

3. Sugawara, K., Kazama, T., Watanabe, T.: Foraging behavior of interacting robots with virtual pheromone. In: Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, vol. 3, pp. 3074–3079 (2004)
4. Garnier, S., Tache, F., Combe, M., Grimal, A., Theraulaz, G.: Alice in pheromone land: An experimental setup for the study of ant-like robots. In: Swarm Intelligence Symposium, 2007. SIS 2007, 1-5 April 2007, pp. 37–44. IEEE, Los Alamitos (2007)
5. Shimoyama, I., Kanzaki, R.: Biological type micromachine. Journal of society of biomechanisms 22(4), 152–157 (1998) (in Japanese)
6. Purnamadjaja, A.H., Russell, R.A.: Guiding robots' behaviors using pheromone communication. Autonomous Robots 23(2), 113–130 (2007)
7. Kurumatani, K.: Macro-model generation for emergent cooperative behaviors in ant colony's foraging (1) -a simple model case. Journal of the Japanese Society for Artificial Intelligence 15(5), 829–837 (2000) (in Japanese)
8. Wilson, E.O.: Sociobiology. Sinsisakusya (1999) (in Japanese)
9. Hangartner, W.: Spezifitat und inaktivierung des spurpheromons von lasium fuliginosus latr. und orientierung der arbeiterinnen im duftfeld. Zeitschrift für vergleichende Physiologie 57, 103–136 (1967)
10. Wilson, E.: The Insect Societies. The Belknap Press of Harvard University Press, Cambridge (1971)
11. Wilson, E.: Chemical communication among workers of the fire ant (*Solenopsis saevissima*) (fr. smith). Animal Behaviour 10(1-2), 134–164 (1967)
12. Hölldobler, B., Wilson, E.: The ants. The Belknap Press of Harvard University Press, Cambridge (1990)
13. Evershed, R., Morgan, E., Cammaerts, M.: 3-ethyl-2,5-dimethyl-pyrazine, the trail pheromone from the venom gland of eight species of (*Myrmica*). Insect Biochemistry 12, 383–395 (1982)
14. Calenbuhr, V., Chrétien, L., Deneubourg, J.L., Detrain, C.: A model for osmotropotactic orientation ii. Journal of theoretical Biology 158, 395–407 (1992)
15. Attygalle, A., Morgan, E.: Trail pheromone of the ant (*Tetramorium caespitum*) 1. Naturwissenschaften 70, 364–365 (1983)
16. Billen, J., Beeckman, W., Morgan, E.: Active trail pheromone compounds and trail following in the ant (*Atta sexdens sexdens*) (hymenoptera, formicidae). Ethology, Ecology and Evolution 4, 197–202 (1992)
17. Janssen, E., Übler, E., Bauriegel, L., Kern, F., Bestmann, H., Attygalle, A.B., Steghaus-Kovac, S., Maschwitz, U.: Trail pheromone of the ponerine ant (*Lepisiotagena pesquetai*) (hymenoptera: Formicidae): a multicomponent mixture of related compounds. Naturwissenschaften 84, 122–125 (1997)
18. Billen, J., Morgan, E.: Pheromone communication in social insects: sources and secretions. In: Vander Meer, R.K., Breed, M.D., Espelie, K.E., Winston, M. (eds.) Pheromone Communication in Social Insects, pp. 3–33. Westview Press (1998)

Dissemination of Information with Fair Load Distribution in Self-organizing Grids

Agostino Forestiero, Carlo Mastroianni, and Giandomenico Spezzano

Institute of High Performance Computing and Networking

ICAR-CNR, Rende, Italy

{[forestiero](mailto:forestiero@icar.cnr.it),[mastroianni](mailto:mastroianni@icar.cnr.it),[spezzano](mailto:spezzano@icar.cnr.it)}@icar.cnr.it

Abstract. This paper presents an ant-inspired algorithm for building a self-organizing information system of a Grid. Ant-inspired mobile agents travel the Grid through P2P (peer-to-peer) interconnections and disseminate “descriptors”, i.e., metadata information about available Grid resources. Descriptors are reorganized and spatially sorted over the Grid, thus facilitating resource management and discovery. Moreover, agents distribute descriptors so as to respect the different capabilities of Grid hosts: hosts with higher storage capacity are assigned a larger number of descriptors than low capacity hosts. The effectiveness of the presented algorithm is assessed by event-driven simulation which proves that the simple operations performed by mobile agents successfully achieve both descriptor reorganization and fair load distribution.

1 Introduction

Owing to the inherent scalability and robustness of P2P algorithms, several P2P approaches have recently been proposed for resource organization and discovery in distributed environments and specifically in Grids [1]. The main goal of these approaches is to allow users to locate Grid resources, either hardware or software, which have the required characteristics. This is reduced to the problem of finding resource *descriptors*, which are metadata documents through which it is possible to obtain information and access the resources. Descriptors are usually indexed through bit strings, or *keys* that can have a semantic meaning (for example, each bit may indicate if the resource focuses on a specific topic or provides a specific functionality) or can be obtained through a hash function. In the latter case, the hash function is often “locality preserving” [2], which assures that resources having similar characteristics are associated to similar descriptor keys.

In this paper, we present an approach for the construction of a P2P-based Grid information system, which is inspired by the behavior of some species of ants that cluster items within their environment [3] [4]. The devised algorithm is able to disseminate and reorder descriptors in order to facilitate and speed up discovery operations. Replication and relocation of descriptors are achieved by means of simple *pick* and *drop* operations performed by ant-like mobile agents. These agents probabilistically copy and relocate descriptors with a tendency to remove a descriptor that differs significantly from other descriptors in current

network neighborhood and place it where it is more similar to its neighbors. These operations allow the possible initial equilibrium (if descriptors having different keys are uniformly distributed among hosts) to be broken, and then reinforce the spatial separation and ordering of descriptors.

The ant algorithm is an improved version of the algorithm published in [5]. The enhancement presented here takes into account the storage capacity of hosts, and aims to cope with the problem of fairly distributing descriptors among hosts, which is a critical issue for the efficient operation of P2P systems. To achieve this objective, enhanced pick and drop operations are defined so as to facilitate the pick operations in hosts with low storage capacity and the drop operations in hosts with high storage capacity.

In summary, the presented algorithm concurrently achieves multiple objectives: (i) it *replicates* and *disseminates* descriptors; (ii) it *spatially sorts* them, so that descriptors with similar indexes are placed in neighbor hosts; (iii) it adapts the distribution of descriptors to the storage capabilities of different hosts. Moreover, thanks to the self-organizing nature of the ant-based approach, agent operations spontaneously adapt to the ever changing environment, for example to the joins and departs of Grid hosts and to the changing characteristics of resources. In this paper, we present the results of an event-based simulation analysis, which shows that the algorithm successfully achieves the mentioned objectives.

2 Reorganization and Fair Distribution of Descriptors

As a peer connects to the network, with a given probability it generates a mobile agent that will travel the Grid through P2P interconnections and offer its contribution to the reorganization of descriptors. Whenever an agent arrives at a new host, it operates as follows: (i) if the agent does not carry any descriptor, it evaluates the *pick probability function* for every descriptor stored in this host, so as to decide whether or not to pick this descriptor; (ii) if the agent carries some descriptors, it evaluates the *drop probability function* for each carried descriptor, so as to decide whether or not to drop it in the current host. The *pick* and *drop* operations are driven by the corresponding probability functions that are defined and discussed in the following.

The pick probability function, as well as the drop probability function discussed later, is defined starting from the similarity function $f(\bar{d}, R)$ reported in formula (1). This function measures the average similarity of a given descriptor \bar{d} with all the descriptors d located in the *visibility region* R . The visibility region includes all the hosts that are reachable from the current host with one hop. In formula (1), N_d is the overall number of descriptors maintained in the region R , and $H(d, \bar{d})$ is the Hamming distance between d and \bar{d} . B is the number of bits that are contained in descriptor keys. The parameter α defines the similarity scale [6]; here it is set to $B/2$, which is half the value of the maximum Hamming distance between binary vectors having B bits. The value of $f(\bar{d}, R)$ assumes values ranging between -1 and 1, but negative values are truncated to 0.

$$f(\bar{d}, R) = \frac{1}{N_d} \cdot \sum_{d \in R} \left(1 - \frac{H(d, \bar{d})}{\alpha}\right) \quad (1)$$

The probability of picking a descriptor \bar{d} from the current host must satisfy two basic requirements:

(i) it must be inversely proportional to the average similarity $f(\bar{d}, R)$, thus obtaining the effect of averting a descriptor from co-located dissimilar descriptors. As soon as the possible initial equilibrium is broken (i.e., descriptors having different keys begin to be accumulated in different Grid regions), a further reorganization of descriptors is increasingly driven, because the probability of picking a dissimilar descriptor increases.

(ii) it must be inversely proportional to the storage capacity of the current host. This assures that in steady conditions high capacity hosts store more descriptors than low capacity hosts, thus respecting the different characteristics of the hosts in a Grid. To cope with this requirement, each host must estimate the average capacity of a Grid host or, more specifically, the average amount of storage space that is offered by a host to store resource descriptors. Estimation is achieved through an exchange of messages with neighbor hosts. Each host assigns a reference value of 1 to the estimated average value and assigns itself a storage index that is proportional to the amount of storage space that this host offers to the network. For example, a host assigns itself a capacity index equal to 5 if it offers an amount of storage space that is 5 times the estimated average storage space offered by a generic host.

The pick probability function P_{pick} , reported in formula (2), is evaluated by an agent for each descriptor \bar{d} stored in the local host, to probabilistically decide whether or not to pick this descriptor. The value of P_{pick} is obtained as the product of two factors. The first factor is inversely proportional to $f(\bar{d}, R)$, the average similarity of the descriptor under consideration with all the other descriptors stored in the visibility region R . The second factor takes into account the capacity of the current peer L_{peer} , and the average capacity of a generic peer \bar{L} , which is set to 1 as discussed before. The formula assures that it is more probable to pick a descriptor if it is an outlier in the local region and/or if the local host has less storage capacity than the estimated average.

$$P_{pick} = \left(\frac{k_p}{k_p + f(\bar{d}, R)} \right)^2 \cdot \left(\frac{k_{pl}}{k_{pl} + \frac{L_{peer} - \bar{L}}{\bar{L}}} \right)^2 \quad (2)$$

In the P_{pick} formula, the parameters k_p and k_{pl} can be tuned to modulate the relative impact of the two factors. In particular, the parameter k_p assumes a value between 0 and 1 and can be used to tune the degree of similarity among descriptors. In fact, the first factor of the pick probability function approaches 1 when $f(\bar{d}, R)$ is much lower than k_p (meaning that \bar{d} is extremely dissimilar from the other descriptors) and 0 when $f(\bar{d}, R)$ is much larger than k_p (meaning that \bar{d} is very similar to others descriptors). Here k_p is set to 0.1, as in [3]. Conversely, the value of k_{pl} assumes a value greater than 1, to assure that the denominator of the second factor is greater than 0. In the case that the value of P_{pick} value

exceeds 1, it is truncated to 1, which corresponds to having a 100% probability of picking a very dissimilar descriptor and/or of picking a descriptor from a host with very low capacity.

The pick operation can be performed with two different modes, *copy* and *move*. If the *copy* mode is used the agent, when executing a pick operation, leaves the descriptor on the current host, *generates a replica* of it, and carries the new descriptor until it drops it into another host. Conversely, with the *move* mode, an agent picks the descriptor and *removes* it from the current host. Each agent first operates in the copy mode, than it switches to the move mode, in order to prevent an excessive proliferation of replicas, which would hinder the correct spatial sorting of descriptors. This mechanism is better described in [5].

After picking some descriptors, an agent must decide whether or not to drop them in the hosts through which it passes. For each carried descriptor \bar{d} , the agent evaluates the drop probability function P_{drop} , which as opposed to the pick probability, must be: (i) directly proportional to the similarity function $f(\bar{d}, R)$, i.e., to the average similarity of \bar{d} with the descriptors maintained in the visibility region; (ii) directly proportional to the storage capacity of the current host. P_{drop} is defined in formula (3), which satisfies the two mentioned requirements. In (3), the parameter k_d is set to a higher value than k_p , specifically to 0.5, in order to limit the frequency of drop operations. This is useful to let the agents carry descriptors to appropriate Grid regions, without dropping them too early. In the same fashion as k_{pl} in formula (2), k_{dl} must be given a value higher than 1.

$$P_{drop} = \left(\frac{f(\bar{d}, R)}{k_d + f(\bar{d}, R)} \right)^2 \cdot \left(\frac{k_{dl}}{k_{dl} + \frac{L - L_{peer}}{L_{peer}}} \right)^2 \quad (3)$$

3 Performance Evaluation

The performance of the ant algorithm was evaluated with an event-based simulator. A Grid network having a number of hosts N_p equal to 2500 is considered. Hosts are linked through P2P interconnections, and each host is connected to 4 peers on average. The topology of the network is built using the scale-free algorithm defined by Albert and Barabasi [7], which incorporates the characteristic of preferential attachment (the more connected a node is, the more likely it is to receive new links) that was proved to exist widely in real networks.

Peers can go down and reconnect. The *average connection time* of a peer is generated according to a Gamma probability function, with an average value set to 100,000 seconds. To maintain a stable number of agents, the lifecycle of agents is correlated to the lifecycle of peers. When joining the Grid, a host generates an agent with a probability P_{gen} , and sets the life-time of this agent to its own average connection time. This setting assures that the overall number of agents is nearly equal to the number of peers times P_{gen} . In our experiments, P_{gen} is set to 0.5, therefore the number of agents is about half the number of peers. Every time a peer disconnects from the Grid, it discards the descriptors previously

deposited by agents, thus contributing to the removal of obsolete descriptors. The average time T_{mov} between two successive agent movements is set to 60 s, whereas the maximum number of P2P hops that are performed in a single agent movement is set to 3. The number of resources published by each host is obtained with a Gamma stochastic function with an average value equal to 15. Resource descriptors are indexed with bit keys having B bits. Descriptor keys are obtained through the application of a locality preserving hash function [2]. This guarantees that similar keys are given to descriptors of similar resources.

The effectiveness of the ant algorithm is evaluated through the spatial *homogeneity function* H . Specifically, for each peer p , the average homogeneity H_p of the descriptors located in the visibility region of p , R_p , is calculated. This is obtained, as shown in formula (4), by averaging the Hamming distance between every couple of descriptors in R_p and then subtracting the obtained value from B , which is the maximum Hamming distance. Thereafter, the value of H_p is averaged over the whole Grid, as formalized in formula (5).

$$H_p = B - \text{AVG}_{\{d_1, d_2 \in R_p\}}(H(d_1, d_2)) \quad (4)$$

$$H = \frac{1}{N_p} \cdot \sum_{p \in Grid} H_p \quad (5)$$

The objective is to increase the homogeneity function as much as possible, because it would mean that similar descriptors are actually mapped and aggregated into neighbor hosts, and therefore an effective sorting of descriptors is achieved. In [5], several performance results concerning the basic version of the algorithm are discussed. The present work, however, focuses on the ability of the enhanced version of the algorithm of achieving a satisfactory load distribution among hosts that have different storage capabilities.

In the literature, the storage capacity of hosts is often assumed to be distributed according to some statistical distribution, for example, the Pareto distribution. Here we assume a simpler distribution, which facilitates a more accurate and assessable analysis of our algorithm. Specifically, Grid hosts are divided into two classes: low capacity and high capacity hosts. They can correspond to ordinary personal computers and high capacity servers, respectively. It is assumed that half the load of the system is equally shared among the hosts of each class, and we vary the percentage of hosts that belong to the two classes. The following notation is adopted: the pattern $\{H : L\}$ means that $H\%$ ($L\%$) of the hosts are high (low) capacity ones, and that the load of each host is obtained by sharing half the overall capacity of the system among the hosts of each class. Following this notation, we analyzed the behavior of the algorithm with patterns $\{10 : 90\}$, $\{20 : 80\}$, $\{30 : 70\}$, $\{40 : 60\}$, and $\{50 : 50\}$. Of course, the last case corresponds to a scenario, used for comparison purposes, in which all the hosts have approximately the same capacity.

A set of simulation experiments were performed to assess the distribution of load obtained with our algorithm. In these experiments, the number of bits B in resource descriptor indexes is equal to 4. Figure 1 shows that the work of agents

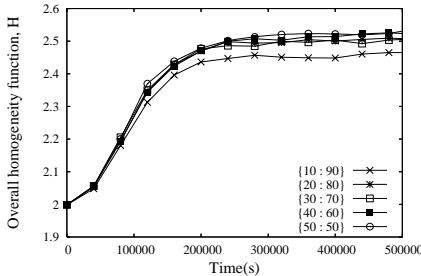


Fig. 1. Overall homogeneity function vs. time, with different patterns of capacity distribution

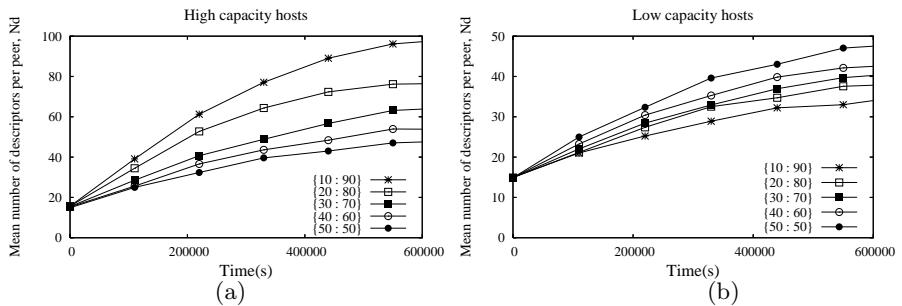


Fig. 2. Average number of descriptors that are stored in high (a) and low (b) capacity hosts, with different patterns of capacity distribution. The factors k_{pl} and k_{dl} of pick and drop probability functions are both set to 3.

makes the value of the spatial homogeneity function H increase from about $B/2$ to much higher values. After a transient phase, the value of H becomes stable: it means that the system reaches an equilibrium state despite the fact that peers go down and reconnect, agents die and others are generated, etcetera. In other words, the algorithm adapts to the varying conditions of the network and is robust with respect to them. Figure 1 also shows that the trend of the homogeneity function is similar for all the tested patterns of capacity distribution. Therefore, the load distribution feature of the algorithm does not affect the accumulation and reorganization of descriptors, which of course is a positive outcome.

The effectiveness of the load distribution approach is confirmed by Figure 2, which shows the average number of descriptors stored in high and low capacity hosts, with different capacity distribution patterns. The factors k_{pl} and k_{dl} of the pick and drop probability functions (see Section 2) are both set to 3. Figure 2(a) shows that, as the percentage of high capacity hosts decreases, and consequently, the average capacity of such hosts increases (because half the system load is divided among a lower number of hosts), these hosts are actually assigned a larger number of descriptors. For example, the average number of descriptors stored by high capacity hosts, in steady conditions, is about 100 with the pattern {10 : 90}, whereas it is less than 60 with the pattern {40 : 60}. The opposite

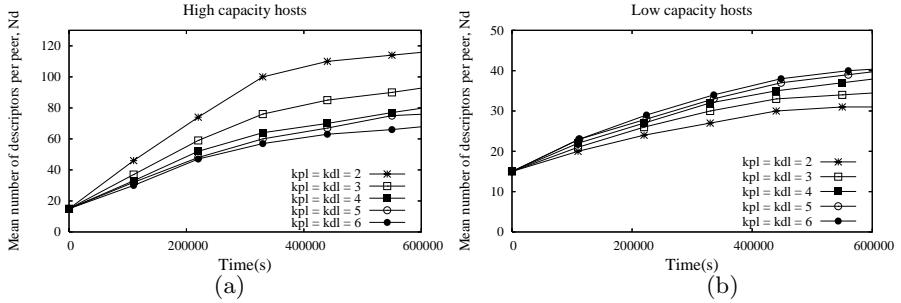


Fig. 3. Average number of descriptors that are stored in high (a) and low (b) capacity hosts, with different values of the factors k_{pl} and k_{dl} . The pattern of capacity distribution is set to {10 : 90}.

effect is observed for low capacity hosts, as can be observed in Figure 2(b). Note also that the trend corresponding to the pattern {50 : 50} is comparable in the two figures, since high and low capacity hosts coincide in this case. Therefore, the objective of assigning more descriptors to hosts that have better storage capabilities, and at the same time of alleviating the load of ordinary hosts, is successfully achieved.

We also calculated the variance and the coefficient of variation CV of the number of descriptors stored by the high and low capacity hosts, to verify how the load is distributed among the hosts of the same class. We found that the value of CV ranges from about 0.73 to about 0.82 for low capacity hosts and from 0.82 to 0.98 for high capacity hosts. These results reveal that the distribution of load within a class of hosts is not highly affected by the pattern of capacity distribution. Moreover, the value of CV decreases as the number of hosts of the class under consideration increases: therefore, the highest values of CV are obtained with pattern {50 : 50} for high capacity hosts and with pattern {10 : 90} for low capacity hosts.

It is also possible to regulate the fraction of load assigned to high and low capacity hosts by tuning the values of the factors k_{pl} and k_{dl} . To analyze this point, a set of experiments were made with the distribution pattern {10 : 90} and different values of those factors. Figure 3 shows that the number of descriptors stored in high (low) capacity hosts is inversely (directly) proportional to the value of the factors k_{pl} and k_{dl} . For example, the average number of descriptors stored in high capacity hosts, in steady conditions, is almost 120 if the factors are set to 2, while it decreases to much lower values for larger values of k_{pl} and k_{dl} . Therefore these factors can be used to balance the load among high and low capacity hosts, according to network and host requirements.

4 Conclusions

In this paper we introduced and evaluated an ant-inspired algorithm for building a P2P information system of a Grid. Grid resources are described by metadata

documents, or “descriptors”, which are indexed by binary keys. Ant-inspired mobile agents exploit probability functions to replicate descriptors, pick them from some hosts and drop them into other hosts. The objective is to reorganize descriptors and spatially sort them on the network. Moreover, descriptors are distributed by agents respecting the different capabilities of Grid hosts. Simulation analysis confirmed the effectiveness of the algorithm both in the spatially sorting of descriptors and in the achievement of a fair distribution of load among hosts having high and low storage capabilities. Indeed, hosts with higher storage capacity are assigned more descriptors than low capacity hosts. Currently, we are designing a discovery algorithm that exploits the characteristics of the obtained information system. According to this algorithm, query messages may be driven to hosts that have a large number of useful descriptors. Preliminary results are confirming that the performance of discovery operations is indeed improved thanks to the relocation and reorganization of information performed by mobile agents.

References

1. Foster, I., Kesselman, C.: *The Grid 2: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann Publishers Inc., San Francisco (2003)
2. Cai, M., Frank, M., Chen, J., Szekely, P.: Maan: A multi-attribute addressable network for grid information services. In: Proc. of GRID 2003, 4th International Workshop on Grid Computing, Washington, DC, USA (2003)
3. Bonabeau, E., Dorigo, M., Theraulaz, G.: *Swarm intelligence: from natural to artificial systems*. Oxford University Press, New York (1999)
4. Forestiero, A., Mastroianni, C., Spezzano, G.: So-Grid: A self-organizing grid featuring bio-inspired algorithms. *ACM Transactions on Autonomous and Adaptive Systems* 3(2) (2008)
5. Forestiero, A., Mastroianni, C., Spezzano, G.: Antares: an ant-inspired P2P information system for a self-structured grid. In: Proc. of Bionetics 2007, 2nd International Conference on Bio-Inspired Models of Network, Information, and Computing Systems, Budapest, Hungary (2007)
6. Lumer, E.D., Faieta, B.: Diversity and adaptation in populations of clustering ants. In: Proc. of SAB 1994, 3rd International Conference on Simulation of Adaptive Behavior: from animals to animats, Cambridge, MA, USA (1994)
7. Barabási, A.L., Albert, R.: Emergence of scaling in random networks. *Science* 286(5439) (1999)

Emergent Sorting in Networks of Router Agents*

Alexander Scheidler¹, Christian Blum²,
Daniel Merkle^{1,3}, and Martin Middendorf¹

¹ Department of Computer Science, University of Leipzig, Leipzig, Germany
`{scheidler,middendorf}@informatik.uni-leipzig.de`

² ALBCOM, Dept. Llenguatges i Sistemes Informàtics
Universitat Politècnica de Catalunya, Barcelona, Spain
`cblum@lsi.upc.edu`

³ Department of Mathematics and Computer Science
University of Southern Denmark, Odense, Denmark
`daniel@imada.sdu.dk`

Abstract. In this paper we study basic properties of so called Emergent Sorting Networks. These are directed networks which consist of router agents and buffer sites that are located between every two neighbouring agents. The agents can move objects from their input buffer sites to their output buffers. It is assumed that the objects may differ in type and are inserted randomly at an input agent. Brueckner (2000) presented a set of simple local policies for the agents which lead to a sorted outflow (batches of objects of the same type) of the objects out of an output agent. In this paper we introduce a pheromone based variant of the routing policies. The different policies are studied on quadratic and linear network topologies in terms of sorting performance and fairness.

1 Introduction

The following description of an industrial problem, arising in various industrial settings, was given in [1]:

Given a segment of a transport system of arbitrary layout in discrete high-volume production [environments] composed of unidirectional line-buffers (e.g., conveyors) and multi-input multi-output sequential routing devices (e.g., rotation tables, lifts), and assuming that the workpieces sent through the segment are all of one product but may be differentiated on the basis of the value of one product parameter; how may the segment

* This work was supported by German Research Foundation (DFG) through the project “Organisation and Control of Self-Organising Systems in Technical Compound” within SPP 1183, by the Integrated Action grant MEC HA2006-0127 (Germany/Spain), by grant TIN2007-66523 (FORMALISM) of the Spanish government, and by the EU project FRONTS (FP7-ICT-2007-1) funded by the European Comission under the FET Proactive Initiative Pervasive Adaptation. In addition, Christian Blum acknowledges support from the Ramón y Cajal program of the Spanish Ministry of Science and Technology of which he is a research fellow.

be controlled in a decentralized manner so that the outflow of workpieces occurs in batches of workpieces of the same product parameter value with the average batch size of the outflow being significantly higher than that of the inflow?

The author of [1] states that the solution to such a sorting problem requires a new approach to control, based on self-organization rather than on a central controller. This is because the problem is highly dynamic: at any time new workpieces may enter the system while others leave it. Moreover, system parameters such as the volume of the inflow may vary strongly over time. It may not even be known how many different product variants have to be handled at a time.

Social insect colonies are an example of distributed systems of simple agents that work without a central control (see, for example, [2,3]). The author of [1] reports on the following distributed solution to the batching problem stated above that was developed taking inspiration from the interaction of individuals in insect colonies: to each sequential routing device is assigned a so-called router agent. These agents act autonomously from other agents. An action of a router agent is to take a workpiece from an entry of its router to one of its exits. In its memory the agent stores for each exit the value of the product parameter of the last workpiece that has passed this exit. Having available multiple entries and multiple exits a router agent must decide on which workpiece to move to which exit. These decisions are taken by a set of simple rules that reportedly result in a batching (sorting) behaviour of the system.

Existing work. To the best of our knowledge, the above mentioned system of router agents for emergent sorting has never been studied in great detail. They were first mentioned in the context of the ESPRIT LTR project MASCADA. In the PhD thesis of Sven Brückner [1] they served for motivating the work carried out in the context of the thesis. And in a poster paper presented in the proceedings of GECCO 2006 [4] some limited experiments were presented.

Our contribution. First our goal was to study the effect of different network structures. The original proposal was limited to networks with square shape. Here we also study simpler networks that are composed of router agents organized in a line. Second, we introduce and study an agent routing policy that is based on pheromones, as used for example by ant colonies while foraging (see [5]). We try to deepen the understanding of emergent sorting networks by means of extensive experiments based on different routing policies, different network layouts, different number of object types, and different number of agents.

2 Sorting Networks of Router Agents

The basic components of sorting networks are *router agents*. Each router agent $a \in \mathcal{A}$ has an input and an output buffer with n , respectively m , positions. We denote the input buffer positions by x_1, \dots, x_n , and the output buffer positions by y_1, \dots, y_m . A buffer position can store exactly one object of k different types t_1, \dots, t_k . In the rest of this paper we use different colors to represent different

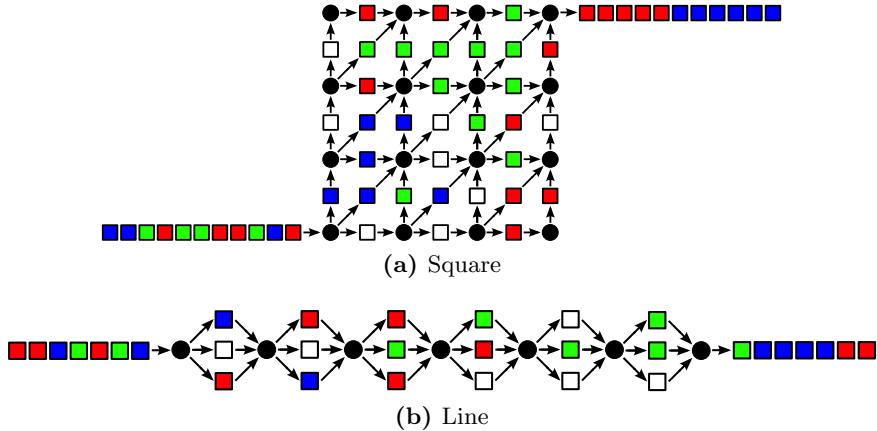


Fig. 1. Sorting networks of different shapes

object types. Router agents are connected to form a network topology by associating output buffer positions with input buffer positions of other agents. This is done with respect to the following conditions: (1) All input/output buffer positions must be involved in connections. (2) A one-to-one relationship between associated output and input buffer positions must hold. (3) The connections must be such that the resulting network is acyclic. Note that the set \mathcal{A} of agents contains two special agents: the so called *inflow agent* serves to feed the network with incoming objects and the *outflow agent* produces the output of the network.

Router agents can pick up objects from their input buffer positions and move them to a free output buffer position (if any). At each time step, first the input position of the inflow agent is filled with a randomly generated object. Then, in random order all router agents perform their local routing policy. Within a time step every object can be moved at most once. The system starts empty.

In Fig. 1a the network structure that was originally proposed in [1] is shown. The buffer positions are depicted as squares. White buffer positions are free, and a colored buffer position indicates the type of object it stores. Agents are shown as black circles. At the agent on the bottom left is the inflow agent and on the top right is the outflow agent. The input/output sequences of objects are shown as sequences of squares, in the same style as buffer positions.

3 Agent Behaviors

In the original system [1] every agent memorizes for each of its output buffer positions the type of the last object that was moved to this position (if any). How this memory affects the agent is given in Alg. 1.

Our work now considers three different aspects. First, to study if networks need to be square-shaped for exhibiting a sorting behaviour, we consider networks that are simply composed of a line of agents, as shown in Fig. 1b. Second, we explore a variation of the agent behaviour by replacing the orginal behaviour

Algorithm 1. Original Behavior of an Agent $a \in \mathcal{A}$

-
- 1: **if** exists an input buffer position x_i that stores an (within this time step) unmoved object o of type t , and a free output buffer position y_j for which the agent has memorized the type t **then**
 - 2: Pick up o from x_i and move it to y_j .
 - 3: **else**
 - 4: Let r be the number of unmoved objects in input buffer positions of a
 - 5: With probability r/n choose an unmoved object from one of the input buffer positions and move it to one of the free output buffer positions.
 - 6: **end if**
-

Algorithm 2. Pheromone-Based Behavior of an Agent $a \in \mathcal{A}$

-
- 1: T is the set of objects in input buffer of a that are unmoved in this timestep
 - 2: Choose a random number $p \in [0, 1]$
 - 3: **if** $p < |T|/n$ **then**
 - 4: **if** exists at least one free output buffer position **then**
 - 5: Choose a type $t_s \in T$ according to the following probability distribution:
$$\mathbf{p}(t_i) = \frac{\tau_i^a}{\sum_{t_l \in T} \tau_l^a} \quad \forall t_i \in T$$
 - 6: Move an object o with color t_s to random free output buffer position
 - 7: Update pheromon values
 - 8: **end if**
 - 9: **end if**
-

of an agent as explained in Alg. 1. For each agent $a \in A$ and for each object type t_i ($i = 1, \dots, k$) we introduce a pheromone value $0 \leq \tau_i^a \leq 1$. All pheromone values are initially set to $1/k$. The pheromone-based agent behaviour is shown in Alg. 2. After an agent $a \in A$ has moved an object of type t_s from an input position to one of its output positions (see lines 7 and 8 of Alg. 2), the pheromone values of agent a are updated as $\tau_j^a := \tau_j^a + \beta(\mu_j - \tau_j^a)$, $j = 1, \dots, k$ where $\mu_j = 1$ in case $j = s$, and $\mu_i = 0$ otherwise. For the learning rate β an appropriate value must be found.

We also test a variation of the two agent behaviours described in Algs. 1 and 2 concerning the so called waiting behavior, by removing in Algs. 1 line 6 and in Alg. 2 line 3 and replacing them by "if no input buffer position is empty **then**". In words, an agent is only allowed to act if all its input buffer positions are occupied by an object.

4 Experimental Evaluation

All results of this work were obtained by simulation. For the remainder of this paper, we use the following notation for the different options outlined before. The notation XYZ consists of three letters, where:

- $X \in \{\mathbf{B}, \mathbf{P}\}$. Hereby, \mathbf{B} denotes the original agent behaviour (Alg. 1), and \mathbf{P} denotes the pheromone-based agent behaviour (Alg. 2).
- $Y \in \{\mathbf{o}, \mathbf{n}\}$. Letter \mathbf{o} refers to the old waiting behavior (the probability to act is proportional to the number of occupied input buffer positions), whereas \mathbf{n} corresponds to the system using the new waiting behavior (the agent only acts when all its input buffer positions are occupied).
- $Z \in \{\mathbf{s}, \mathbf{l}\}$. These identifiers refer to the network structure. Letter \mathbf{s} indicates a square-shaped network, and letter \mathbf{l} refers to a network in shape of a line.

4.1 Measures of System Performance

Most of our measurements concern the following sorting measure computed with respect to the output sequence of the system. When the system is stopped after a predefined number of time steps, we compute the probability of a color change in the output sequence. This measure will henceforth be denoted by p_c . In general the simulation runs for different parameters are done over 50.000 time steps.

Another measure of interest is the number of time steps that an object on average spends in the system. In addition to the average time that a workpiece spends in the system, we will also look at the maximum time and at the standard deviation to investigate how fair the system is. We also took a look at the throughput, by measuring how many objects leave the systems when simulating 1.000.000 time steps (simulations were always using 100 agents).

4.2 Tuning

As mentioned previously, the pheromone-based agent behaviour requires an appropriate setting of the parameter $\beta \in [0, 1]$ which corresponds, in some sense, to the pheromone evaporation parameter of the ACO metaheuristic (see [6]). For all the experiments described in the following we used the following options: $\beta \in \{0, 0.05, 0.1, \dots, 1.0\}$. The tuning results are used in the rest of the paper to choose a good parameter β for every different pheromone based agent behavior.

The graphics of Fig. 2 present tuning results (in terms of measure p_c , see Sect. 4.1) of the pheromone based systems. Each graphic contains four performance curves, corresponding to four different network sizes: 16, 64, 144, and 256 agents. Following conclusions can be made: First, in the square-shaped networks an agent should always try to repeat the action of the previous time step ($\beta = 1$). Second, in the line-shaped networks the more agents used, the smaller should be the value of β . Third, the optimal value for β also depends on the different waiting rules and also slightly on the number of colors (results not shown).

4.3 Results

Figure 3 presents the measure p_c (that is, the probability of a color change) for the original system as well as the variants that we proposed. Results are shown for different numbers of colors $\{3, 5\}$ and for different numbers of agents.

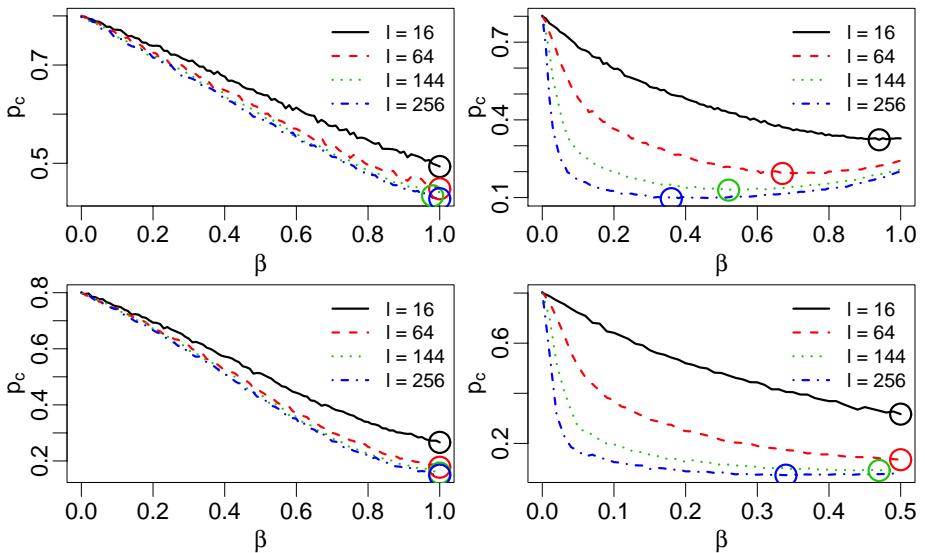


Fig. 2. Tuning results for systems \mathbf{Po}^* (top) and \mathbf{Pn}^* (bottom) for the square (left) and the line topology (right); 5 colors used; Circles indicate the best value for β

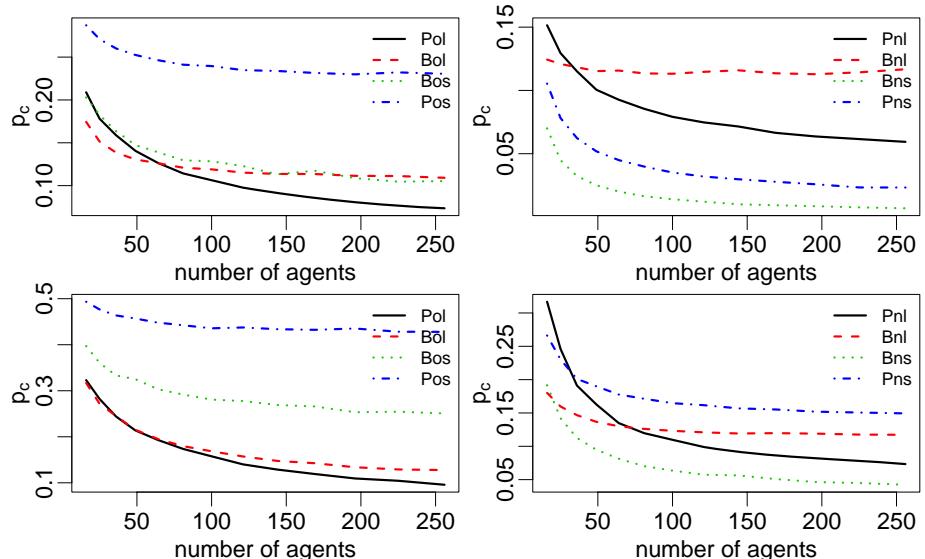


Fig. 3. Performance p_c over the number of agents for systems $\mathbf{*o}^*$ (left) and systems $\mathbf{*n}^*$ (right) for 3 (top) and 5 (bottom) colors

Concerning the original agent behaviour (systems $\mathbf{*o}^*$), one can observe that the sorting behaviour of the line-shaped networks is in general much better than the sorting behaviour of the square-shaped networks. This trend becomes

stronger when the number of colors grows. Second, the pheromone-based system greatly improves over the original system when line-shaped networks with many agents are concerned. But opposite is the case for square-shaped networks.

Interestingly, the results concerning the new waiting rule of the agent behaviours (systems ***n***) look quite different. Here the original system (**Bns**) in conjunction with a square-shaped network, works best. When 3 colors are concerned, the sorting behaviour of the square-shaped networks outperform the sorting behaviour of the line-shaped networks. However, when the number of colors grows, the performance of the pheromone-based system on a square-shaped network drops but the performance of the line-shaped networks improves.

In order to study this effect more in detail we performed experiments concerning higher number of colors. In the outcome the three systems **Bns**, **Bnl**, and **Pnl** had the best performance. The results are shown in form of a 3D-plot in Fig. 4(left). An interesting effect can be observed here. First the results show that for low number of colors the system **Bns** outperforms both line-based systems. But when there are more colors in the system suddenly the number of agents becomes important, since with few agents system **Bnl** is best, whereas with many agents the system **Pnl** performs best.

When comparing the performance of the systems with the original waiting behavior (***o***) to the performance of the changed systems (***n***), there is a clear advantage of the modified systems. The best-performing changed system (**Bns**) performs always better than the best-performing original system (**Pol**).

In the second column of Table 1 we provide the average time an object spends in the different systems. The third column gives the standard deviation of these times and the fourth column provides the maximum time an object spent in the system. In general, the pheromone-based systems are characterized by a longer "average time in system" but the "maximum time in system" is greatly reduced as compared to the original systems. The line-shaped networks show a short "maximum time in system" and are more fair since the times the different objects stay in the system has a lower variance. In addition, we can observe that the changed waiting behavior (systems ***n***) leads to an increase in both average and maximum time that an object stays in the system.

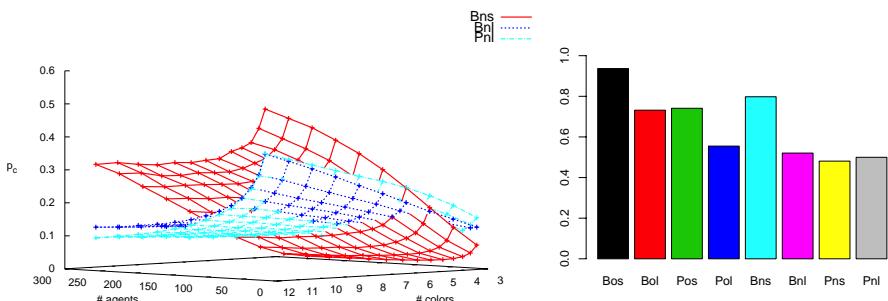


Fig. 4. Performance results concerning higher numbers of colors (left); Throughput of the different systems (right)

Table 1. Results concerning the time objects stay in the system

System	Av	SD	Max	System	Av	SD	Max
Bos	257.5	354.7	16194	Bns	302.9	301.1	11155
Bol	241.7	16.10	341	Bnl	433.8	19.20	587
Pos	335.8	203.9	5600	Pns	527.6	310.2	6158
Pol	381.1	13.30	482	Pnl	496.0	11.20	600

In Fig. 4(right) the throughput measurement is given. It can be seen that in systems using the unmodified waiting behavior more objects leave the system, because the agents can act more often and do not need to wait. Pheromone based systems have a lower throughput than the original systems, because here the waiting behavior is always applied, whereas in the original system the agents will always act if there is a "good" move possible (e.g. line 1 in Alg.1).

5 Conclusions

In this paper we presented a study of emergent sorting effects exhibited by a certain type of networks of router agents. In addition to the original proposal of such networks, we examined variants and extensions, including a pheromone-based agent behaviour. The experimental results show that the sorting performance strongly depends on the shape and the size of the network, the number different object types, and the agent behaviour.

References

1. Brueckner, S.A.: Return From the Ant—Synthetic Ecosystems for Manufacturing Control. PhD thesis, Humboldt University, Berlin (2000)
2. Bonabeau, E., Dorigo, M., Theraulaz, G.: Swarm Intelligence: From Natural to Artificial Systems. Oxford University Press, Oxford (1999)
3. Blum, C., Merkle, D. (eds.): Swarm Intelligence – Introduction and Applications. Natural Computing. Springer, Berlin (2008)
4. Tozier, W.A., Chesher, M.R., Devgan, T.S.: The brueckner network: An immobile sorting swarm. In: Cattolico, M., et al. (eds.) Proceedings of GECCO 2006 – Genetic and Evolutionary Computation Conference, pp. 91–92. ACM Press, New York (2006)
5. Wyatt, T.D.: Pheromones and Animal Behaviour: Communication by Smell and Taste. Cambridge University Press, Cambridge (2003)
6. Dorigo, M., Stützle, T.: Ant Colony Optimization. MIT Press, Cambridge (2004)

Enhancing the Cooperative Transport of Multiple Objects

Antoine Decugnière¹, Benjamin Poulain¹, Alexandre Campo¹, Carlo Pinciroli¹, Bruno Tartini², Michel Osée², Marco Dorigo¹, and Mauro Birattari¹

¹ IRIDIA, CoDE, Université Libre de Bruxelles, Brussels, Belgium

{adecugni,bpoulain,acampo,cpinciro,mdorigo,mbiro}@ulb.ac.be

² BEAMS, Université Libre de Bruxelles, Brussels, Belgium

{btartini,mosee}@ulb.ac.be

Abstract. In this paper we present an approach to the cooperative transport of multiple objects in swarm robotics. The approach is motivated by the observation that the performance of cooperative transport in insect colonies as well as in groups of robots grows in a super linear way with the number of individuals participating in the transport. The transport relies on a cart in which multiple objects are collected and stored before being moved to destination. The cart is carried by a group of robot that would be otherwise allocated to the transport of single objects. The cart is endowed with computational and communication abilities that allow it to cooperate with the transporting robots. This research is carried out within the framework of the Swarmanoid project and aims at enhancing the transport capabilities of the robot swarms developed in this project.

1 Introduction

Cooperative transport is a classical problem studied in the collective robotics literature [1,2,3,4,5]. Robots have to perform cooperative transport when a task requires the transport of an object that a single robot is not able to handle and move. Approaches to solve this problem have typically taken inspiration from biological systems, especially from colonies of social insects [6,7,8].

In particular, ant colonies display two different transport behaviours: solitary transport and group transport. In some ant species, group transport is observed when a prey is larger or heavier than the transporting capability of a single ant. In this case, ants have the option either to cut the prey in pieces and to transport each piece individually, or to cooperate to transport the whole prey as a group. Studies show that transporting the prey as a whole in group is usually more efficient than the solitary transport of pieces [9,10].

Kube and Zhang developed a model of cooperative transport inspired by the behaviour of social insects [7]. This model was then demonstrated by Kube and Bonabeau with experiments involving a group of real robots [11]. Additional experiments have shown that transport performance grows in a super linear way with the number of robots, which is similar to what is observed with ants [12,13].

In this paper, we present an approach to enhance the ability of a group of robots to transport objects cooperatively. The objects to be transported are gathered in a cart and are then transported collectively as a single entity. The research presented in this paper is carried out within the framework of the *Swarmanoid* project [14], a Future and Emerging Technologies (FET-OPEN) project funded by the European Commission. The goal of the project is to develop a *swarmanoid*, that is, a swarm-based alternative to a humanoid robot. In a *swarmanoid*, the functionalities that one typically expects in a humanoid robot are distributed among the individuals of an heterogeneous swarm of robots. The *swarmanoid* is composed of (i) a number of *eye-bots*, flying robots that are able to explore a scene and to localize objects; (ii) a number of *hand-bots*, manipulators that are able to climb and to grasp object; and (iii) a number of *foot-bots*, robot rovers that are able to move on the ground and to transport objects or other robots. In the spirit of the *Swarmanoid* project, we design a fourth kind of robot: the *cart-bot*. A *cart-bot* is able to store a number of objects, to cooperate with hand-bots in order to facilitate the loading and unloading of objects, and to cooperate with foot-bots to ease its own transport.

The rest of the paper is organized as follows: in Section 2 we detail the integration of the *cart-bot* in the *Swarmanoid* project. In Section 3 we describe the final design of the *cart-bot* hardware. Finally, in Section 4 we conclude the paper.

2 Integration of the *Cart-Bot* in the Swarmanoid Project

The *Swarmanoid* project is developed around a main scenario, which illustrates the use of morphologically specialised robots to solve a complex mission in a human-like environment. In the following, we describe how the *cart-bot* can be used to enhance transport tasks within the scope of this scenario.

In the *Swarmanoid* scenario, an heterogeneous group of robots is employed to locate books situated on shelves, collect them, and bring them to a target place. For practical reasons, rather than with real books, robots deal instead with models of books that we call *book-bots*. A *book-bot* is made of foam so that it causes less damages if it inadvertently falls on the ground or on a robot; moreover, it features a number of LEDs on the spine to facilitate its localisation.

A typical unfolding of the scenario is as follows: first the *eye-bots* explore the environment. Once they have located a number of *book-bots*, they guide other robots to them. The *hand-bots* that have extended manipulation capabilities are transported by the *foot-bots* close to these *book-bots*. The *hand-bots* get a hold of the *book-bots* one by one. In a scenario without the *cart-bot*, each *book-bot* is transported by a system composed of a *hand-bot* holding a *book-bot* and a number of *foot-bots* carrying the *hand-bot*.

With the *cart-bot* included in the scenario, once a *hand-bot* grasps a *book-bot*, it lays it on the depository area of the *cart-bot*, as shown in Figure 1(a). The *cart-bot* automatically swallows the *book-bot* to store it inside his rack and the *hand-bot* is therefore free to pursue the collection of another *book-bot*. As soon as the *cart-bot* is full or has loaded the required number of *book-bots*, it advertises to surrounding *foot-bots* that it is ready to be transported. Once

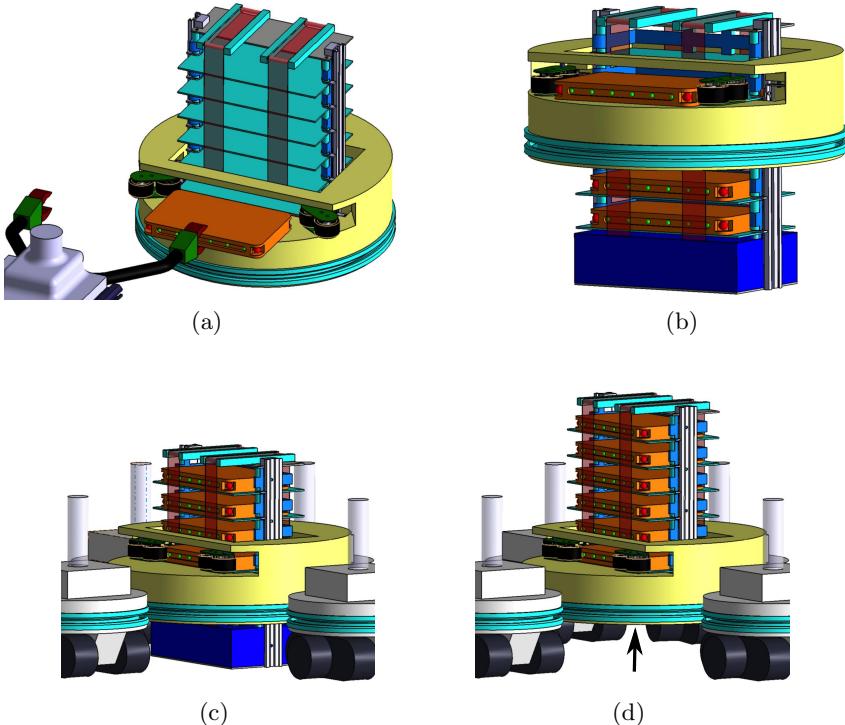


Fig. 1. (a) *Book-bot* put on the loading bay by a *hand-bot*. (b) Base elevation on the rack to reach a high slot. (c) *Foot-bots* docked to the *cart-bot*. (d) Rack elevated to remove friction with the ground.

foot-bots are physically connected to the *cart-bot*, it lifts up the part of its body that was previously in contact with the floor— see Figures 1(c) and 1(d). By doing this, the *cart-bot* prevents any friction with the ground, therefore facilitating its transport. Eventually, the *cart-bot* communicates with the connected *foot-bots* to let them know that transport can be performed at any time.

The integration of the *cart-bot* in *Swarmanoid* allows to study stacked transport against single transport of multiple objects. Furthermore, contrary to the *hand-bot*, the *cart-bot* is specialized for transport: it is able to store a number of *book-bots*, it has no fragile external components like manipulators do, *book-bots* are stabilized inside the *cart-bot* and may not fall on the floor. Lastly, the *cart-bot* is large and *foot-bots* can connect to it all around such that the resulting assembly is very stable.

3 Hardware Design

The two main functionalities of the *cart-bot* are storing *book-bots* in a single rack and allowing the single cooperative transport of all the gathered *book-bots* by the *foot-bots*. The design of these functionalities is described in the following.

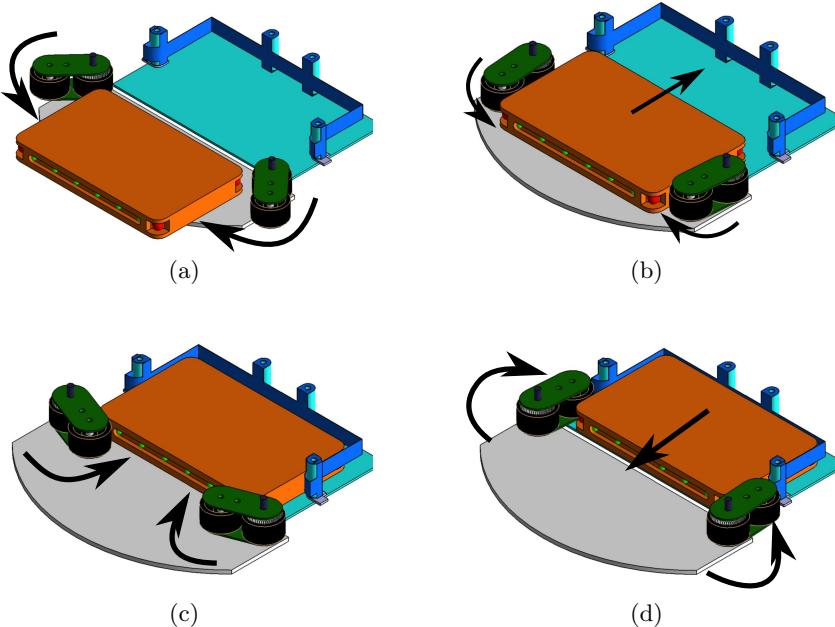


Fig. 2. (a) The arms rotating towards a *book-bot* on the loading bay. (b) The wheels pushing the *book-bot* inside a slot. (c) The arms finishing to push the *book-bot* in the slot. (d) The arms reversed to get a *book-bot* from a slot.

3.1 Storing Ability

In order to store the *book-bots*, the *cart-bot* introduces them inside the slots of a storing system called the *rack*. This can be done in two different ways: either by having the *hand-bot* inserting a *book-bot* directly in a selected slot, or by having the *hand-bot* laying a *book-bot* down on a specific bay and then moving the *book-bot* to the appropriate slot through some loading and unloading mechanisms. The second approach turns out to be simpler to implement and requires less accuracy in the positioning of the *book-bot* by the *hand-bots*.

In the design we developed, the loading and the unloading functions are carried out by a single mechanism. This loading/unloading mechanism consists of two rotating arms with wheels at their ends. The arms are located on both sides of the loading bay, right in front of the slots as it can be seen in Figures 2(a), 2(b), 2(c) and 2(d). By rotating in one direction or in the opposite, these arms can reach a *book-bot* either when the *book-bot* is on the loading bay—as shown in Figure 2(a)—or when the *book-bot* is inside the slot—as shown in Figure 2(d). In particular, Figure 2(b) shows how the arms, pressed against a *book-bot*, push the *book-bot* itself inside the slot. The wheels turn and push the *book-bot* in the same direction. This always ensures a tight contact and a good grip with the *book-bot*. Once the *book-bot* has been swallowed by the wheels, the arms complete their rotation and push the *book-bot* inside the slot to their final storage position—as shown in Figure 2(c).

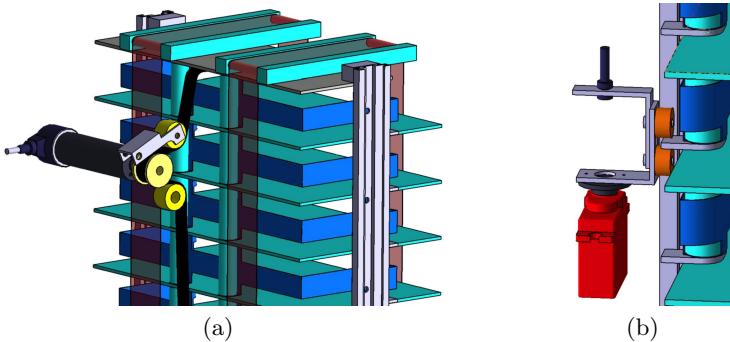


Fig. 3. (a) Belt transmission and linear guides. (b) Brake mechanism on a slot.

During the loading phase, the trajectory followed by the *book-bot* is determined by the angular position of the arms. In particular, in order to insure that the *book-bot* enters the slot with a correct angle, the rotating arms must assume a symmetric angular position. During the whole procedure, the speed and the direction of the motors are adapted in order to correct the trajectory of the *book-bot*. If an incoming *book-bot* gets stuck, the failure is detected by a current peak on the blocked motor or by the incomplete angular positioning of the rotating arms. The motors are reversed and the whole procedure is repeated.

The slots where the *book-bots* are stored are engineered in a way that they form a vertical rack of horizontal slots—see Figure 1(a). The storing system manages the slots and takes care to align the loading bay and the loading/unloading system with the slot in which the *book-bot* has to be inserted. In order to achieve correct alignment, the rack can be moved relatively to the base platform or the base platform can be moved relatively to the rack, depending on whether the *cart-bot* is being held by *foot-bots* or it lies on the ground—see Figure 1(b). This allows the system to lift up the rack and ease transport by preventing any friction with the ground, as described in Section 3.2.

The structure of this system is very similar to an elevator architecture and can be controlled through a classical elevator-like regulation. It uses linear guides and a timing (toothed) belt—see Figure 3(a). The belt is driven by a pulley fixed directly on the shaft of a motor with an encoder. The belt transmission was preferred to a screw drive transmission because of weight and efficiency concerns, but it has the drawback of not being auto-blocking. This led us to implement a braking system that insures precise alignment of the slot to the loading bay. The braking system is designed to be finely adjustable in height and is based on simple and lightweight components—see Figure 3(b).

The presence of a *book-bot* in a slot is perceived as the success of the loading and is memorized by the storing system. To ensure a reliable perception of the status of the slots, we introduce redundancy with optical sensors. This information is critical to know where are the free slots and when the *cart-bot* is ready to be transported.

3.2 Transportability

To display its status, for example when it is ready to be transported, the *cart-bot* uses a ring of RGB LEDs. This color ring can also be used coordinate the *foot-bots* by indicating them in what position they should dock. Docking is achieved by the *foot-bots* by gripping a docking ring on the *cart-bot*. This docking ring is positioned on the lowest part of the base platform. The *cart-bot* uses the mechanism of the storing system to elevate the ring up to the height of the gripper of the *foot-bots*. This prevents the ring to be in the way of a *book-bot* brought horizontally by a *hand-bot* to the loading bay when the base platform is lowered—see Figure 1(a).

During all these manipulations the rack of slots is resting on the floor. It presents high resistance to the *foot-bots* trying to push or pull it. Docked *foot-bots* may rely on this resistance to decide to recruit more *foot-bots*, as exposed in [15]. Once enough *foot-bots* are docked, the rack is lifted up so that it doesn't touch the floor anymore and the friction to the ground is canceled—see Figure 1(c). Thanks to this mechanism, the *cart-bot* has a good ground clearance when transported. The system formed by the *cart-bot* and the *foot-bots* is very stable with a large base and has a good mobility.

4 Conclusions

In the paper we have discussed the problem of the cooperative transport of multiple objects by a swarm of robots. In the approach we adopted, rather than transporting objects one by one, these are gathered in a cart which is subsequently transported by the swarm. With this approach, the transport performance grows super linearly with the size of the swarm, with the classic limitations induced by the problem of coordinating the movement of the swarm [12]. In the paper, we have analyzed the problem of transporting multiple objects within the framework of the *Swarmanoid* project. In particular, we have described the features and the design of the *cart-bot*, a robot which is able to store objects, the *book-bots*, and ease the task of other robots, such as the *foot-bots* that are intended to carry out transport.

The *cart-bot* enhances the robustness provided by the distributed hardware and control [16] in the transport of *book-bots*. The *cart-bot* secures the stored *book-bots* to avoid losing them during the transport due to collisions or to the roughness of the terrain. The large round shape of the *cart-bot* allows the *foot-bots* to dock all around it: this results in a very stable assembly. Moreover, the transporting assembly is not much sensitive to terrain condition, thanks to a good ground clearance obtained through the elevation of the rack of the *cart-bot* after the docking with the *foot-bots*.

To ensure a robust transport, the *cart-bot* is designed to be simple and reliable. During transport, the *cart-bot* can be seen as a single resistant entity, as it has no fragile external parts. The use of the same mechanisms for different functionalities minimizes the complexity and reduces the global weight of the *cart-bot*: rotating arms are used for loading and unloading and an elevation

system is used for reaching different slots of the rack when loading and unloading and for elevating the rack itself during the transport. Finally, the *cart-bot* is mainly built with generic electronic and mechanical components, which makes the *cart-bot* easy to maintain.

Acknowledgments. The authors thank Francesco Mondada and Michael Bonani for the useful discussions and the valuable advice. The research described in the paper was carried out in the framework of *Swarmanoid*, a project funded by the Future and Emerging Technologies programme (IST-FET) of the European Commission under grant IST-022888. The work was partially supported by the project ANTS, an *Action de Recherche Concertée* funded by the Scientific Research Directorate of Belgium's French Community. Alexandre Campo, Marco Dorigo, and Mauro Birattari acknowledge support from the fund for scientific research F.R.S.-FNRS of Belgium's French Community.

References

1. Matarić, M., Nilsson, M., Simsarian, K.: Cooperative multi-robot box-pushing. In: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, Pittsburg, PA, pp. 556–561. IEEE Computer Society Press, Los Alamitos (1995)
2. Pereira, G.A.S., Kumar, V., Spletzer, J., Taylor, C.J., Campos, M.F.M.: Cooperative transport of planar objects by multiple mobile robots using object closure. In: Proceedings of the 8th International Symposium on Experimental Robotics, ISER 2002, Sant'Angelo d'Ischia, Italy (2002)
3. Wang, Z., Kumar, V.: Object closure and manipulation by multiple cooperative mobile robots. In: Proceedings of the IEEE International Conference on Robotics and Automation, ICRA2002, Washington, DC, pp. 394–399. IEEE Computer Society Press, Los Alamitos (2002)
4. Groß, R., Dorigo, M.: Cooperative transport of objects of different shapes and sizes. In: Dorigo, M., Birattari, M., Blum, C., Gambardella, L.M., Mondada, F., Stützle, T. (eds.) ANTS 2004. LNCS, vol. 3172, pp. 107–118. Springer, Heidelberg (2004)
5. Campo, A., Nouyan, S., Birattari, M., Groß, R., Dorigo, M.: Negotiation of goal direction for cooperative transport. In: Dorigo, M., Gambardella, L.M., Birattari, M., Martinoli, A., Poli, R., Stützle, T. (eds.) ANTS 2006. LNCS, vol. 4150, pp. 191–202. Springer, Heidelberg (2006)
6. Bonabeau, E., Dorigo, M., Theraulaz, G.: Swarm Intelligence: From Natural to Artificial Systems. Oxford University Press, New York (1999)
7. Kube, R.C., Zhang, H.: Collective robotics: from social insects to robots. *Adaptive Behaviour* 2(2), 189–218 (1994)
8. Martinoli, A., Mondada, F.: Collective and cooperative group behaviours: Biologically inspired experiments in robotics. In: Proceedings of the Fourth Symposium on Experimental Robotics, ISER-1995, Stanford, California, USA (June 1995)
9. Moffett, M.W.: Cooperative food transport by an asiatic ant. *National Geographic Research* 4, 386–394 (1988)
10. Traniello, J.F.A., Beshers, S.N.: Maximization of foraging efficiency and ressource defense by group retrieval in the ant *Formica schaufussi*. *Behavioral Ecology and Sociobiology* 29, 283–289 (1991)

11. Kube, C., Bonabeau, E.: Cooperative transport by ants and robots. *Robotics and Autonomous Systems* 30(1 - 2), 85–101 (2000)
12. Mondada, F., Bonani, M., Guignard, A., Magnenat, S., Studer, C., Floreano, D.: Superlinear physical performances in a swarm-bot. In: Capcarrère, M.S., Freitas, A.A., Bentley, P.J., Johnson, C.G., Timmis, J. (eds.) ECAL 2005. LNCS (LNAI), vol. 3630, pp. 282–291. Springer, Heidelberg (2005)
13. O’Grady, R., Groß, R., Christensen, A., Mondada, F.M., Bonani, M.D.: Performance benefits of self-assembly in a swarm-bot. In: Proceedings of the 2007 IEEE/RSJ Intl. Conference on Intelligent Robots and Systems, IROS 2007, San Diego, CA (2007)
14. Dorigo, M., Tuci, E., Groß, R., Trianni, V., Labella, T., Nouyan, S., Ampatzis, C., Deneubourg, J.L., Baldassarre, G., Nolfi, S., Mondada, F., Floreano, D., Gambardella, L.: The SWARM-BOTS project. In: Şahin, E., Spears, W.M. (eds.) Swarm Robotics 2004. LNCS, vol. 3342, pp. 31–44. Springer, Heidelberg (2005)
15. Groß, R.: Self-assembling robots. PhD thesis, Université Libre de Bruxelles, Brussels, Belgium (2007)
16. Mondada, F., Pettinaro, G.C., Guignard, A., Kwee, I., Floreano, D., Deneubourg, J.L., Nolfi, S., Gambardella, L., Dorigo, M.: Swarm-bot: a new distributed robotic concept. *Autonomous Robots* 17(2–3), 193–221 (2004)

Formal Modeling of *BeeAdHoc*: A Bio-inspired Mobile Ad Hoc Network Routing Protocol

Muhammad Saleem¹, Syed Ali Khayam², and Muddassar Farooq³

¹ Center for Advanced Studies in Engineering (CASE) Islamabad, Pakistan

² WisNeT, SEECS-NUST Rawalpindi, Pakistan

³ nexGIN RC, NUCES-FAST Islamabad, Pakistan

msaleem@case.edu.pk, khayam@niit.edu.pk,

muddassar.farooq@nu.edu.pk

Abstract. Design and development of routing protocols for Mobile Ad Hoc Networks (MANETs) is an active area of research. The standard practice among researchers working in this emerging domain is to evaluate the performance of their routing protocols in a network simulator. It is now a well known fact that the simulation studies are scenario specific and hence their results can not be generalized. In this paper, we present mathematical models of two key performance metrics, routing overhead and route optimality, of *BeeAdHoc* MANET routing protocol. One of the key components of our *BeeAdHoc* model is the collision model at Medium Access Control (MAC) layer. The mathematical expressions of the performance metrics provide valuable insight about the behavior of *BeeAdHoc* in particular, and a typical ad hoc routing protocol in general, without resorting to scenario specific time consuming simulations.

1 Introduction

Design of novel routing techniques for Mobile Ad Hoc Networks (MANETs) received a significant amount of attention by researchers that resulted in a number of Bio-inspired routing protocols: AntHocNet [1], ARA [2], *BeeAdHoc* [3,4] and ANSI [5]. As topology of ad hoc network is dynamic, such protocols need to adapt themselves to continuously changing routes between corresponding source and destination pairs. The frequency of route changes is dependent on node density and their deployment pattern, nodes speed etc. In order to understand the behavior of protocols over a wide operational spectrum, designers create a number of different scenarios by changing the values of the above-mentioned parameters and then collect relevant performance metrics through extensive simulations that might take several days or even weeks to finish.

The simulation-based study has three shortcomings: (1) the results are scenario-specific, (2) simulation tools have limited scalability, and (3) large simulation time significantly slows down the protocol engineering cycle. Therefore, we argue that mathematical tools must be utilized to complement the simulation studies in order to overcome their shortcomings. This is also the corollary of the paper by Kurkowski et al. [6].

Formal modeling of MANET routing protocols is difficult due to various reasons: dynamic topology, diverse flooding patterns, contention at the MAC layer etc. In this paper, we model two well known metrics, for *BeeAdHoc* protocol by incorporating the collisions at the MAC layer. The derived metrics not only provide an unbiased analysis of *BeeAdHoc* protocol but also unveils some interesting facts about its behavior. Derived metrics may be used to derive a number of other evaluation metrics: total energy consumption, computational complexity, packet latency etc.

Related Work. The formal modeling of Bio-inspired protocols have received little attention. The exceptions are the work of Roth and Wicker [7], Saleem et al. [8] and Zahid et al. [9]. The later is however limited to fixed networks. In [7], Roth has developed an analytical framework based on the Markov chains for the analysis of probabilistic routing protocols.

Organization of Paper. The rest of the paper is organized as follows. Section 2 contains system description and modeling assumptions. A brief description of *BeeAdHoc* protocol is presented in section 3. Section 4 describes the modeling of routing overhead followed by the route optimality model in Section 5. Finally we conclude our paper with an outlook to our future research.

2 System Description and Modeling Assumptions

2.1 Basic Graph Terminology

A typical graph is denoted by $G(V, E)$ in which V is a set of vertices in the graph and E represents the set of edges. This model can be used to represent an ad hoc network in which individual nodes are the vertices of the graph connected through wireless links (or edges of the graph). In this section, definitions of some basic graph-theoretic terms are provided.

Node degree: Degree of a node x , $d(x)$, represents the number of nodes directly connected with x . Minimum degree of a graph G is then defined as:

$$d_{min}(G) = \min \{d(x)\} \quad \forall x \in G$$

A similar term is the average node degree defined as:

$$d_{avg}(G) = \frac{1}{n} \sum_{x=1}^n d(x)$$

Connected and disconnected graphs: A graph is connected if at least a path exists between each pair of nodes in the graph [10]; otherwise, it is a disconnected graph.

2.2 Network Topology

Ad hoc network topology plays the pivotal role in modeling of an upper layer protocol. Connectivity of graph depends upon the nodes deployment pattern, their

transmission powers, environmental conditions etc. Bettstetter's seminal work [10] addresses the connectivity of wireless networks. Assuming that N nodes are randomly distributed and connected through symmetric links, Bettstetter derived an expression that, for a given node density ρ , determines the transmission range r_0 to ensure that a randomly chosen node will have exactly n_0 neighbors. Probability P that a node has exactly n_0 neighbors is given below:

$$P(d = n_0) = \frac{(\rho\pi r_0^2)^{n_0}}{n_0!} \cdot e^{-\rho\pi r_0^2}. \quad (1)$$

Expected or average degree of the node is $E(d) = d_{avg} = \rho\pi r_0^2 - 1$. To be sure with a certain probability p that the network having $n >> 1$ nodes is connected,

$$r_0 \geq \sqrt{\frac{-\ln(1 - p^{\frac{1}{n}})}{\rho\pi}}. \quad (2)$$

We can use (1) and (2) to generate the underlying network topology. Interested readers are referred to [10] for further information.

2.3 Modeling Assumptions

We assume a network in which N nodes are deployed according to a homogeneous Poisson distribution with node density ρ and an average node degree of d_{avg} . Each node has a transmission radius r_0 and an 802.11b distributed coordination function (DCF) acts as an underlying MAC layer protocol.

3 BeeAdHoc

BeeAdHoc is an on-demand multi-path routing algorithm for mobile ad hoc networks inspired from the foraging principles of honey bees [4]. *BeeAdHoc* works with four types of agents: *packers*, *scouts*, *foragers* and *swarms*. The *packers* locate a *forager* and hand over the data packet to the discovered *forager*. *Scouts* discover new routes from the launching node to the destination node through broadcasting principle and an expanding time to live (TTL) timer. *Foragers*, the main workers of *BeeAdHoc*, receive the data packets from the *packers* and transport them to the destination. Transportation of *foragers* back to the source node, in case of unreliable transport protocol, is the key role of *swarms*. Interested readers are referred to [4] for more details.

4 Routing Overhead Model

We define *routing overhead* as the number of *scouts* generated in the network up to a particular number of hops (h) during a route discovery process. Our definition of *routing overhead* reflects the time to live mechanism employed in *BeeAdHoc* during a route discovery. When TTL expires, nodes stop rebroadcasting the scouts. A typical propagation pattern of *scouts* in *BeeAdHoc* is shown

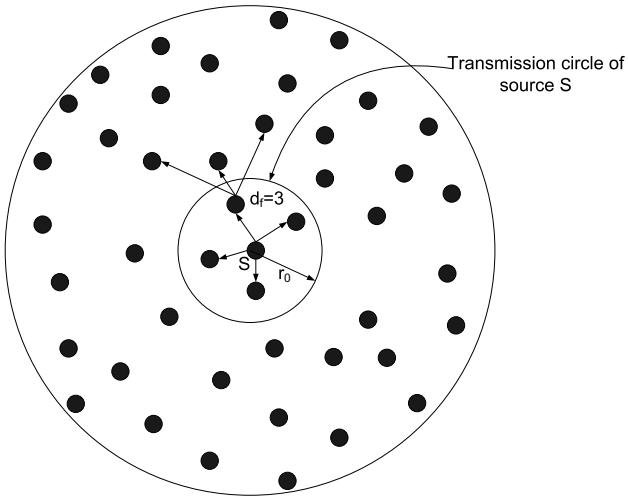


Fig. 1. Route discovery pattern in an ad hoc network

in Fig. 1 in which source S, located at the center, broadcasts a *scout* to all its neighbors (i.e. nodes located within its transmission circle). In the rest of this section, we derive an expression to calculate the expected number of *scouts* up to h hops from the source node.

4.1 Routing Overhead in Terms of Expected Forward Degree

The first *scout* broadcasted by the source is received by d_{avg} nodes, where d_{avg} is the average degree of the node. Each one of the d_{avg} neighbors rebroadcast the *scout* unconditionally. However, due to collisions, a fraction of the rebroadcasted scouts get lost and we do not count them in the calculation of routing overhead. Hence, the first hop rebroadcasting nodes equal $\overline{p_c}d_{avg}$ where $\overline{p_c}$ is the probability of no collision. To compute the routing overhead of *BeeAdHoc*, $C_p^{(beeadhoc)}$, we accumulate the *scouts* generated at each hop up to a distance of h hops from the source node.

$$C_p^{(beeadhoc)} = 1 + \overline{p_c}d_{avg} + (\overline{p_c})^2 d_{avg} \times d_{f[1]} + ((\overline{p_c})^3 \cdot d_{avg} \cdot d_{f[1]}) \times d_{f[2]} + \dots + ((\overline{p_c})^h \cdot d_{avg} \cdot d_{f[1]} \dots d_{f[h-2]}) \times d_{f[h-1]}. \quad (3)$$

Equation (3) contains a new term known as *expected forward degree* of a node. We define it as: "*the number of new nodes that will receive the scout of that node and are likely to rebroadcast it to the next hop*". The terms, $d_{f[1]}, d_{f[2]}, \dots, d_{f[h-1]}$ in (3) represent the *expected forward degree* of nodes at $1, 2, \dots, h - 1$ hops from the source node respectively. Knowing the *expected forward degree* of a node at a stage, we can multiply it with the number of rebroadcasting nodes at the same

stage to calculate the number of nodes that are likely to rebroadcast *scout* at the next stage. Getting back to (3), we have a total of $h + 1$ terms. Hence, its closed form is

$$C_p^{(beeadhoc)} = \begin{cases} 1 + \bar{p}_c d_{avg} & \text{if } h = 1 \\ 1 + \bar{p}_c d_{avg} + \bar{p}_c d_{avg} \sum_{i=1}^{h-1} (\bar{p}_c)^i \prod_{j=1}^i d_{f[j]} & \text{otherwise} \end{cases} \quad (4)$$

The above expression validates the intuition that routing overhead $C_p^{(beeadhoc)}$ is a function of the number of hops. As the value of h increases, the routing overhead increases as well. Similarly, routing overhead of *BeeAdHoc* varies directly with *expected forward degree*. On the other hand, probability of no collision has reverse effect. Theoretically, in no collision case, number of scouts must be equal to the total number of nodes within the ring. However, being a fractional term, \bar{p}_c is likely to reduce this number substantially in dense networks.

Assuming that $d_{f[1]} = d_{f[2]} = d_{f[3]} = \dots = d_{f[h-1]} = d_f$, (4) simplifies to

$$C_p = \begin{cases} 1 + h \bar{p}_c d_f d_{avg} & \text{if } \bar{p}_c d_f = 1 \\ 1 + \bar{p}_c d_{avg} \left(\frac{1 - (\bar{p}_c d_f)^h}{1 - \bar{p}_c d_f} \right) & \text{otherwise} \end{cases} \quad (5)$$

Assumption of a constant *expected forward degree* in (5) is valid under two scenarios. Either the network is sparse where nodes are scattered and they have no or extremely small overlapping transmission regions. Secondly, it also applies to small sized networks. With these limitations, an expression for d_f is derived in [8] and we simply reproduce it here for the sake of brevity.

$$d_f \simeq \frac{d_{avg} - \rho r_0^2 \left(\frac{2\pi}{3} - \frac{\sqrt{3}}{2} \right)}{2}. \quad (6)$$

Equation (6) shows that d_f is a function of d_{avg} and r_0 respectively. The concept of constant forward degree, however, does not hold for large and dense networks. As we move away from the source node, ideally *expected forward degree* should reduce at every step. Intuitively, as more and more nodes receive the broadcasted scouts, the uncovered area reduces. Consequently, the probability to deliver to a new area reduces and hence the *expected forward degree*. Apart from its dependence upon the number of hops from the source node, *expected forward degree* must also be marginalized i.e. *expected forward degree* of nodes near the edges will fall sharply.

Now the only unknown variable in (5) is \bar{p}_c for which an expression is derived below.

4.2 Collision Modeling

As wireless channel is shared by mobile nodes, collisions are bound to happen. Having an 802.11b ad hoc network with its distributed coordination function

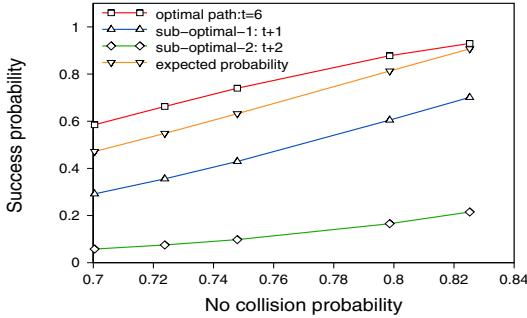


Fig. 2. Routing overhead and success probabilities of route discovery in *BeeAdHoc*

(DCF) as MAC layer protocol, probability of collision is given by the following expression [11].

$$p_c = 1 - \left(1 - \frac{1}{CW_{min}}\right)^{M-1} \quad (7)$$

where CW_{min} ($= 31$ for 802.11b) is the minimum contention window and M represents the number of contending nodes. Each of the d_{avg} neighbors may be contending for the channel access under heavy traffic. However, minimum contention case is the one in which a source initiates a route discovery process. Each node in this case is not a potential rebroadcasting node. Keeping in mind the concept of forward degree, $M = d_{avg} - d_f - 2$. Therefore, using (7), the probability of no collision is

$$\overline{p_c} = \left(1 - \frac{1}{CW_{min}}\right)^{d_{avg}-d_f-2} \quad (8)$$

(8) shows that $\overline{p_c}$ decreases exponentially with an increase in the number of nodes contending for the channel access.

5 Route Optimality

BeeAdHoc maintains multiple paths to a given destination. Each *scout* reaching the destination is likely to discover a new route. We do not consider the route caching behavior of *BeeAdHoc* in this model. Let us assume that there are k edge disjoint paths between the source-destination pair with t hops path as an optimal path. We further assume that a function $f[i-t]$ gives the total number of edge-disjoint paths of length i between source-destination pair.

5.1 Probability of Optimal Path Discovery

BeeAdHoc discovers each link with a probability $\overline{p_c}$ and hence the probability of discovering an optimal path is $\epsilon = (\overline{p_c})^{t-1}$. The probability of failure in finding

an optimal path is then $(1 - \epsilon)$. Now the probability of finding j optimal paths out of a total of $f[0]$ optimal paths is simply a binomial distribution given as

$$b(j; f[0], \epsilon) = P(X[t] = j) = \binom{f[0]}{j} \epsilon^j (1 - \epsilon)^{f[0]-j}. \quad (9)$$

where $X[t]$ represents the number of t hop paths discovered successfully. Using (9), the probability of discovering at least a single optimal path is

$$P(X[t] \geq 1) = 1 - (1 - \epsilon)^{f[0]}, \quad (10)$$

$(1 - \epsilon)^{f[0]}$ is the probability of failure in discovering an optimal path which can be minimized either by increasing the value of $f[0]$ or \bar{p}_c . However, $f[0]$ can be increased by increasing node density (ρ) which in turn reduces \bar{p}_c . Therefore, reducing the failure probability of route discovery is a tricky problem.

5.2 Probability of Suboptimal Path Discovery

Using (10), probability of discovering a suboptimal path of length $t + n$ hops where $n = 1, 2, \dots, n$ is

$$P(X[t + n] \geq 1) = 1 - (1 - \epsilon(\bar{p}_c)^n)^{f[n]} \quad (11)$$

where $f[n]$ is the number of edge-disjoint paths of length $t+n$ hops. Equation (11) shows that as $n \rightarrow \infty$, the probability of discovering a path of length $t + n$ hops approaches zero. Hence, suboptimal paths have a lower discovery probability as compared to optimal paths.

5.3 Expected Probability of Path Establishment

In addition to the path optimality problem, it is important to evaluate the marginal probability of path discovery. Using (10) and (11), the expected probability of finding a path (irrespective of the path length) is

$$E\{X\} = \sum_{i=0}^n w[i] \left(1 - (1 - \epsilon(\bar{p}_c)^i)^{f[i]}\right) \quad (12)$$

where $w[i] = \frac{f[i]}{k}$ is the normalized weight of the paths of lengths i . Expected probability of route discovery in sparse networks is close to 1 due to high value of \bar{p}_c . However, it substantially reduces in high density networks. We plotted (10), (11) and (12) (assuming $f[0] \gg f[1] \gg \dots$) for varying values of \bar{p}_c and the results are shown in Fig. 2. As the probability of no collision (\bar{p}_c) rises, so does the probability of route discoveries. Second important observation is that as \bar{p}_c approaches 1, expected probability of route discovery equals the probability of optimal path discovery. Finally, we also note that optimal paths are more probable than suboptimal paths.

6 Conclusion and Future Work

In this paper, we developed mathematical models of two key performance metrics for *BeeAdHoc* protocol: routing overhead and route optimality. Routing overhead is a function of the expected forward degree, number of hops and collisions probability. We quantified the intuition that severe contention adversely affects the route optimality and may result in suboptimal route discovery. Route optimality, on the other hand, is a function of the collision probability, number of available paths and their corresponding distribution. We also conclude that the probability of discovering optimal paths is usually higher than the probability of discovering suboptimal paths. In future work we intend to derive an exact expression for *expected forward degree* and incorporate the mobility model.

References

1. Caro, G.D., Ducatelle, F., Gambardella, L.M.: AntHocNet: An adaptive nature-inspired algorithm for routing in mobile ad hoc networks. *Telecommunications (ETT), Special Issue on Self Organization in Mobile Networking* 16(2) (2005)
2. Genes, M., Sorges, U., Bouazizi, I.: ARA - the ant-colony based routing algorithm for manets. In: *Proceedings of ICPP Workshop on Ad Hoc Networks* (2002)
3. Farooq, M.: Bee-inspired Protocol Engineering: From Nature to Networks. *Natural Computing Series*. Springer, Heidelberg (in press)
4. Wedde, H., Farooq, M., Pannenbaecke, T., Vogel, B., Mueller, C., Meth, J., Jerschkat, R.: BeeAdHoc: an energy efficient routing algorithm for mobile ad hoc networks inspired by bee behavior. In: *Proceedings of GECCO* (2005)
5. Rajagopalan, S., Shen, C.C.: ANSI: a swarm intelligence-based unicast routing protocol for hybrid ad hoc networks. *Journal of System Architecture* 52(8-9), 485–504 (2006)
6. Kurkowski, S., Camp, T., Colagrosso, M.: MANET simulation studies: The incredibles. *ACM SIGMOBILE Mobile Computing and Communications Review* 9(4), 50–61 (2005)
7. Roth, M.: The markovian termite: A soft routing framework. In: *Proceedings of IEEE Swarm Intelligence Symposium, SIS* (April 2007)
8. Saleem, M., Khayam, S.A., Farooq, M.: A formal performance modeling framework for bio-inspired ad hoc routing protocols. In: *Proceedings of GECCO* (2008)
9. Zahid, S., Shehzad, M., Ali, S.U., Farooq, M.: A comprehensive formal framework for analyzing the behavior of nature inspired routing protocols. In: *Proceedings of IEEE Congress on Evolutionary Computing (CEC)* (2007)
10. Bettstetter, C.: On the minimum node degree and connectivity of a wireless multihop network. In: *Proceedings of MobiHoc* (2002)
11. Heusse, M., Rousseau, F., Sabbaté, G.B., Duda, A.: Performance anomaly of 802.11b. In: *Proceedings of IEEE INFOCOM* (2003)

Incorporating Heuristics in a Swarm Intelligence Framework for Inferring Gene Regulatory Networks from Gene Expression Time Series

Kyriakos Kentzoglaniakis, Matthew Poole, and Carl Adams

School of Computing, University of Portsmouth, Portsmouth, UK
{kyriakos.kentzoglaniakis,matthew.poole,carl.adams}@port.ac.uk

Abstract. In this paper, we address the problem of reverse-engineering a gene regulatory network from gene expression time series. We approach the problem by implementing an ant system to generate candidate network structures. The quality of a candidate structure is evaluated using a particle swarm optimization algorithm that tunes the parameters of the corresponding model, by minimizing the error between the actual time series and the trained model's output. We extend this approach by incorporating domain-specific heuristics to the ant system, as a mechanism that has the potential to bias the pheromone amplification effect towards biologically plausible relationships. We apply the method to a subset of genes from a real world data set and report on the results.

1 Introduction

Gene expression is the process by which a gene's DNA sequence is converted through a series of steps into a functional product: the protein. This cellular process constitutes the central dogma of molecular biology, i.e. that genes code for proteins. During this process, DNA is first transcribed (copied) to an intermediate macromolecular form, the mRNA (messenger RNA), which is then translated to protein. Proteins are involved in essential functions of a living organism, including transcription, the catalysis of chemical reactions, cell signalling etc.

Certain genes code for special proteins called transcription factors, which are responsible for regulating the expression of other genes (targets). Transcription factors bind a *cis*-regulatory site in the promoter region of the target gene, thus inducing a change in the target's rate of transcription. The nature of change specifies this effect as either activatory, in case of an increase in the target's rate of transcription, or repressive (inhibitory) in case of a decrease [1].

A gene regulatory network (GRN) is a complex network of causal relationships between genes, where connections represent regulatory interactions between activators or repressors and targets.

With the advent of DNA microarray technology that measures the mRNA levels of thousands of targets, it has become possible to observe such complex biological processes by taking snapshots of the cellular state and capturing the

expression profiles of thousands of genes simultaneously. Gene expression data can either be static, with gene profiles from different organisms, each typically characterized by a class value, or dynamic in the form of gene expression time series from the same organism.

The problem of reverse-engineering GRNs from gene expression data is a major issue in systems biology [2]. A principal obstacle is the relative insufficiency of observations (typically tens or a few hundreds) compared to the number of genes measured (in the order of thousands or a few tens of thousands), the so-called curse of dimensionality.

Additionally, the common practice of validating the biological plausibility of inferred causal relationships by consulting the relevant literature, albeit unavoidable, is controversial because, in the absence of such experimental evidence for a putative connection, there is no apparent method of classifying it either as a previously unknown interaction or as just a spurious edge [3].

In this paper, we describe a swarm intelligence approach to the problem of reverse-engineering GRNs from gene expression time series. We model a GRN as a graph, upon which the ant colony optimization (ACO) meta-heuristic is implemented for the selection of putative GRN architectures. The selected structure is then modelled as a recurrent neural network (RNN), whose parameters (weights and bias terms) are optimized using particle swarm optimization (PSO), so as to minimize the error between the model's output and the actual time series.

Our approach extends the work by Ressom et al. [4], first by changing the way candidate architectures are constructed by individual artificial ants and, second, by introducing a heuristic metric with the intention to bias the probabilistic edge selection process towards biologically plausible relationships.

In the next section, we present an overview of existing approaches to the problem of GRN inference from time course gene expression data. In section 3, the proposed framework is outlined by describing its components and their interrelationships. In section 4, we report on the results of applying the method to a subset of known genes from the yeast gene expression data set and we discuss some of the issues that emerged, before the paper's conclusion in section 5.

2 Existing Approaches

The earliest approaches to the problem of inferring gene relationships from time course gene expression data, were cluster analysis methods, mostly based on global correlation metrics, such as Pearson correlation coefficient, mutual information etc., that extracted co-regulation information out of co-expressed gene clusters [5,6]. These pioneering, model-free methods essentially group genes according to their expression levels, providing an insight into the functionality of unknown genes based on the cluster in which they belong. However, they do not take the temporal nature of data into consideration and do not assign regulatory roles to genes, since, given two genes that are co-expressed (have similar expression), it is not clear which regulates the other. Nevertheless, cluster analysis is

still useful, primarily as a technique to reduce the search space and improve the performance of algorithms.

Model-based methods, on the other hand, operate by assuming the existence of a model that represents the gene regulatory network and attempt to train this model based on the available artificial or experimental data. In essence, they attempt to reconstruct the architecture by reproducing the system dynamics. Such models include Boolean networks, Bayesian networks, linear additive models, systems of differential equations, power law systems etc. [7]

In Boolean networks, the state of a node at one time point is a boolean function of the states of K other nodes at the previous time point. As such, they constitute binary idealizations of genetic network architectures that, while succeeding in the simulation and analysis of global dynamics [8], seem to suffer from the problem of information loss during data binarization.

Dynamic Bayesian networks are models of joint, multivariate probability distributions that attempt to represent conditional independence relationships between variables. Their strength in representing noisy, stochastic processes due to their probabilistic nature, makes them good candidates for addressing the problem of inferring gene regulatory networks [9].

In linear additive (neural) models [10,11], the output of each node is a combination of inputs from all other nodes, a function of the weighted sum of their expression levels. Zero weights indicate no regulation, positive weights signify activation, while negative weights signify repression. The assumption of linearity is not a severe one [12], especially if one considers the statistical treatment of microarray data and the increased levels of noise.

Ressom et al. [4] implement a swarm intelligence framework where an ant system, driven only by pheromone amplification, is used for the selection of putative network structures. For each gene (regulator), each artificial ant considers all 2^n regulator-target combinations, where n is the number of genes, for the construction of a candidate architecture. After a structure has been formed, the corresponding model (RNN) is optimized using PSO, in order to evaluate the quality of the selected structure.

Xu et al. [13] deploy a discrete version of PSO for structure selection and a continuous version for model training. They also discuss the relative difficulty of reconstructing the correct regulatory network structure over reproducing the correct dynamics, explaining that there is no unique network to satisfy the data upon which inference is based. Reconstructing the structure depends upon reproducing the system dynamics and, therefore, is a problem of higher order.

3 Methods

Our approach uses an ACO implementation, on a graph with nodes representing genes and directed edges representing regulatory (causal) relationships, to select putative network architectures, driven by pheromone amplification and heuristic information, where:

- pheromone trails are updated according to the ability of the model (RNN) that represents the selected structure to reproduce the time series, after having been trained using a PSO algorithm.
- the desirability value for a particular edge is calculated by a suitably defined heuristic function.

A candidate gene network structure is represented by a recurrent neural network model, whose update equation is given by:

$$x_i(t) = f\left(\sum_{j=1}^N w_{ij} x_j(t-1) + b_i\right) \quad (1)$$

where $x_i(t)$ is the value (expression level) of node i at time t , b_i a bias term and weights w_{ij} express the influence of node j to node i , ranging from -1 (gene j represses gene i) to 1 (gene j activates gene i). A value of 0 signifies no regulation. f is a nonlinear transfer function, either the logistic or the hyperbolic tangent.

Network architectures are constructed using the ACO meta-heuristic [14], whereby artificial ants navigate a graph of N nodes, where N is the number of genes in the time series. Each artificial ant probabilistically selects K regulator nodes for each target node in the graph, resulting in a candidate network structure $S = \{e_{ji}\}$ of NK connections. The parameter K reflects the fact that gene networks are sparse and that a gene is regulated by only a handful of other genes. An edge e_{ji} represents a regulatory relationship from node j to node i . The probability of selection of node j as a potential regulator of node i is given by:

$$p_{ij} = \frac{\tau_{ij}^\alpha \eta_{ij}^\beta}{\sum_{j=1}^N \tau_{ij}^\alpha \eta_{ij}^\beta} \quad (2)$$

where τ_{ij} is the pheromone value of edge e_{ji} , η_{ij} is the selection desirability of edge e_{ji} based on a suitably defined heuristic function and α , β are their respective relative influences.

After a candidate structure S has been constructed, its quality is assessed by tuning the corresponding model's parameters in order to compare its predicted output with the actual time series. The synaptic weights of the edges that are not part of the selected structure are locked to 0.

Optimization of the model's parameters is performed using a PSO algorithm [15], where each particle's position is encoded as a vector \mathbf{w}_S of size $N(K+1)$ that contains the weights of the selected edges, as well as the bias terms. The quality of a particle's position is determined by calculating the MSE between the predicted model output and the actual time series:

$$\epsilon(\mathbf{w}_S) = \frac{1}{TN} \sum_{t=1}^T \sum_{i=1}^N [x_i(t) - x_i^{\mathbf{w}_S}(t)]^2 \quad (3)$$

where T is the number of available time points, N is the number of genes, $x_i(t)$ is the actual expression level of the i^{th} gene at time t and $x_i^{\mathbf{w}_S}(t)$ is the predicted

expression level of the i^{th} gene at time t . The predicted time series are calculated by setting up the model using \mathbf{w}_S and running it using each state of the actual time series, in order to obtain the next state of the predicted time series.

After the threshold of maximum allowed PSO iterations has been reached, the minimum achieved error $\epsilon(\mathbf{w}_S)$ is returned to the ACO algorithm as the quality of the selected structure S . The pheromone matrix is then updated according to:

$$\tau_{ij} = \frac{1}{\epsilon(\mathbf{w}_S)} \quad \forall e_{ji} \in S \quad (4)$$

The incorporation of heuristics to probabilistic structure selection offers a way of enriching a domain-agnostic procedure with problem-specific insights. The heuristic factor η_{ij} from equation (2) can be defined as a function $\eta : \mathcal{N} \times \mathcal{N} \rightarrow \mathcal{R}$ that maps a pair (i, j) to a score that reflects the strength and nature of gene's j influence on gene i . In this context, strength means the likelihood of regulation and nature means the type of regulation (activation or repression).

For the purpose of demonstrating our approach, we are using a heuristic proposed by Kwon et al. [16]. They hypothesize that if a rise in the expression of gene A is followed by a rise in the expression of gene B, then this indicates that gene A potentially activates gene B. Conversely, if a rise in the expression of gene A is followed by a fall in the expression of gene B, then gene A is a potential repressor for gene B.

These expression changes in a gene's temporal profile are encoded as 'events', by calculating the slope of the expression profile at every time interval and classifying it as either 'R' (rising), 'F' (falling) or 'C' (constant). A variation of the Needleman-Wunsch algorithm for sequence alignment [17] is then used to determine the best possible alignment for a pair of event strings, by using the event scoring matrix shown in Table 1.

Given the expression levels of two genes, one of which is assumed to be the regulator and the other the target, the algorithm first calculates the score for the presumed activatory relationship and then for the inhibitory relationship, by complementing the event string of the target. This is done by swapping 'R's with 'F's, while 'C's remain intact. The maximum score of the two is returned to ACO as the overall score of the particular relationship and is cached to avoid recalculation.

Table 1. Scoring matrix for event matching. The score of a pair of symbols is a function of the time lag dt between two events. $S(dt)$ is a linearly decreasing function with $0 < S(dt) < 1$, so that the bigger the time lag, the less likely a causal effect is to be assumed. In case of a negative dt , the match is assigned a maximum penalty. Parameters a and b range from 0 to 1 and their role is to emphasize particular matching forms, based on biological arguments [16].

	R	C	F
R	$S(dt)$	0	$-bS(dt)$
C	0	0	0
F	$-bS(dt)$	0	$aS(dt)$

4 Results

We selected 5 cyclin genes that are known to be involved in cell cycle regulation, from the *S. cerevisiae* (yeast) data set published in Spellman et al. [18], for the purpose of comparing our results to those of Ressom et al. [4]. The yeast data set contains multiple time series from the yeast cell cycle; we chose the *cdc15* time series, consisting of 24 time points (more than the others). Gene expression levels were first smoothed, by using a sliding window method (convolution of a scaled Hann window with the expression profile), and consequently normalized between 0 and 1.

Table 2. The known relations for the collection of selected genes come from PathwayStudio software, as reported in Ressom et al. [4]. The last column summarizes how our algorithm compares with their predictions.

Relation Type	Known Relation	Predicted by [4]	Our Prediction
Expression	CLB1 \leftarrow CLB6	yes (reversed)	yes
Expression	CLB1 \leftrightarrow CLB2	yes	yes
Regulation	CLB6 \rightarrow CLB5	yes (reversed)	yes
Regulation	CLB6 $\rightarrow\rightarrow$ CLB2	no	yes (opposite sign)
Regulation	CLB1 \leftrightarrow CLB5	yes (reversed)	no
MolSynthesis	CLB1 $\rightarrow\rightarrow$ CLB2	yes	yes
Direct Regulation	CLB6 $\rightarrow\rightarrow$ cdc28	yes (reversed)	yes (reversed)
Direct Regulation	CLB5 $\rightarrow\rightarrow$ cdc28	yes	yes
Direct Regulation	CLB2 $\rightarrow\rightarrow$ cdc28	yes	yes
Direct Regulation	CLB2 \leftrightarrow CLB5	no	yes (opposite sign)
Direct Regulation	CLB1 $\rightarrow\rightarrow$ cdc28	yes	no

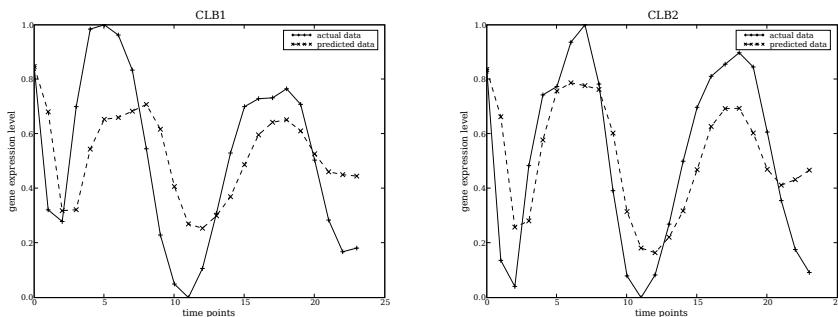
For the PSO implementation we used a swarm with the global best topology, a population size of 15 particles, a maximum number of 2000 iterations, $\phi_1 = \phi_2 = 2$ and a random inertia weight ω drawn from a uniform distribution, ranging from $\omega_{min} = 0.3$ to $\omega_{max} = 0.8$.

The settings for ACO were set as follows: the relative influences of pheromone and heuristic value $\alpha = 1$ and $\beta = 1$ respectively, the pheromone evaporation rate $\rho = 0.1$ and the number of regulators for a given target gene $K = 2$. The colony size was set to 5 and it was allowed to run for 50 steps.

We performed 10 such experiments and recorded the number of times each edge was selected. We considered a particular relationship to be inferred if the corresponding graph edge was selected at least half of the times, during all experiments. The average MSE of RNN training was 0.058 with a standard deviation of 0.0026.

The results, as shown in Table 2, do not indicate a notable (if any) improvement over the predictions in [4]. The incorporation of the selected heuristic metric does not seem to influence structure selection in a decisive manner. Perhaps, this is due to the relative influences of pheromone value and heuristic desirability, α and β , being equally weighted.

Table 3. Two examples of actual gene expression levels from the original time series and predicted levels from the optimal RNN that resulted from the experiments



Two of our predicted, putative connections, namely $\text{CLB2} \rightarrow \text{CLB6}$ and $\text{CLB5} \rightarrow \text{CLB6}$, are not reported as known relationships by [4] and their biological plausibility can only be verified experimentally.

5 Further Work

The reported early results that have been presented in this paper, form part of an ongoing study into a swarm intelligence perspective to the problem of reverse-engineering gene regulatory networks. The proposed framework allows for the incorporation of an arbitrary number of problem-specific heuristics, perhaps with an appropriately defined weighting scheme, to a model-based optimization approach.

The behaviour of the ant system needs to be studied in relation to the values of its parameters and the aggregation of heuristics. The suitability of different models, representing selected structures, is also a path to be explored.

Furthermore, we note that our experiments have used a hand-picked subset of temporal gene expression profiles. An investigation of the algorithm's scalability is necessary, particularly when considering the full set of genes, whose expression levels are captured in a real world data set.

References

1. Alon, U.: An introduction to systems biology: design principles of biological circuits. Chapman & Hall/CRC, Boca Raton (2007)
2. Kitano, H.: Computational systems biology. *Nature* 420, 206–210 (2002)
3. Husmeier, D.: Sensitivity and specificity of inferring genetic regulatory interactions from microarray experiments with dynamic Bayesian networks. *Bioinformatics* 19(17), 2271–2282 (2003)
4. Ressom, H., Zhang, Y., Xuan, J., Wang, Y., Clarke, R.: Inference of gene regulatory networks from time course gene expression data using neural networks and swarm intelligence. In: IEEE Symposium on Computational Intelligence and Bioinformatics and Computational Biology, pp. 1–8 (2006)

5. Eisen, M., Spellman, P., Brown, P., Botstein, D.: Cluster analysis and display of genome-wide expression patterns. *PNAS* 95(25), 14863–14868 (1998)
6. D'Haeseleer, P., Wen, X., Fuhrman, S.: Mining the gene expression matrix: inferring gene relationships from large scale gene expression data. In: Second International Workshop on Information Processing in Cell and Tissues, pp. 203–212 (1998)
7. de Jong, H.: Modeling and simulation of genetic regulatory systems: a literature review. *Journal of Computational Biology* 9(1), 69–105 (2002)
8. Somogyi, R., Fuhrman, S., Askenazi, M.: The gene expression matrix: towards the extraction of genetic network architectures. *Nonlinear Analysis, Theory, Methods & Applications* 30(3), 1815–1824 (1997)
9. Perrin, B., Ralaivola, L., Mazurie, A., Bottani, S., Mallet, J., d'Alche Buc, F.: Gene networks inference using dynamic Bayesian networks. *Bioinformatics* 19(suppl. 2), ii 138–ii 148 (2003)
10. Vohradsky, J.: Neural model of the genetic network. *Journal of Biological Chemistry* 276(39), 36168–36173 (2001)
11. Wahde, M., Hertz, J.: Modeling Genetic Regulatory Dynamics in Neural Development. *Journal of Computational Biology* 8(4), 429–442 (2001)
12. Pournara, I., Wernisch, L.: Factor analysis for gene regulatory networks and transcription factor activity profiles. *BMC Bioinformatics* 8(61) (2007)
13. Xu, R., Wunsch, D.C.I., Frank, R.: Inference of genetic regulatory networks with recurrent neural network models using particle swarm optimization. *IEEE/ACM Transactions on Computational Biology and Bioinformatics* 4(4), 681–692 (2007)
14. Bonabeau, E., Dorigo, M., Theraulaz, G.: *Swarm Intelligence: from natural to artificial systems*. Oxford University Press, Oxford (1999)
15. Kennedy, J., Eberhart, R.: Particle Swarm Optimization. In: *IEEE International Conference on Neural Networks*, vol. 4, pp. 1942–1948 (1995)
16. Kwon, A., Hoos, H., Ng, R.: Inference of transcriptional regulation relationships from gene expression data. *Bioinformatics* 19(8), 905–912 (2003)
17. Needleman, S., Wunsch, C.: A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology* 48, 443–453 (1970)
18. Spellman, P., Sherlock, G., Zhang, M., Iyer, V., Anders, K., Eisen, M., Brown, P.O., Botstein, D., Futcher, B.: Comprehensive Identification of Cell Cycle-regulated Genes of the Yeast *Saccharomyces cerevisiae* by Microarray Hybridization. *Molecular Biology of the Cell* 9, 3273–3297 (1998)

Incorporating Preferences to a Multi-objective Ant Colony Algorithm for Time and Space Assembly Line Balancing*

Manuel Chica¹, Óscar Cordón¹, Sergio Damas¹,
Jordi Pereira², and Joaquín Bautista²

¹ European Centre for Soft Computing, Mieres (Asturias), Spain
`{manuel.chica,oscar.cordon,sergio.damas}@softcomputing.es`

² Universitat Politècnica de Catalunya, Barcelona, Spain
`{joaquin.bautista,jorge.pereira}@upc.edu`

Abstract. We present an extension of a multi-objective algorithm based on Ant Colony Optimisation to solve a more realistic variant of a classical industrial problem: Time and Space Assembly Line Balancing. We study the influence of incorporating some domain knowledge by guiding the search process of the algorithm with preferences-based dominance. Our approach is compared with other techniques, and every algorithm tackles a real-world instance from a Nissan plant. We prove that the embedded expert knowledge is even more justified in a real-world problem.

1 Introduction

The Time and Space Assembly Line Balancing Problem (TSALBP) [1] belongs to a family of academic problems which came up with the name of Simple Assembly Line Balancing Problem (SALBP) [2]. It considers an additional space constraint to become a simplified but closer version to real-world problems. In this paper we tackle the 1/3 variant of the TSALBP, which tries to minimise the number of stations and their area for a given product cycle time, a very complex and realistic multi-criteria problem in the automotive industry.

In our previous work [3], we successfully solved the TSALBP-1/3 by means of multi-objective and constructive approaches. We selected the Multiple Ant Colony System (MACS) algorithm [4] because of its good performance to develop a multi-objective ant colony optimisation (MOACO) proposal [5] for it. In the current contribution we aim to extend this proposal by incorporating problem-specific information provided by the plant experts in the form of preferences in the decision variable space which will allow us to guide the search. We will use an *a priori* approach to incorporate these expertise although *a posteriori* and *interactive* procedures have been mostly applied in the Evolutionary Multi-objective Optimisation and Operations Research communities [6].

* UPC Nissan Chair as well as the Spanish Government partially funded this work by means of PROTHIUS-II project: DPI2007-63026 including EDRF fundings.

This improved model aims to reduce the size of the Pareto set and, collaterally, increase the quality of the Pareto front by increasing the MACS convergence capability and focusing on actually interesting and useful solutions for the decision-maker using problem-specific knowledge.

Our MACS algorithm is applied both to academic real-like problem instances and to a real-world instance which has more peculiarities than the others. It corresponds to the assembly process of the Nissan Pathfinder engine, developed at the Nissan industry plant of Barcelona (Spain).

The paper is structured as follows. In Section 2, the problem formulation, and our previous MOACO proposal are explained. The preferences based on domain knowledge are detailed in Section 3. Experiments and analysis of results are shown in Section 4. In Section 5, some concluding remarks are discussed.

2 Preliminaries

In this section the problem preliminaries and our previous MOACO proposal are presented. First, an overview of the assembly line balancing problem is discussed. Then, the main features of the MACS algorithm are briefly described.

2.1 The Time and Space Assembly Line Balancing Problem

The manufacturing of a production item is divided up into a set V of n tasks. Each task j requires an operation time for its execution $t_j > 0$ that is determined as a function of the manufacturing technologies and the employed resources. Each station k is assigned to a subset of tasks S_k ($S_k \subseteq V$), called its workload. A task j must be assigned to one station k .

Each task j has a set of direct predecessors, P_j , which must be accomplished before starting it. These constraints are normally represented by means of an acyclic precedence graph, whose vertices stand for the tasks and where a directed arc (i, j) indicates that task i must be finished before starting task j on the production line. Thus, if $i \in S_h$ and $j \in S_k$, then $h \leq k$ must be fulfilled. Each station k presents a station workload time $t(S_k)$ that is equal to the sum of the tasks' lengths assigned to the station k . SALBP [2] focuses on grouping tasks in workstations by an efficient and coherent way. There is a large variety of exact and heuristic problem-solving procedures [7], even an hybrid ant colony algorithm-beam search approach has been recently developed in [8].

The need of introducing space constraints in the assembly lines' design is based on two main reasons: (a) the length of the workstation is limited in the majority of the situations, and (b) the required tools and components to be assembled should be distributed along the sides of the line. Hence, an area constraint may be considered by associating a required area a_j to each task j and an available area A_k to each station k that, for the sake of simplicity, we shall assume it to be identical for every station and equal to $A : A = \max_{\forall k \in \{1..n\}} \{A_k\}$. Thus, each station k requires a station area $a(S_k)$ that is equal to the sum of areas required by the tasks assigned to station k .

This leads us to a new family of problems called TSALBP in [1]. It may be stated as: given a set of n tasks with their temporal t_j and spatial a_j attributes ($1 \leq j \leq n$) and a precedence graph, each task must be assigned to a single station such that: (i) every precedence constraint is satisfied, (ii) no station workload time ($t(S_k)$) is greater than the cycle time (c), and (iii) no area required by any station ($a(S_k)$) is greater than the available area per station (A).

TSALBP presents eight variants depending on three optimization criteria: m (the number of stations), c (the cycle time) and A (the area of the stations). Within these variants there are four multi-objective problems and we will tackle one of them, the TSALBP-1/3. It consists of minimising the number of stations m and the station area A , given a fixed value of the cycle time c . We chose this variant because it is quite realistic in the automotive industry since the annual production of an industrial plant (and therefore, the cycle time c) is usually set by some market objectives. Besides, the search for the best number of stations and areas makes sense if we want to reduce costs and make workers' day better by setting up less crowded stations. For more information we refer to [3].

2.2 A MACS Algorithm to Solve TSALBP-1/3

In this section, a brief summary of our previous multi-objective proposal based on the MACS algorithm is presented. The complete MACS description can be found in [4], and our proposal is detailed in [3].

Since the number of stations is not fixed, we use a constructive and station-oriented approach (as usually done for the SALBP [7]) to face the precedence problem. Thus, our algorithm will open a station and select one task till a stopping criterion is reached. Then, a new station is again opened to be filled.

Experiments showed that the performance is better if MACS is only guided by the pheromone trail information. Such information has to memorise which tasks are the most appropriate to be assigned to a station. Hence, pheromone has to be associated to a pair $(\text{station}_k, \text{task}_j)$, $k = 1 \dots n$, $j = 1 \dots n$, so our pheromone trail matrix has a bi-dimensional nature. We used two station-oriented single-objective greedy algorithms to obtain the initial pheromone value τ_0 .

In addition, we introduced a new mechanism in the construction algorithm to close a station according to a probability distribution, given by the filling rate of the station: $p(\text{closing}) = (\sum_{\forall i \in S_k} t_i) / c$. It helps the algorithm to reach more diverse solutions from closing stations by a deterministic process. The probability is computed at each construction step so its value is progressively increased. Then, a random number is generated to decide if the station is closed.

Besides, there is a need to achieve a better intensification-diversification trade-off. That was achieved by introducing different filling thresholds associated to the ants. These thresholds make the different ants have a different search behaviour. The higher the ant's threshold, the more filled the station will be (there will be less possibilities to close the station during its creation process).

In this way, the ant population will show a highly diverse search behaviour, allowing the algorithm to properly explore the different parts of the optimal Pareto front by appropriately spreading the generated solutions.

3 Adding Preferences Based on Domain Knowledge

In this work, we have included some expert information into the dominance definition considered by the algorithm (using an *a priori* approach). This addition of domain knowledge allows us to derive a Pareto set composed of a smaller number of more likely solutions for the final user as well as to induce a better convergence to the actual Pareto front as a collateral effect [6]. In this section we describe how we have modified the original dominance definition including specific preferences given by the expert for the current problem.

We have applied these preferences based on domain knowledge when there are some solutions with the same objective values¹, i.e. the same value for area and number of stations (A and m) for a fixed cycle time (c). Decision between two solutions with different c , A and m values is made by using the traditional dominance relationship.

Since the number of solutions of this kind can be quite large, thus difficulting the final choice of the best one for each combination of objective values to the user, it is important to establish a criterion based on the expert's preferences to choose, among those solutions, the one having the best quality according to the industrial context. Thus, we can discriminate between two solutions with same c , A and m values considering that:

- (a) The workload of the plant must be well-balanced in every station. For m stations, all the station workload times $t(S_k)$ for $k = 1..m$ are alike. Due to this criterion, and considering the same number of employees per station, a well-balanced plant provides less human resources' conflicts. Likewise, it eliminates the need of shifts among the workers of the different stations.
- (b) The required space for worker's instruments must be as similar as possible. This preference aims to offer solutions in which every worker has the same working conditions. If we reduce the extra effort in movements and the crowding sensation, it will eliminate industrial disputes.

The latter criteria are formulated through the following preference measures:

$$P_t(\sigma) = \sum_{k=1}^m (c - t(S_k))^2 \quad P_a(\sigma) = \sum_{k=1}^m (A - a(S_k))^2$$

where σ represents a solution with known c , A and m values. $S_k, \forall k = 1..m$ is the assignment of the different tasks to the k -th station in σ .

Bearing in mind these measures, the following preferences-based dominance relations can be considered:

Definition 1. A solution σ_1 is said to partially dominate another solution σ_2 with respect to time, both with identical c , A and m values, if $P_t(\sigma_1) < P_t(\sigma_2)$.

¹ Notice that preferences are usually applied to guide the search to the specific Pareto front zones that are interesting for the user (i.e. in the objective space) while, in our case, it is applied on the decision variable space to reduce the number of solutions considering different task assignments presenting the same objective values.

Table 1. Used parameter values

Parameter	Value	Parameter	Value
GENERAL		MACS/ACS	
Number of runs	10 (except for ACS)	Number of ants	10
Max. run time	900 seconds	β	2
PC Specs.	Intel Pentium™ D 2 CPUs at 2.80GHz	ρ	0.2
OS	CentOS Linux 4.0 GCC 3.4.6	q_0	0.2
		Ants' thresholds	{0.2, 0.4, 0.6, 0.7, 0.9} (2 ants per threshold)
MORGA		ACS	
α_{MORGA}	0.3	Number of runs	11 (one for each α_{ACS} value)
Diversity thresholds	{0.2, 0.4, 0.6, 0.7, 0.9}	α_{ACS} for the objective aggregation	{0, 0.1, ..., 0.9, 1}

Definition 2. A solution σ_1 is said to partially dominate another solution σ_2 with respect to space, both with identical c , A and m values, if $P_a(\sigma_1) < P_a(\sigma_2)$.

Definition 3. A solution σ_1 is said to completely dominate another solution σ_2 with respect to time and space, both with identical c , A and m values, if:

$$([P_t(\sigma_1) \leq P_t(\sigma_2)] \wedge [P_a(\sigma_1) < P_a(\sigma_2)]) \vee ([P_t(\sigma_1) < P_t(\sigma_2)] \wedge [P_a(\sigma_1) \leq P_a(\sigma_2)])$$

4 Experiments

We present two different algorithms, MORGA and ACS, to compare our MACS algorithm with them. Then, the considered experimental setup is showed. We finally lay out the developed experimentation and discuss the obtained results considering multi-objective metrics and some graphics.

4.1 Other Approaches to TSALBP-1/3

Apart from the MACS algorithm, we have also designed a modified ACS algorithm and a multi-objective randomised greedy algorithm (MORGA), already proposed in [3]. In both cases, we use the same constructive approach, non-dominated solution archive, and all the remaining multi-objective mechanisms than in our MACS algorithm. The ACS algorithm has been developed according to the original version [9] but including a weighted aggregation of the two objectives. An approximate Pareto set is built by fusing all the obtained solutions of the different runs of the algorithm. On the other hand, MORGA has a diversification generation mechanism that behaves similarly to a GRASP construction phase. We use an α_{MORGA} parameter which stands for the diversification-intensification trade-off control parameter in the decision step of the algorithm. The most suitable value for α_{MORGA} is 0.3 (see [3]), the one used in this work.

4.2 Problem Instances and Parameter Values

Five real-like problem instances have been selected for the experimentation (see Table 2). Originally, these instances were SALBP-1 instances ² only having time

² Available at <http://www.assembly-line-balancing.de>

Table 2. Mean and standard deviation values ($\bar{x}(\sigma)$) of the unary metrics for barthol2, barthold, lutz2, nissan, scholl and weemag instances

	barthol2	barthold	lutz2	nissan	scholl	weemag
No. of non-dominated solutions						
MORG A	12.1 (1.5)	<u>2437.2</u> (578.3)	<u>1227.3</u> (652.7)	<u>809.9</u> (103.6)	<u>595.4</u> (67.5)	10.8 (2.3)
ACS	5 (0)	2 (0)	5 (0)	3 (0)	3 (0)	5 (0)
MACS	<u>13.5</u> (2.8)	12 (1.4)	268.9 (22.4)	571.9 (81.1)	50.6 (85.7)	<u>15.6</u> (4.4)
MACS prefs.	10.8 (1.5)	12 (1.2)	7.6 (1.0)	7.2 (0.8)	13 (2.1)	7.9 (1.2)
No. of different Pareto front solutions						
MORG A	7.1 (0.5)	9.3 (1.6)	<u>7.4</u> (0.8)	<u>7.6</u> (0.7)	3.9 (0.7)	6.7 (0.9)
ACS	5 (0)	2 (0)	5 (0)	3 (0)	3 (0)	5 (0)
MACS	<u>12.8</u> (2.8)	11 (0.9)	6.7 (0.6)	<u>7.6</u> (1.0)	<u>14.6</u> (2.0)	<u>8.2</u> (1.5)
MACS prefs.	10.8 (1.5)	<u>12</u> (1.2)	7 (0.8)	7.2 (0.8)	13 (2.1)	7.8 (1.2)
Metric S						
MORG A	<u>393060.69</u>	711632.81	26115	8815.17	16337440	63679.2
	(28.2)	(58.3)	(178.0)	(7.9)	(752.6)	(21.5)
ACS	390161 (0)	<u>709528</u> (0)	<u>26470</u> (0)	8713 (0)	16458204 (0)	65062 (0)
MACS	391719.09	725348.19	26027.3	<u>8889.75</u>	16550586	65148.1
	(1204.8)	(2127.4)	(3.5)	(0.7)	(6318.0)	(5.7)
MACS prefs.	391410.59	<u>726088</u>	26071.1	8864.45	<u>16552952</u>	<u>65151.6</u>
	(166.4)	(2202.9)	(135.7)	(31.9)	(7248.2)	(17.5)
Metric M2*						
MORG A	5.82 (0.7)	7.29 (1.0)	<u>6.89</u> (0.7)	6.73 (0.3)	3.85 (0.6)	6.35 (0.9)
ACS	4 (0)	2 (0)	5 (0)	3 (0)	2 (0)	4.5 (0)
MACS	<u>10.86</u> (2.1)	9.49 (0.6)	6.21 (0.5)	<u>6.88</u> (0.8)	<u>11.51</u> (1.4)	<u>7.46</u> (1.3)
MACS prefs.	9.38 (1.2)	<u>10.19</u> (1.0)	6.62 (0.7)	6.54 (0.7)	10.09 (1.7)	7.15 (1.1)
Metric M3*						
MORG A	<u>80.6</u> (0.2)	684.09 (0)	<u>22.45</u> (3.7)	21.11 (1.6)	2163.08 (0)	20.56 (1.7)
ACS	58.6 (0)	<u>712.08</u> (0)	21.21 (0)	8.25 (0)	<u>3563.24</u> (0)	24.19 (0)
MACS	61.99 (12.9)	407.91 (21.0)	19.73 (0.6)	<u>21.12</u> (1.3)	1645.49 (38.7)	<u>24.61</u> (1.0)
MACS prefs.	64.82 (6.6)	403.31 (23.3)	20.27 (0.9)	19.62 (2.6)	1658.79 (40.8)	24.39 (1.6)
No. of applications of preferences-based dominance						
MACS prefs.	8.3 (3.0)	5.6 (2.9)	478.1 (105.4)	935.4 (231.4)	240.2 (145.1)	39.5 (18.2)

information. However, we have created their area information by reverting the task graph to make them bi-objective (as done in [1]). Apart from these real-like instances, we have considered a real-world problem corresponding to the assembly process of the Nissan Pathfinder engine, developed at the Nissan industry plant of Barcelona (Spain). See [1] for additional information on this problem instance.

The MACS algorithm and MORG A have been run ten times with ten different seeds for each of the five real-like and the Nissan instances. The ACS algorithm has been run eleven times with different values for the α_{ACS} parameter in order to spread all the extent of the Pareto front. Every considered parameter value is shown in Table 1.

4.3 Results Analysis

We have applied some multi-objective unary metrics to measure the performance of the different approaches: the number of total and different (in the objective

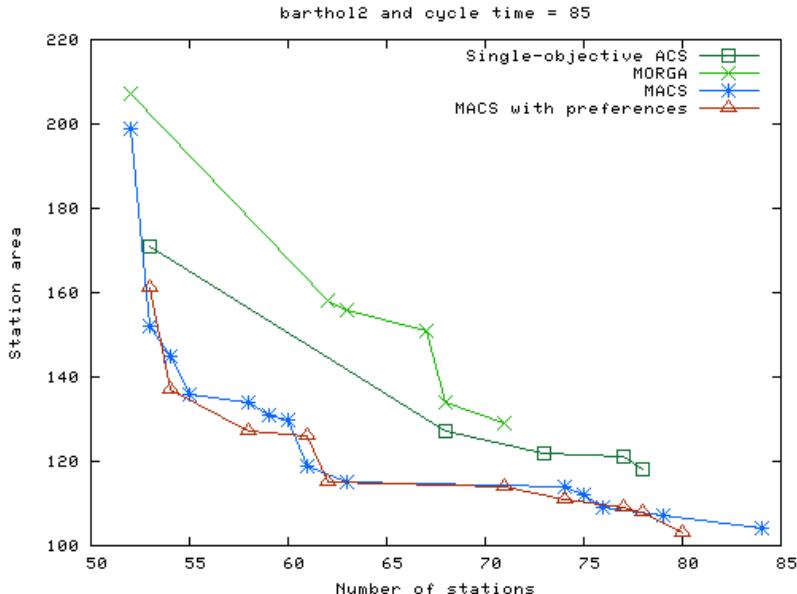


Fig. 1. Pareto fronts for the `barthol2` problem instance

vectors) Pareto solutions returned by each algorithm, and the S , $M2^*$ and $M3^*$ metrics [6]. S , the size of the space covered, measures the volume enclosed by the Pareto front, $M2^*$ evaluates the distribution of the solutions and $M3^*$ evaluates the extent of the obtained Pareto fronts³. In addition, the number of applications of the preferences-based dominance criterion for each problem is shown in Table 2. We can also observe the obtained values for the different unary metrics. The different variants of the MACS algorithm achieve better and vaster Pareto fronts than MORGA and ACS in almost every case. Mainly, they obtain more diverse Pareto fronts with a better convergence. The single-objective ACS algorithm is the worst choice because it only achieves few solutions in the Pareto set. However, the values of $M3^*$ for ACS are higher than MACS. This behaviour can be explained since ACS convergence to the actual Pareto front is not good enough, above all, in its most left part.

The preferences-based MACS variant shows the best convergence and reduces the number of non-dominated solutions with the same objective values as expected while keeping a similar value of different solutions. In some cases, this reduction is quite important (see `nissan` instance, from an average of 571.9 solutions to 7.2). We should highlight that the real-world instance of Nissan is the most appropriate to use preferences based on domain knowledge. Indeed, the number of applications of the preferences-based dominance is the highest.

The graphical representation of the returned aggregated Pareto fronts for the `barthol2` instance is shown in Figure 1. We can arrive to the same previous

³ $M1^*$ has not been applied because we do not know the optimal Pareto fronts.

conclusions by observing it. MACS variants are the best approaches, and even MACS with preferences achieves a better convergence. MORGA and ACS are not able to reach all the Pareto front surface, getting only few solutions. We have only included the Pareto front of this problem instances for the lack of space but pretty similar behaviours are obtained in the remainder. For the same reason, the binary metric C [6] is not included here, but their values and boxplots can be found in an external appendix at <http://www.nissanchair.com/TSALBP/>

5 Concluding Remarks

An existing MOACO proposal has been extended by introducing preferences based on domain knowledge to tackle the TSALBP-1/3. Other competitive alternatives to baseline the performance of our MOACO proposal, MORGA and a modified ACS, have been used. From the obtained results we have found out that the MACS algorithm gets better results than MORGA and ACS in every considered metric. It has a better convergence, distribution and number of solutions. The enrichment of MACS with domain knowledge provides excellent results, reducing the number of solutions in the Pareto set which have the same objective values and getting a better convergence. Our improved proposal is especially suitable for real-world problems like Nissan's one.

References

1. Bautista, J., Pereira, J.: Ant algorithms for a time and space constrained assembly line balancing problem. *European Journal of Operational Research* 177, 2016–2032 (2007)
2. Scholl, A.: Balancing and Sequencing of Assembly Lines, 2nd edn. Physica-Verlag, Heidelberg (1999)
3. Chica, M., Cordón, O., Damas, S., Bautista, J., Pereira, J.: Multi-objective, constructive heuristics for the 1/3 variant of the time and space assembly line balancing problem: ACO and randomised greedy. Technical Report AFE-08-01, European Centre for Soft Computing, Asturias (Spain) (2008)
4. Barán, B., Schaeerer, M.: A multiobjective ant colony system for vehicle routing problem with time windows. In: 21st IASTED Conf., Germany, pp. 97–102 (2003)
5. García Martínez, C., Cordón, O., Herrera, F.: A taxonomy and an empirical analysis of multiple objective ant colony optimization algorithms for the bi-criteria TSP. *European Journal of Operational Research* 180, 116–148 (2007)
6. Coello, C.A., Lamont, G.B., Van Veldhuizen, D.A.: Evolutionary Algorithms for Solving Multi-objective Problems, 2nd edn. Springer, Heidelberg (2007)
7. Scholl, A., Becker, C.: State-of-the-art exact and heuristic solution procedures for simple assembly line balancing. *European Journal of Operational Research* 168, 666–693 (2006)
8. Blum, C., Bautista, J., Pereira, J.: Beam-ACO applied to assembly line balancing. In: Dorigo, M., Gambardella, L.M., Birattari, M., Martinoli, A., Poli, R., Stützle, T. (eds.) ANTS 2006. LNCS, vol. 4150, pp. 96–107. Springer, Heidelberg (2006)
9. Dorigo, M., Gambardella, L.: Ant colony system: a cooperative learning approach to the traveling salesman problem. *IEEE Transactions on Evolutionary Computation* 1(1), 53–66 (1997)

KANTS: Artificial Ant System for Classification

Carlos Fernandes^{1,2}, Antonio Miguel Mora², Juan Julián Merelo²,
Vitorino Ramos¹, Juan Luís Laredo², and Agostinho Rosa¹

¹ LASEEB-ISR/IST, University of Lisbon, Portugal
`{cfernandes,vramos,acrosa}@laseeb.org`

² Dep. de Arquitectura y Tecnología de Computadores, University of Granada, Spain
`{amorag,jmerelo,juanlu}@geneara.ugr.es`

Abstract. This paper investigates a new model that takes advantage of the cooperative self-organization of Ant Algorithms to evolve a naturally inspired pattern recognition (and also clustering) method. The approach considers each data item as an ant that changes the environment as it moves through it. The algorithm is successfully applied to well-known classification problems and yields better results than some other classification approaches, like K-Nearest Neighbours and Neural Networks.

1 Introduction and State of the Art

Clustering is performed naturally by some types of ants in two different ways. First, ant colonies recognize by odour other members of their colony [1] leading to a natural clustering of ants belonging to the same nest; second, ants do physically cluster their larvae and dead bodies, putting them in piles whose position and size is completely self-organized [2]. Ant Algorithms inspired by these models, such as those proposed in [1,3,4], have been applied to clustering and classification. In general, these methods follow the second clustering behavior: data for training the clusters is represented as *dead bodies* that ants have to pick up (following some rule) and drop, while at the same time deposit and follow pheromone, but another approach is possible by considering each data item as an ant.

This paper presents KohonAnts (or KANTS), an Ant Algorithm that merges the biologically inspired concepts of Kohonen's Self-Organizing Map (SOM) [5] with Chialvo and Millona's Ant System (AS) [6] and introduces a few new ideas in order to deal with classification. First, every ant represents a data item. Ants move in a grid dropping *vectorial* pheromones. The grid is initially filled with random vectors (of the same dimension as the data), and every time an ant visits a cell it changes the pheromone following a method similar to that used by SOM, by "pushing" the pheromone closer to the data stored in the ant itself. Since ants move around in the grid, ant position and pheromone co-adapt, so that eventually ants with similar data are close together in the grid (*nesting* behavior), and the grid itself contains vectors similar to those stored in the ants on top of them. The grid can then be used to classify in the same way as SOM, while ants can be used to visually identify the clusters' position. The interesting part of the method is that self-organization comes through stigmergy:

ants change the environment (pheromones in the grid), and that influences the behavior of the rest of the ants.

2 Preliminary Concepts

SOM, introduced by Teuvo Kohonen [5] as a non-supervised neural network that tries to imitate the self-organization of the sensory cortex of the human brain, may be used as a clustering/classification tool or as a method to find unknown relationships in a set of variables that describe a problem. SOM makes a nonlinear projection from a high-dimensional data space (one dimension per variable) on a regular, low-dimensional (usually 2D) grid of neurons. Since this type of network is distributed in a plane it can be concluded that the projections preserve the topologic relations while simultaneously creating a dimensional reduction of the representation space. SOM processes a set of input vectors (samples or patterns), which are composed by variables (features) typifying each sample, and creates an output topological network where each neuron is associated also to a vector of variables (model vector) which is representative of a group of the input vectors. The consecutive iterations of the method have the effect of 'moving' the model vectors from the winning neuron towards the input vector: vectors tend to follow the distribution of the input vectors. Consequently, the algorithm leads to a topological arrangement of the characteristic map of the input space, in the sense that adjacent neurons in the network tend to have similar weight vectors. Then, looking at the SOM's display, it is possible to recognize some clusters as well as the metric-topological relations of the data items and the variables.

AS [6] is an ant model where trails and networks of ant traffic emerge without impositions by any special boundary conditions, lattice topology, or additional behavioral rules. The state of an ant can be expressed by its position r and orientation θ . Transition rules were derived from noisy response functions, which in turn were found to reproduce a number of experimental results with real ants. The response function can effectively be translated into a two-parameter transition rule between the cells by using a pheromone weighting function,

$$W(\sigma) = \left(1 + \frac{\delta}{1 + \sigma \cdot \delta}\right)^\beta \quad (1)$$

that measures the relative probabilities of moving to a cell r with pheromone density $\sigma(r)$. Parameter β (associated with the osmotropotaxic sensitivity [7]) controls the degree of randomness with which each ant follows the pheromone's gradient [6]. Sensory capacity $1/\delta$ describes the fact that each ant's ability to sense pheromone decreases somewhat at high concentrations. (In addition to the former equation, there is a weighting factor $w(\Delta\theta)$, where $\Delta\theta$ is the change in direction at each time step. This factor ensures that very sharp turns are much less likely than turns through smaller angles.) In test some assumptions, a discretization of the model is necessary and for that purpose a square lattice where ants can move around, taking one step at every iteration, was created. The decision (where to go) is made according to the pheromone concentration in all eight neighboring cells (Von Neumann neighbourhood). As an additional

Algorithm 1. KANTS Algorithm

```

initialize_randomly_grid_vectors
place_randomly_ants_in_grid
for N_iterations do
    for each ant  $a$  at cell  $(x, y)$  do
         $j = \text{decide\_where\_to\_go}(a, (x, y))$ 
    end for
    update_grid // Using Equation 4
    evaporate_grid // Using Equation 7
    update_neighbourhood_radio // decreases radio
end for

```

condition, each individual leaves a constant amount η of pheromone at the cell where it is located at every time step t . This pheromone decays at each time step at a rate k . Please note that there is no direct communication between the organisms but a type of indirect communication through the pheromone field.

3 Self-Organizing Ants Model

The proposed model has some common features with Chialvo and Millona's AS and Kohonen's SOM. The later inspired the model's name: *KohonAnts* (or KANTS). KANTS was designed as a clustering algorithm, in order to be capable of grouping a set of input (training) samples into clusters with similar features. In addition, it behaves as a classification algorithm, working in a non-supervised way, without considering the class of the input patterns during the process.

The main idea is to assign each input sample (a vector) to an ant, and then dropping it into an habitat designed as a toroidal $X \cdot Y$ grid. Then, the ants/vectors move around changing the environment. Every cell also contains a vector of the same dimension and range as the training set. Since every ant tends to move towards areas in the grid which hold vectors more similar to themselves (to their associated vectors), ant position and pheromone content co-adapt. This means that ants with similar data items tend to gather closer in the grid, and the area itself will contain similar vectors to those stored in the ants on top of them. Then, the grid can be used as a classification tool (as a SOM after training), while ants will be grouped in clusters of similar individuals. The model's pseudo-code is shown in Algorithm 1.

The function *DecideWheretoGo* is described here in detail. Transition probabilities are computed as follows:

$$P_{ij} = \begin{cases} \frac{W(\sigma_{ij})}{\sum_{u \in N_i^t} W(\sigma_{iu})} & \text{if } j \in N_i^t \\ 0 & \text{otherwise} \end{cases} \quad (2)$$

Algorithm 2. Decide_Where_To_Go ($a, (i, j)$)

```

for all cells  $(x, y)$  in neighbourhood of  $(i, j)$  do
     $\sigma_{ij,xy} = E_D((i, j), \text{centroid}((x, y)))$ 
    compute  $W(\sigma_{ij,xy})$  // Using Equation 1
end for
 $q = \text{random}(0,1)$ 
if  $q \leq q_0$  then
     $(k, l) = \text{MAX}(P_{ij,xy})$  // selected cell = the one with maximum probability
else
     $(k, l) = \text{roulette\_wheel}(P_{ij,xy})$  // selected cell = roulette_wheel
end if

```

where N_i is the neighbourhood of the cell i . There is also a *neighbourhood radius* (nr) which diminish along the run, meaning that the neighbourhood varies with t . Values σ are defined by:

$$\sigma_{ij} = \sqrt{V_i(v)^2 - CTR_j(v)^2} \quad \forall v = 1..nvars \quad (3)$$

where V_i is the vector associated to the cell i and CTR_j is the centroid of a zone centered in the cell j . The formula is equivalent to compute the Euclidean distance between the vector associated to the cell i and the centroid vector for the cell j , both vectors have a number of variables $nvars$. Finally, in the *DecideWheretoGo* rule, $W(\sigma)$ is the Ant System pheromone weighting function (Equation 1). The rule works as follows: when an ant is building a solution path and is placed at one node i , a random number q in $[0,1]$ is generated; if $q \leq q_0$ the best neighbour j is selected as the next node in the path. Otherwise, the ant chooses the neighbour by using a roulette wheel considering P_{ij} as probability for every feasible neighbour j (Equation 2).

The *UpdateGrid* function is similar to those performed by classical Ant Algorithms when depositing pheromone on the trails. At every step, each ant k updates its cell i , using a formula similar to SOMs' learning function [5]. Bearing in mind that every sample/ant and cell in the grid is a vector of $nvars$ variables, the formula is as follows:

$$V_i^t(v) = V_i^{t-1}(v) + R \cdot [a_k(v) - V_i^{t-1}(v)] \quad \forall v = 1..nvars \quad (4)$$

where V_i is the vector associated to the cell i , t is the current iteration, and a_k is the vector associated to the ant k . R is a kind of reinforcement rate:

$$R = \alpha \cdot (1 - \overline{D}(a_k, CTR_i)) \quad (5)$$

where α is the learning rate factor typical in SOM (which is constant in this algorithm), CTR_i is again the centroid of a zone centered in the cell i . Finally, \overline{D} is the mean Euclidean distance between the ant's vector and the centroid vector:

$$\overline{D} = \sum_{v=1}^{nvars} \frac{\sqrt{a_k(v)^2 - C_i(v)^2}}{nvars} \quad (6)$$

As in all the Ant Algorithms, it is very important that environment reverts to its previous (or initial) state. The evaporation in KANTS is performed in every cell once all the ants have moved and updated the environment.

$$V_i(v) = V_i(v) - \rho \cdot V_{i0}(v) \quad \forall v = 1..nvars \quad (7)$$

where ρ is the usual evaporation factor and V_{i0} is the initial vector associated to the cell i . The function changes the vector in order to be closer to its initial values. This can be interpreted as an evaporation of the trails in the environment.

4 Experiments and Results

The datasets used to test and validate the model are well-known real world databases. IRIS contains data of 3 species of iris plant (Iris Setosa, Versicolor and Virginica), 50 samples of each one and 4 numerical attributes (the sepal and petal lengths and widths in cms.). GLASS contains data from different types of glasses studied in criminology. There are 6 classes, 214 samples (unevenly distributed in classes) and 9 numerical features related to the chemical composition of the glass. PIMA (Pima Indians Diabetes database) contains data related to some patients and a class label representing their diabetes diagnostic according to the world-wide health organization's criterion. There are 768 samples with 8 numerical features (medical data).

For each of the databases, 3 sets were built by transforming the original into 3 disjoint sets of equal size. The original class distribution (before partitioning) is maintained within each set. Then 3 pairs of datasets 'training-test' are considered by splitting the 3 previous into half size ones; they are named *50tra-50tst*. In addition, 3 other pairs are created, but considering a distribution of 90% of samples for training and 10% for test. These sets are named *90tra-10tst*. In order to classify with KANTS, a parameter was introduced: the number of neighbours to compare with the test sample. This way, the algorithm searches for the K nearest vectors in the grid (using the Euclidean distance) to the vector correspondent to the sample which it wants to classify. It assigns the class of the majority. It is similar to the one used in K-Nearest Neighbours (KNN) method (see [8] for details), but in this case it is used once the grid has been trained (with the training dataset) and many times the algorithm works well even considering $K = 1$. Ten runs were made for each pair of datasets (training and test). Results are presented in Table 1.

Results are compared with those yielded using some techniques. Two statistical methods: the traditional deterministic *KNN* and the *Linear Discriminant Analysis* (LDA) [9], which determines if an instance is of a class or not using linear classification based on the covariance matrix. In addition, we consider the results yielded by a Neural Network for classification (the *Multilayer perceptron for classification problems with Conjugate Gradient based training* (MLPCG) [10]), which is a typical back-propagation algorithm [11] where the weights are adjusted by using a method (for non-linear optimization) that is called conjugate gradient. All these algorithms have been applied using the Keel Project.¹

¹ <http://www.keel.es/>

Table 1. Classification with Iris, Glass, Pima (6 different datasets each time)

IRIS Dataset	KANTS		KNN	LDA	MLPCG	
	Best	Mean	Best	Best	Best	Mean
50tra-50tst-Set1	98.67	98.00 \pm 0.67	97.30	97.00	97.00	95.00 \pm 2.22
50tra-50tst-Set2	98.67	97.60 \pm 0.53	96.00	99.00	95.00	93.00 \pm 1.06
50tra-50tst-Set3	100.00	98.80 \pm 0.40	94.60	97.00	96.00	94.00 \pm 0.80
90tra-10tst-Set1	100.00	100.00 \pm 0.00	100.00	100.00	100.00	99.00 \pm 2.10
90tra-10tst-Set2	100.00	99.33 \pm 2.00	93.33	100.00	100.00	100.00 \pm 0.00
90tra-10tst-Set3	100.00	100.00 \pm 0.00	93.33	100.00	100.00	100.00 \pm 0.00
GLASS Dataset	KANTS		KNN	LDA	MLPCG	
	Best	Mean	Best	Best	Best	Mean
50tra-50tst-Set1	68.22	65.42 \pm 1.62	62.60	65.00	33.00	33.00 \pm 0.00
50tra-50tst-Set2	67.29	64.86 \pm 1.52	64.40	61.00	33.00	33.00 \pm 0.00
50tra-50tst-Set3	74.77	71.03 \pm 2.17	64.40	60.00	33.00	33.00 \pm 0.00
90tra-10tst-Set1	69.57	65.65 \pm 1.30	47.80	52.00	30.00	30.00 \pm 0.00
90tra-10tst-Set2	73.91	73.48 \pm 1.30	60.80	48.00	30.00	30.00 \pm 0.00
90tra-10tst-Set3	91.30	83.48 \pm 3.25	82.60	65.00	33.00	31.00 \pm 1.04
PIMA Dataset	KANTS		KNN	LDA	MLPCG	
	Best	Mean	Best	Best	Best	Mean
50tra-50tst-Set1	75.52	74.32 \pm 0.61	70.03	78.00	76.00	70.00 \pm 2.02
50tra-50tst-Set2	77.34	76.61 \pm 0.58	71.80	77.00	73.00	71.00 \pm 1.61
50tra-50tst-Set3	77.60	75.13 \pm 0.85	72.90	77.00	77.00	72.00 \pm 2.61
90tra-10tst-Set1	83.12	80.52 \pm 1.42	64.90	76.00	75.00	67.00 \pm 3.08
90tra-10tst-Set2	79.22	75.32 \pm 1.42	73.60	77.00	79.00	75.00 \pm 2.12
90tra-10tst-Set3	84.42	80.65 \pm 2.05	70.10	77.00	81.00	77.00 \pm 2.79

Results obtained by KANTS are always better when compared with the rest of the methods, even in the comparison with a traditional clustering and classification approach such as KNN and with the MLPCG method which also yields very good results. Glass and Pima datasets usually obtain a low classification rate (both are difficult databases as the LDA classifications show), while KANTS achieves in some cases a rate 10% higher than MLPCG and KNN. It is important to comment that the running time of the algorithm is just a few seconds, depending on the dataset size: 8 seconds in Iris, 10 seconds in Glass and 20 seconds in Pima. The second best method (MLPCG) takes about 10 times more. All the experiments have been performed in a Pentium 1.6 GHz.

In [6], the authors performed a study on the distribution of ants in AS, with different configurations in the β - δ parameter space. Three types of behavior were observed when looking at the snapshots of the system after 1000 iterations: disorder, patches and trails. The results follow theoretical prediction: a second order phase transition is observed, when a region of the parameter space which gives rise to disorder regimes "turns into" a region where trails are formed. Moving away from the order-disorder line, the system loses its ability to evolve lines/trails of ants and patches gradually appear. In another experiment the system was tuned to a region in the parameter space where trails emerge, and, after the traffic network was formed, β was decreased in order to tune the system below the transition line; then, the ants started executing random walks and left their previously formed trails. Once β was set again to the initial value, the ants self-organized again on a similar traffic network.

A similar test was performed with KANTS. Parameters β and δ were varied, and the resulting ants' distribution after 100 iterations is depicted in Figure 1. (Parameters α , (nr) and (cr), were set to 1, 1 and 3, respectively.) It is not possible to

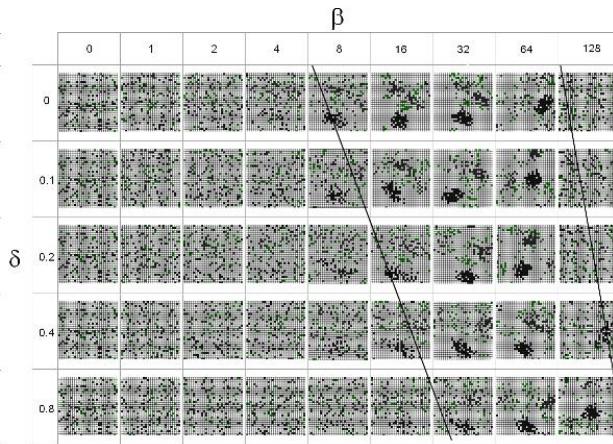


Fig. 1. Snapshots of the ants in the system after 100 iterations for different β and δ values. The straight lines roughly delimit the region where clusters emerge.

distinguish three different types of behavior, as in Chialvo and Millonas' experiments described above, but it is clear that there is a transition line from a disordered state, where ants/data do not cluster, and an ordered state where clusters emerge. Further away from the transition line, the model's ability to form clusters gradually decays. As in the original model, there is only a small region of the parameter space that gives rise to self-organization, but while AS forms trails, KANTS forms clusters that represent data samples.

Considering this results, KANTS appear to be also promising tool for data clustering. With proper tuning of β and δ , data represented by (and behaving as) ants form clusters that are easily distinguishable in the grid. Even if some kind of local search is necessary in order to tackle real-world problems, KANTS by now come forward as a core (and simple) model where hybridization may be performed in order to build algorithms to solve hard problems.

5 Conclusions and Future Work

This paper investigates KohonAnts, a new method for data classification, based on the hybridization of Ant Algorithms and Kohonen Self-Organizing Maps. The model turns n -variable data samples into artificial ants that evolve in a 2D toroidal grid paved with n -dimensional vectors. Data/Ants act on the habitat vectors by pushing the values towards their own. In addition, ants are attracted by regions where the vector values are closer to their own data. In this way, similar ants tend to aggregate in common regions of the grid. There is indirect communication between ants through the grid (stigmergy) leading, with a proper setting of the model's parameters, to the emergence of data clusters. In addition, pheromone deposition and evaporation creates a kind of cognitive field that can be used for classification purposes. The concept is very simple and naturally

inspired but the results obtained on classification are quite competitive when compared with traditional methods. As future lines of work, further tests will be performed on clustering, in order to properly evaluate KANTS's abilities on these kind of problems. In addition, a lot of enhancements are still possible in the original KANTS model presented in this paper. A neighbourhood function may be considered, similar to the one used in Self-Organizing Maps for updating the environment in a radius. And, as in [12], reproduction may improve speed and accurateness of the algorithm.

Acknowledgements. First author wishes to thank FCT, Ministério da Ciência e Tecnologia, his Research Fellowship SFRH/BD/18868/2004, also partially supported by FCT (ISR/IST plurianual funding) through the POS_Conhecimento Program that includes FEDER funds. This work has also been supported by NOHNES project from the Spanish Ministry of Science and Education (TIN2007-68083-C02-01).

References

1. Labroche, N., Monmarche, N., Venturini, G.: AntClust: Ant Clustering and Web Usage Mining. In: GECCO 2003. LNCS, vol. 2723, pp. 25–36. Springer, Heidelberg (2003)
2. Deneubourg, J., Goss, S., Franks, N., Sendova-Franks, A., Detrain, C., Chrétien, L.: The dynamics of collective sorting robot-like ants and ant-like robots. In: Proceedings of the first international conference on simulation of adaptive behavior on From animals to animats table of contents, pp. 356–363 (1991)
3. Bonabeau, E., Theraulaz, G., Fourcassié, V., Deneubourg, J.: Phase-ordering kinetics of cemetery organization in ants. Physical Review E 57(4), 4568–4571 (1998)
4. Ramos, V., Merelo, J.J.: Self-organized stigmergic document maps: Environment as a mechanism for context learning. In: Alba, E., Fernández, F., Gómez, J.A., Herrera, F., Hidalgo, J.I., Merelo-Guervós, J.J., Sánchez, J.M. (eds.) Actas primer congreso español algoritmos evolutivos, AEB 2002, pp. 284–293. Universidad de Extremadura (2002), <http://citeseer.nj.nec.com/ramos02selforganized.html>
5. Kohonen, T.: The Self-Organizing Maps. Springer, Heidelberg (2001)
6. Chialvo, D., Millonas, M.: How swarms build cognitive maps. In: The Biology and Technology of Intelligent Autonomous Agents. NATO ASI Series, vol. 144, pp. 439–450 (1995)
7. Wilson, E.: The Insect Societies. Belknam Press, Cambridge (1971)
8. Fix, E., Hodges, J.L.: Discriminatory analysis: Nonparametric discrimination: Consistency properties. In: International Statistical Review, vol. 57, pp. 238–247 (1989)
9. Friedman, J.H.: Regularized discriminant analysis. Journal of the American Statistical Association 84, 165–175 (1989)
10. Moller, F.: A scaled conjugate gradient algorithm for fast supervised learning. Neural Networks 6, 525–533 (1990)
11. Widrow, B., Lehr, M.: 30 years of adaptive neural networks: Perceptron, madaline, and backpropagation. In: Proceedings of the IEEE, vol. 78, pp. 1415–1442 (1990)
12. Fernandes, C., Ramos, V., Rosa, A.C.: Varying the population size of artificial foraging swarms on time varying landscapes. In: Duch, W., Kacprzyk, J., Oja, E., Zadrożny, S. (eds.) ICANN 2005. LNCS, vol. 3696, pp. 311–316. Springer, Heidelberg (2005)

Lattice Formation in Space for a Swarm of Pico Satellites

Carlo Pincioli¹, Mauro Birattari¹, Elio Tuci¹, Marco Dorigo¹,
Marco del Rey Zapatero², Tamas Vinko², and Dario Izzo²

¹ IRIDIA, CoDE, Université Libre de Bruxelles, Brussels, Belgium
`{cpinciro,mbiro,etuci,mdorigo}@ulb.ac.be`

² European Space Agency, Noordwijk, The Netherlands
`{marco.del.rey.zapatero,tamas.vinko,dario.izzo}@esa.int`

Abstract. We present a distributed control strategy that lets a swarm of satellites autonomously form a lattice in orbit around a planet. The system, based on the artificial potential field approach, proposes a novel way to split the artificial field in two main terms: a global artificial field that gathers the satellites around a predefined meeting point, and a local term that allows a satellite to place itself in the correct position relative to its closest neighbors. We apply the method to the problem of forming a two dimensional hexagonal lattice, using the well-known Lennard-Jones potential as local artificial field. The control parameters have been obtained with a genetic algorithm to maximize the precision of the formed lattice. The precision does not depend on the number of satellites and convergence is achieved from all initial distributions of the satellites.

1 Introduction

In this paper, we propose a control system that allows a swarm of small space-craft, called *pico satellites*, to build an hexagonal lattice in orbit around a planet. This is considered an important prerequisite for applications such as formation flight, coordinated observation [1,2], autonomous self-assembly of solar powered satellites [3], large antennas and large reflectors in space. The control system follows the principles of swarm intelligence: it is distributed and interactions among satellites are only local. Thanks to these characteristics, the system is highly scalable. Moreover, the system converges to the desired configuration for any initial distribution of the satellites. The validity of our results has been tested in simulations of up to 500 satellites. This paper builds on top of our preliminary work [4], in which we introduce the control system and we briefly study the precision of the lattice with control parameters chosen by hand in a flat space (no gravitational forces). In this paper, we optimize the control parameters for a real orbital environment and we study the precision of the formed lattice with a varying number of satellites and with different initial conditions.

The swarm is represented as a set of N identical point-masses. Initially, the satellites are randomly distributed in space under the gravitational influence of a near planet. A point \mathbf{p} orbiting around the planet is defined at design time as

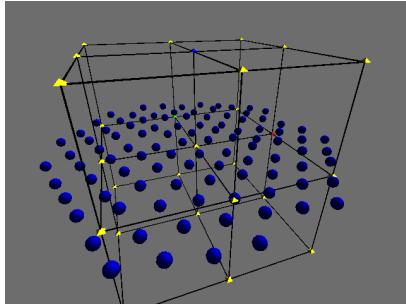


Fig. 1. An example of hexagonal lattice with 100 satellites

the origin of a reference frame. The control strategy we present in this paper lets the satellites position themselves around \mathbf{p} to form a regular hexagonal lattice located on the xy plane (see Figure 1). The satellites keep a mutual target distance σ which is a control parameter fixed at design time. In our simulations, the motion of a satellite has been modeled with the Hill's system of differential equations [5] assuming that the orbit of \mathbf{p} around the planet is circular. The satellites have a mass $m = 100$ kg and a thrusting capability $T_{max} = 50$ mN. The swarm has been tested in various orbital scenarios, reported in Table 2: geostationary orbits (GEO), low Earth orbits (LEO), and Jupiter orbits close to those of its satellites Amalthea, Metis and Io.

The main goals of the work are to ensure that satellites are not lost in space, that collisions are avoided and that the overall system is scalable, i.e., that the control strategy does not depend on the number N of satellites.

2 The Control Strategy

The control strategy studied in this work follows the artificial potential approach [6]. This idea has been first introduced for robot path planning [7] and proved effective also in satellite control problems [8]. The agent is imagined immersed in a virtual potential field calculated by its control system. The control action \mathbf{u} is the virtual force due to the virtual potential. The goal position of the agent corresponds to the status of minimum energy in the potential field. The artificial potential approach has been applied for formation control of wheeled robots. Balch and Arkin [9] proposed a system in which a small group of robots, uniquely identified, keep preassigned relative positions with a very simple attraction potential. Spears *et al.* [10] devised a distributed system in which robots form an hexagonal lattice through local interactions inspired by gravitational forces. This work neglects collision avoidance and assumes the robots to be initially very tightly distributed.

Orbit	ω (rad/s)	R (km)	T (s)
LEO	$1 \cdot 10^{-3}$	7,000	6,283
GEO	$7.3 \cdot 10^{-5}$	42,000	86,071
Amalthea	$1.5 \cdot 10^{-4}$	181,000	41,888
Metis	$2.5 \cdot 10^{-4}$	129,000	25,133
Io	$4.1 \cdot 10^{-5}$	421,600	153,248

Fig. 2. Different types of orbital environments considered. ω is the angular speed of rotation around the planet, R is the distance from its center and T is the time needed to complete one orbit.

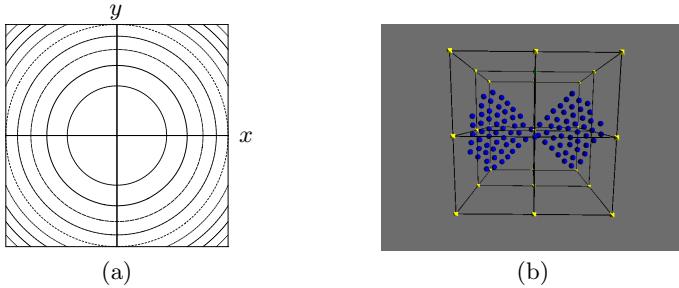


Fig. 3. (a) The equipotential curves of g on the xy plane (b) A butterfly shaped lattice obtained with a suitable \mathbf{g} potential

The features of the task considered in this paper suggested a novel definition of the virtual potential field. The control strategy \mathbf{u} has been expressed as the superposition of three contributions:

$$\mathbf{u} = \mathbf{g} + \mathbf{l} + \mathbf{d}, \quad (1)$$

where \mathbf{g} is a force that attracts each satellite towards the origin of the common reference frame, \mathbf{l} is a force that creates locally flat lattices with the neighboring satellites while avoiding in-swarm collisions, and \mathbf{d} is a damping factor analogous to viscosity, used to stabilize the behavior of the swarm and to ensure convergence.

2.1 Global Attraction to the Origin

We assume that the satellites know their position with respect to the reference frame defined by \mathbf{p} . This is not a stringent requirement in a space application because many well known techniques can be employed, spanning from the use of triangulation with fixed star positions to placing a special satellite in \mathbf{p} that broadcasts its position in space.

Defining \mathbf{q} as the position of a satellite with respect to \mathbf{p} , and defining the normalized vector $\bar{\mathbf{q}} = [\bar{q}_x \quad \bar{q}_y \quad \bar{q}_z]^T = \mathbf{q}/\|\mathbf{q}\|$, then the virtual force that attracts satellites towards the origin is defined as:

$$\mathbf{g} = -\eta \|\mathbf{q}\|^2 \bar{\mathbf{q}}, \quad (2)$$

where η is a design parameter. Thanks to virtual viscosity (term \mathbf{d} of Equation 1, see also Section 2.3), a satellite starting from any point in space converges, after some time, to the origin.

As shown in Figure 3a, sections of the potential that defines \mathbf{g} cut parallel to the xy plane are circle shaped. Therefore, the global shape of the swarm is a circle. Using a potential with a different section contour, it is possible to change the global shape of the formation. As an example, Figure 3b depicts a butterfly shape obtained with a different \mathbf{g} .

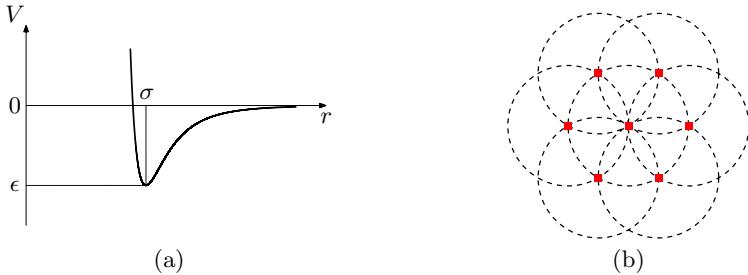


Fig. 4. (a) The Lennard-Jones potential (b) Its equilibrium state, an hexagonal lattice

2.2 Local Lattice Formation

The local potential field lets a satellite interact with its neighbors to create a lattice, while avoiding collisions. In this work, the local potential is inspired by a simple and very well known model of molecular interaction, the Lennard-Jones potential [11]:

$$V(r) = \epsilon \left[\left(\frac{\sigma}{r} \right)^{12} - 2 \left(\frac{\sigma}{r} \right)^6 \right], \quad (3)$$

r being the distance between two molecules. The force $\mathbf{F}(r)$ between two molecules is given by

$$\mathbf{F}(r) = -\nabla V(r) = -\frac{d}{dr} V(r) \hat{\mathbf{r}}, \quad (4)$$

where $\hat{\mathbf{r}}$ is a normalized vector directed as the line going from the center of the first molecule to the center of the second. This force is null when the distance coincides with the target distance σ ; the force is increasingly repulsive as $r < \sigma$ decreases; the force is attractive when $r > \sigma$. As Figure 4a shows, the attraction is very strong when r is not much larger than σ , but after a certain distance this force fades to zero, thus explaining the reason why we call this potential *local*. The stable arrangement of two molecules is such that they keep the mutual target distance σ (in our experiments, $\sigma = 300$ m). Increasing the number of molecules, the stable arrangement is an hexagon (see Figure 4b).

The design parameters of the potential are few and very intuitive to set: σ is the mutual distance among the satellites in the lattice, while ϵ is the depth of the potential well, which accounts for the attractiveness and stability of the minimum located at distance σ . Notably, the lattice is formed on the basis of positional information only: no communication is needed.

From Equations 3 and 4, the magnitude of the virtual force parallel to the xy plane between a satellite and its i -th neighbor is given by

$$l_i^{xy} = -\frac{d}{dr} V(r) = \frac{12\epsilon}{r} \left[\left(\frac{\sigma}{r} \right)^{12} - \left(\frac{\sigma}{r} \right)^6 \right].$$

To flatten the lattice, a virtual force parallel to the z axis is defined as $l_i^z = -\psi \text{sign}(r_z) r_z^2$, where r_z is the projection of $\mathbf{r} = r\hat{\mathbf{r}}$ on the z axis. The force

is then $\mathbf{l}_i = [l_i^{xy} \bar{q}_x \quad l_i^{xy} \bar{q}_y \quad l_i^z]^T$. Eventually, \mathbf{l} is defined as the average of the virtual forces due to the M closest neighbors (in our experiments, $M = 6$):

$$\mathbf{l} = \frac{1}{M} \sum_{i=1}^M \mathbf{l}_i.$$

It is possible to control the shape of the local lattice by using a different potential. A natural choice is a molecular model already studied in crystallography. In other terms, the work presented here suggests a link between crystallography and lattice formation in robotics.

2.3 Ensuring Convergence

The virtual forces \mathbf{g} and \mathbf{l} are defined by conservative fields. This means that convergence is impossible without a dissipative term. To obtain convergence, we imagine that the satellites are immersed in a viscous medium. Thus, the expression of \mathbf{d} is analogous to viscosity: $\mathbf{d} = -\xi \dot{\mathbf{q}}$, where ξ is a design parameter, usually smaller than 0.2.

2.4 Formation Stabilization After Convergence

Experiments revealed that once the swarm converges to its final configuration, the satellites oscillate around their equilibrium points, thus wasting propellant. A solution to this problem is increasing the damping factor ξ after the final configuration has been reached:

$$\dot{\xi} = \begin{cases} \xi_{conv} e^{-\xi/2} & \text{if } \xi < \xi_{stab}, \\ 0 & \text{otherwise.} \end{cases}$$

The value of ξ_{conv} is the one that ensures convergence when satellites form the lattice (see Section 2.3). The value ξ_{stab} for which oscillations disappear depends on the orbit at which \mathbf{p} is located. In our experiments $\xi_{stab} = 0.7$.

A further problem is when to trigger the stabilization. Currently we adopt a simple time-based criteria. Each satellite individually measures the time elapsed since the beginning of the shape formation process. After a certain time, stabilization is triggered. A more elegant method would be to trigger the stabilization with a distributed consensus algorithm [12].

3 Results

Experimental evaluation shows that even with suboptimal parameters the system works reasonably well, although good parameters for an orbital environment are not equally good for another [4]. Here, we optimize the parameters to minimize positioning errors in the lattice. With these parameters, we study scalability. Finally, we test the dependence of the control system on initial conditions (placement of satellites).

Parameter	Value
Number of generations	1000
Population size	50
Mutation probability	0.2
Crossover probability	0.9
Elitism	<i>the best survives</i>

Fig. 5. Parameters of the genetic algorithm employed for setting the control parameters

Parameter	Value
η	$1.6295 \cdot 10^8$
ψ	$5.96201 \cdot 10^8$
ϵ	$4.5332 \cdot 10^4$
ξ	0.165984

Fig. 6. Values of the control parameters obtained via the genetic algorithm

3.1 Optimizing the Control Parameters

Good values of some control parameters, such as η , ψ , ϵ and ξ , are not easy to find. We chose to optimize them with Goldberg's simple genetic algorithm [13]. Table 5 summarizes the parameter values used for the genetic algorithm.

Evolutions were performed with 10 satellites in a GEO environment. The trials lasted 1000 time steps, each time step being 12.5 s long. The placement of a satellite has been evaluated as follows:

$$\chi_i = \frac{1}{N_i} \sum_{j \in \mathcal{N}_i} \frac{|\sigma - r_{ij}|}{\sigma}$$

where \mathcal{N}_i is the set containing the N_i closest neighbors of satellite i and r_{ij} is the relative distance between the satellites i and j at the final lattice acquisition time. The genetic algorithm minimizes the worst satellite placement, defined as $\chi = \max_i \chi_i$. The best control parameters that we obtained are reported in Table 6. They yield a score $\chi = 0.012842$, which corresponds to a positioning error of 3.85 m ($\sigma = 300$ m).

3.2 Scalability

Scalability makes it possible to optimize the parameters with a minimal number of satellites, thus finding quickly a convenient setup. Figure 7 reports the behavior of the placement error for different numbers of satellites. The placement error is calculated as $\bar{\chi} = \frac{1}{N} \sum_{i=1}^N \chi_i$. Although the parameters were obtained through trials involving only 10 satellites, $\bar{\chi}$ remains practically constant around the value 0.02 (that corresponds to 6 m), with a minimum of 0.007 (2.1 m) and a maximum of 0.035 (10.5 m). Only with 500 satellites the maximum error is slightly larger: 0.088 (26.4 m).

3.3 Initial Conditions

Convergence to the final structure can be mathematically proven by the presence of the global attractor located at the origin of the virtual global field and by the known results about the Lennard-Jones potential.

Table 8 shows the results of a set of experiments run to test if $\bar{\chi}$ is affected by the initial spatial distribution of the swarm. In the *centered cubic* distribution,

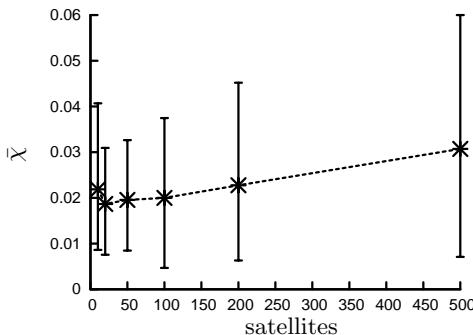


Fig. 7. Average placement error for a different number of satellites

the satellites are placed uniformly in a cube with side of 6 km and centered around the origin. The *centered spheric* distribution is a hollow sphere centered around the origin with radius 3 km and 300 m thick. The *shifted* distributions are centered in point [3 3 3] (coordinates in km). For all the experiments, the same experimental conditions described in Section 3.1 have been used with swarms of 100 satellites. The results show that $\bar{\chi}$ has values similar to those found for the scalability tests.

4 Conclusions

The presented work deals with a decentralized control strategy for swarms of satellites that allows them to autonomously form a bi-dimensional hexagonal lattice under the gravitational influence of a near planet. The method is based on the artificial potential field approach. In the paper, a novel way of defining the potential is proposed. This method allows the designer to split the problem of forming the lattice into two more intuitive subproblems: an artificial field attracts globally the satellites towards a meeting point and controls the shape of the formation; another artificial field takes care of defining the interactions among the satellites to form local lattices. In this work, the Lennard-Jones potential has been used to define the local field. The control parameters to be set by the designer are few and very intuitive, and acceptable results can be obtained even by setting the parameters by hand. We have optimized the control parameters to minimize the placement error and results show that such error is independent of the number of satellites and of the initial spatial distribution of the swarm.

The way here proposed to define the artificial potential field suggests a possible link between lattice formation in robotics and known results in crystallography. We plan to further study this link by trying other potentials that are known in the literature.

Acknowledgments. This research was funded by the European Space Agency under the Ariadna scheme. It was also partially supported by COMP2SYS and

Distribution	$\bar{\chi}$
Centered Cubic	0.0215913
Centered Spheric	0.0199159
Shifted Cubic	0.0207984
Shifted Spheric	0.0191019

Fig. 8. Placement error $\bar{\chi}$ obtained with different initial spatial distributions

by the ANTS project. COMP2SYS is a Marie Curie Early Stage Training Site funded by the European Commission under contract MEST-CT-2004-505079. ANTS is an *Action de Recherche Concertée* funded by the French Community of Belgium. Mauro Birattari and Marco Dorigo acknowledge support from the Belgian F.R.S.-FNRS.

References

1. Curtis, S., Mica, J., Nuth, J., Marr, G., Rilee, M., Bhat, M.: ANTS (Autonomous Nano-Technology Swarm): An artificial intelligence approach to asteroid belt resource exploration. In: International Astronautical Federation, 51th Congress (2000)
2. D'Arrigo, P., Santandrea, S.: The ARIES mission. ASTRIUM Ltd./ESA-ESTEC Feasibility Study A0/1-3846/02/NL/JA, Stevenage, UK (2004)
3. Mori, M., Nagayama, H., Saito, Y., Matsumoto, H.: Summary of studies on space solar power systems of the national space development agency of Japan. *Acta Astronautica* 54(5), 337–345 (2004)
4. Pincioli, C., Birattari, M., Tuci, E., Dorigo, M., del Rey Zapatero, M., Vinko, T., Izzo, D.: Self-organizing and scalable shape formation for a swarm of pico satellites. In: Proceedings of the NASA/ESA Conference on Adaptive Hardware and Systems (AHS-2008). IEEE Computer Society Press, Washington (in press, 2008)
5. Clohessey, W., Wiltshire, R.: Terminal guidance systems for satellite rendez-vous. *Journal of the Aerospace Sciences* 27(9), 653–658 (1960)
6. Khatib, O.: Real-time obstacle avoidance for manipulators and mobile robots. *International Journal of Robotics Research* 5(1), 90–98 (1986)
7. Ge, S.S., Cui, Y.I.: Dynamic motion planning for mobile robots using potential field method. *Autonomous Robots* 13(3), 207–222 (2002)
8. Badawy, A., McInnes, C.: On-orbit assembly using superquadric potential fields. *Journal of Guidance, Control, and Dynamics* 31(1), 30–43 (2008)
9. Balch, T., Arkin, R.C.: Motor schema-based formation control for multiagent robot teams. In: Lesser, V., Gasser, L. (eds.) *Proceedings of the First International Conference on Multiagent Systems (ICMAS 1995)*, pp. 10–16. AAAI Press, San Francisco (1995)
10. Spears, W., Spears, D., Hamann, J., Heil, R.: Distributed, physics-based control of swarms of vehicles. *Autonomous Robots* 17(2-3) (2004)
11. Kittel, C.: *Introduction to Solid State Physics*. Wiley, New York (1986)
12. Lamport, L.: Lower bounds on asynchronous consensus. In: Schiper, A., Shvartsman, A.A., Weatherspoon, H., Zhao, B.Y. (eds.) *Future Directions in Distributed Computing*. LNCS, vol. 2584, pp. 22–23. Springer, Heidelberg (2003)
13. Goldberg, D.E.: *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley, Boston (1989)

Merging Groups for the Exploration of Complex State Spaces in the CPSO Approach

Stefanie Thiem, Jörg Lässig, and Peter Köchel

Department of Computer Science, Chemnitz University of Technology
Chemnitz, Germany

{stefanie.thiem,joerg.laessig,peter.koechel}@cs.tu-chemnitz.de

Abstract. In recent years many investigations have shown that *Particle Swarm Optimization* (PSO) is a very competitive global optimization heuristic. However, in very complex state spaces the classical PSO algorithm converges too fast and hence provides only suboptimal results. Looking at swarm robotics it seems natural to adopt a repulsive force to avoid this undesired behavior as suggested in Charged PSO but the downside of this is the problem of final convergence in static applications.

The contribution of this paper is to introduce a dynamic *charge reduction* over time defining particle groups which are iteratively merged, reducing the number of charged particles during the optimization run.

A visualization of this process shows spontaneous formation of independent particle groups, redolent very much of swarm movement in nature. Optimization results are superior compared to other PSO approaches especially in very complex high dimensional search spaces.

1 Introduction

Particle Swarm Optimization (PSO), invented by Kennedy and Eberhart in 1995 [1], has shown to be competitive compared to other global optimization heuristics as e.g. *Genetic Algorithms* or stochastic approaches as *Simulated Annealing* or *Threshold Accepting* [2]. In recent years, many different versions and additional heuristics have been introduced.

The general idea of PSO is the movement of each particle partly in the direction of its current velocity, in the direction of its local best solution (cognitive component) and to the so far global optimum (social component) in each iteration. In this paper a fourth repulsive force component is added, preventing early convergence as suggested by Blackwell *et al.* with *Charged PSO* (CPSO) [3]. Besides strongly positive effects as superior results for instances with complex state spaces or high dimensionality one faces now the problem to encompass final convergence if applied to static problems. We solve this by introducing a dynamic *charge reduction* over time defining *particle groups* which are iteratively merged, reducing the overall charge of the system gradually during the optimization run.

A merging of groups of particles *deterministically* or by *random* according to different schemes is considered. Repulsive terms are applied only between particles of different groups. Then the dynamic reduction of the number of groups

eventually conducts convergence to classical PSO and hence an excellent convergence of the algorithm itself. We dub the mentioned deterministic and random approaches *Deterministic CPSO* and *Random CPSO*, respectively (D/RCPSO).

Details of both approaches are introduced in the remaining sections, organized as follows: In the next section classical PSO is introduced formally and different repulsive components are discussed. Section 3 considers the new approaches D/RCPSO stating also theoretical properties. Section 4 focuses on experimental results applying D/RCPSO to standard continuous test functions in high dimensions. It is shown that our new approach provides results substantially better compared to classical PSO variants in many cases. In chapter 5 the results are briefly summarized and an outlook on future work is given as well.

2 The Theoretical Framework

The optimization process is represented by the movement of the particles, i.e. the change rate of the velocity and the position of each one. A commonly used PSO and very competitive version is based on the following equations of motion for a particle i and a discrete time parameter t [4]

$$\begin{aligned}\mathbf{v}_i^{t+1} &= w [\mathbf{v}_i^t + c_1 \cdot \mathbf{R}_1 \cdot (\mathbf{l}_i^t - \mathbf{r}_i^t) + c_2 \cdot \mathbf{R}_2 \cdot (\mathbf{g}^t - \mathbf{r}_i^t)] + \mathbf{a}_i^t \\ \mathbf{r}_i^{t+1} &= \mathbf{r}_i^t + \mathbf{v}_i^t.\end{aligned}\quad (1)$$

The diagonal matrices \mathbf{R}_1 and \mathbf{R}_2 contain uniform random numbers and thus randomly weight each component of the connecting vector from the current position \mathbf{r}_i to the local position \mathbf{l}_i respectively global optimum \mathbf{g} . The new position is obtained by adding the velocity to the current position.

2.1 Common PSO Version

In the classical model $\mathbf{a}_i^t = 0$ and the weight w is given by

$$w = \frac{2}{|2 - c - \sqrt{c^2 - 4c}|} \quad \text{with } c = c_1 + c_2. \quad (2)$$

Often a disadvantage of this approach is the fast convergence behavior preventing the algorithm to escape from local minima later on in the optimization process. In particular, a known result is that the particles quickly converge to a small area of the search space with a rate of shrinkage following a scaling law for spherical symmetric functions [5]. Comparing Equ. (1) with the equations of motion for swarm models one can identify four different force/acceleration components [2]:

- an *attractive force* in direction of the global best \mathbf{g} and local best \mathbf{l}_i ,
- a *stochastic component* to prevent the particles from early convergence,
- a velocity dependent *friction* term influenced by the weight factor w and
- a *repulsive force* \mathbf{a}_i^t to avoid obstacles and collisions in swarm models [6],

where the acceleration term \mathbf{a}_i^t is unused so far. *Random PSO* (RPSO) and CPSO introduce this concept to swarm optimization. RPSO replaces the attraction to the global minimum with a repulsive component depending linearly on the distance of the current particle and a random other particle [7]. One disadvantage is that the repulsive force increases with distance, which seems to be unnatural. In CPSO the repulsive term is based on the well-known COULOMB law [3], which is also similar to the commonly used gravitational force in swarm robotics [6].

2.2 Charged Particle Swarm Optimization

The CPSO approach is based on the already introduced PSO version of Equ. (1), where the acceleration of a particle i is then determined additively from the repulsive force term between particle i and k by

$$\mathbf{a}_i = \sum_k \mathbf{a}_i(k). \quad (3)$$

The force itself is given by the COULOMB law as already mentioned above depending on the charges Q_i of the particle and assuming a particle mass $m_i = 1$. In order to prevent divergence and reduce computational efforts, the force law is restricted for small distances with r_c and large ones with r_p [3] and is given by

$$\mathbf{a}_i(k) = \begin{cases} \frac{Q_i Q_k}{|\mathbf{r}_i - \mathbf{r}_k|^3} (\mathbf{r}_i - \mathbf{r}_k) & r_c \leq |\mathbf{r}_i - \mathbf{r}_k| \leq r_p \\ \frac{Q_i Q_k}{r_c^2 \cdot |\mathbf{r}_i - \mathbf{r}_k|} (\mathbf{r}_i - \mathbf{r}_k) & |\mathbf{r}_i - \mathbf{r}_k| \leq r_c \\ 0 & |\mathbf{r}_i - \mathbf{r}_k| > r_p \end{cases}. \quad (4)$$

One is still free to choose the charge of each particle, so that there are several versions of this approach possible. If all particles are charged, we call it a *charged swarm*. But, if only a certain percentage of the particles are charged and the others remain neutral, this is called an *atomic swarm* because the charged particles move in a sphere around the core built from the neutral particles [8]. And clearly when all particles are neutral, we again have the classical PSO.

3 CPSO with Random/ Deterministic Charging

With the introduction of a repulsive force we face the problem of final convergence. There are variants of the CPSO algorithm with groups of charged and uncharged particles but if a particle i is assigned a charge $Q_i = 1$ within these approaches, it keeps this for all times. The idea now is to modify the number of repelling particles with time by reorganizing particles in groups.

On algorithm start, the group number G_0 is defined to equal to the number of particles P . Only particles which are in different groups repel each other, which is an invariant in all iterations of the algorithm. So this means all particles are charged and the behavior is just like in CPSO as explained above. If we sum up all pairs which repel each other to a variable S (*charge sum*), we have

$$S_0 = \binom{G_0}{2} = \binom{P}{2} = \frac{P^2 - P}{2}. \quad (5)$$

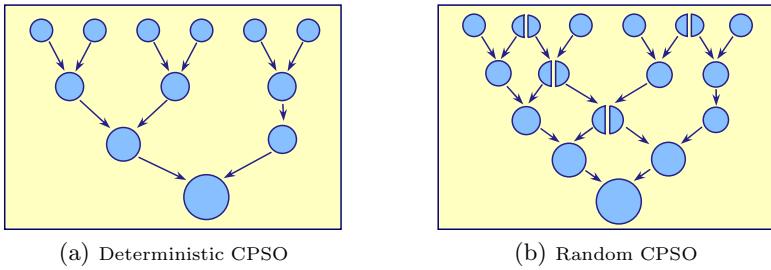


Fig. 1. Visualization of Grouping and Reorganization

The subscripts of S_i and C_i count the number of reorganizations, where with each reorganization the number of groups is reduced gradually. A straightforward approach is to unify pairs of groups in each iteration. If two groups are merged in this early phase then the number of groups is $P - 1$ and we have $S_1 = S_0 - 1$. Finally after n iterations, all particles end up in one single group and $S_n = 0$.

3.1 Deterministic Grouping

Basically, we distinguish between two merging methods, i.e. *deterministic* and *random* grouping. *Deterministic grouping* denotes merging processes where only complete groups are merged and single particles cannot change the group belonging. *Random grouping* allows the arbitrary reorganization of groups like breaking up one group and assigning the particles to different groups, see Fig. 1.

There are also different ways to realize *deterministic grouping* like for instance the merging of the two smallest groups in a *merging iteration*. Another possibility is to perform several merges in one *merging iteration* i such as the merging of $\lfloor G_i/2 \rfloor$ pairs of groups, further denoted as *exponential merging*, because the number of groups G_i shrinks exponentially in the number of iterations.

3.2 Random Grouping

Instead of only deterministically finding the group belongings of the particles, e.g. different groups are just collapsed into one group, this version allows more general randomized reorganization schemes, see Table 1 and Algorithm 1.

Table 1. Random grouping for a swarm with ten particles using *integer division*

The number of *merging iterations* in *random grouping* is $P - 1$, caused intrinsically by the behavior of the *integer division* and *modulo* operators. Considering the modulo operator, the group number is reduced by exactly one in each merging iteration, which results in an asymmetric reduction of S_i comparable to deterministic *exponential merging*. But, in the integer division case the number of groups in each iteration is not reduced exactly by one (Table 1). In the first iterations the number of groups decreases faster but afterwards the group number keeps constant over many iterations, which results in a *quasi constant merging speed*. While the average charge sum reduction in each merging iteration is

$$\binom{P}{2} / (P - 1) = \frac{P \cdot (P - 1)}{2 \cdot (P - 1)} = \frac{P}{2},$$

due to the fixed number of steps, the largest possible charge sum reduction in one merging operation is $P - 1$, which is only twice this value. Further, every reduction scheme would have at least one step of size $P - 1$. Furthermore, it is computationally very efficient to reduce grouping to a few integer operations.

3.3 Simplified Force Law

Additionally, in our experiments we noticed that the complicated force law of Equ. (4) can be simplified without loosing the good performance results. One possibility is the usage of the exponential function for the force, which does not diverge for small particle distances and is motivated from the LENNARD-JONES-potential. The force term does not include charges anymore but is again only active when particles i and k are in different groups. It reads

$$\mathbf{a}_i(k) = \exp^{-|\mathbf{r}_i - \mathbf{r}_k|} (\mathbf{r}_i - \mathbf{r}_k) . \quad (6)$$

Unfortunately, the computation of the exponential function is very time consuming, but we obtained equally competitive performance for the simple COULOMB

Algorithm 1. Position / Velocity Update Rule in Random CPSO.

```

Require: Position  $r_i$ , velocity  $v_i$  and local best position  $l_i$  for each particle  $i$   

         global best position  $g$  and cognitive parameter  $c_1$  and social parameter  $c_2$ 
Ensure: New position  $r_i$  and velocity  $v_i$  for each particle
1:  $c \leftarrow c_1 + c_2$ ,  $w \leftarrow 2/|2 - c - \sqrt{c^2 - 4c}|$ ,  $t \leftarrow 1$ 
2: for all particles  $i$  do
3:    $a_i \leftarrow 0$ 
4:   for  $k = i + t$  to  $k < i + P$  with  $k = k + t$  do {equivalent to  $i \text{ div } t \neq k \text{ div } t$ }
5:      $a_i \leftarrow a_i + a_i(k \bmod P)$                                 {see equation (3) and (4)}
6:   for all dimensions  $d$  do
7:      $r_1, r_2 \leftarrow \text{get\_uniform\_random\_number}(0, 1)$ 
8:      $v_{id} \leftarrow w \cdot (v_{id} + c_1 \cdot r_1 \cdot (l_{id} - x_{id}) + c_2 \cdot r_2 \cdot (g_d - x_{id}) + a_{id})$ 
9:      $x_{id} \leftarrow x_{id} + v_{id}$ 
10:    if ( $t < N \wedge \text{fixed\_iterations\_reached}()$ ) then  $t \leftarrow t + 1$ 
11: return  $\{r_i\}, \{v_i\}$ 

```

force law. The basic advantage is the abolishment of the two parameters r_c and r_p by taking the first case of Equ. (4) unbiased. Another approach would be to use a probability distribution for the force law as used in *Quantum PSO* [8].

4 Parameter Setup and Experimental Results

The different PSO algorithms are applied to a selection of test functions, summarized in Table 2. The parameter setup is oriented at [10] with $c_1 = 2.8$, $c_2 = 1.3$, which outperforms the standard setup with $c_1 = c_2 = 2.05$ already. The initial positions are chosen randomly in the solution space and the initial velocities here are 0. As swarm size $N = 50$ in 30 dimensions and $N = 100$ in 50 dimensions are used. For the charged variants we assume a charge of $Q_i = 1$ for charged particles and 0 otherwise. For CPSO half of the particles carry a charge and the other ones are neutral and we use the fixed values $r_c = 0.5$ and $r_p = 3$. In DCPSO we use exponential merging and in RCPSO the explained integer division scheme.

All results for the selected test functions are summarized in Table 3, where the average solutions for ten runs are given. The optimization run terminates when the global optimum is approximated by an additive error of 10^{-6} or at most 1,000 iterations (50,000 function evaluations) in 30 dimensions and 6,000 iterations (600,000 function evaluations) in 50 dimension are done. In the cases where the global optimum is found, the number of such runs is given in brackets.

Experimental results show that the achieved diversification of search behavior and delayed convergence evolves better solutions especially in high dimensions. This has been shown exemplary on the RASTRIGIN and ACKLEY function in 50 dimensions, where D/RCPSO provided final minimization values about half of the value compared to conventional PSO. For less difficult search spaces, represented by less demanding functions as the Sphere or GRIEWANGK function, this advantage over other methods reduces gradually with the problem complexity, showing the same behavior for D/RCPSO and CPSO in rather simple domains.

Table 2. Selection of N-dimensional functions [9]

Name	Function Definition	Domain and Minimum
Ackley Function	$f(\mathbf{x}) = -20 \exp\left[-\frac{1}{5} \sqrt{\frac{1}{N} \sum_{i=1}^N x_i^2}\right] - \exp\left[\frac{1}{N} \sum_{i=1}^N \cos(2\pi x_i)\right] + 20 + e$	$x_i \in [-32.768, 32.768]$ Min. Pos.: $x_i^* = 0$ Min. Value: $f(\mathbf{x}^*) = 0$
Griewangk Function	$f(\mathbf{x}) = 1 + \frac{1}{4000} \sum_{i=1}^N x_i^2 - \prod_{i=1}^N \cos\left(\frac{x_i}{\sqrt{i}}\right)$	$x_i \in [-600.0, 600.0]$ Min. Pos.: $x_i^* = 0$ Min. Value: $f(\mathbf{x}^*) = 0$
Rastrigin Function	$f(\mathbf{x}) = 10N + \sum_{i=1}^N [x_i^2 - 10 \cos(2\pi x_i)]$	$x_i \in [-5.12, 5.12]$ Min. Pos.: $x_i^* = 0$ Min. Value: $f(\mathbf{x}^*) = 0$
Rosenbrock Function	$f(\mathbf{x}) = \sum_{i=1}^{N-1} \left[100 (x_i^2 - x_{i+1})^2 + (1 - x_i)^2 \right]$	$x_i \in [-2.048, 2.048]$ Min. Pos.: $x_i^* = 1$ Min. Value: $f(\mathbf{x}^*) = 0$
Sphere Function	$f(\mathbf{x}) = \sum_{i=1}^N x_i^2$	$x_i \in [-5.12, 5.12]$ Min. Pos.: $x_i^* = 0$ Min. Value: $f(\mathbf{x}^*) = 0$
Step Function	$f(\mathbf{x}) = \sum_{i=1}^N \lfloor x_i \rfloor + 6N$	$x_i \in [-5.12, 5.12]$ Min. Pos.: $x_i^* \in [-5.12, -5.0]$ Min. Value: $f(\mathbf{x}^*) = 0$

Table 3. Performance of PSO in 30 and 50 dimensions

D	Function	PSO	CPSO	DCPSO	RCPSO
30	Ackley	1.045803 (3)	1.413774	0.000006 (9)	0.000000 (10)
	Griewangk	0.023132 (2)	0.010601 (2)	0.013776 (3)	0.012546 (4)
	Rastrigin	40.196296	40.0596210	37.722015	38.504870
	Rosenbrock	22.564743	29.375666	23.496527	23.429378
	Sphere	0.000000 (10)	0.000077 (2)	0.000000 (10)	0.000000 (10)
	Step	8.500000	14.300000	9.800000	7.500000
50	Ackley	2.271681	1.833939	0.268883	0.000012 (4)
	Griewangk	0.028530 (1)	0.028230	0.017136 (4)	0.024520 (3)
	Rastrigin	99.923959	100.590839	71.636957	55.877123
	Rosenbrock	42.442448	69.775139	40.9471946	41.389139
	Sphere	0.000000 (10)	0.000053	0.000000 (10)	0.000000 (10)
	Step	15.300000	19.800000	6.600000	3.000000 (1)

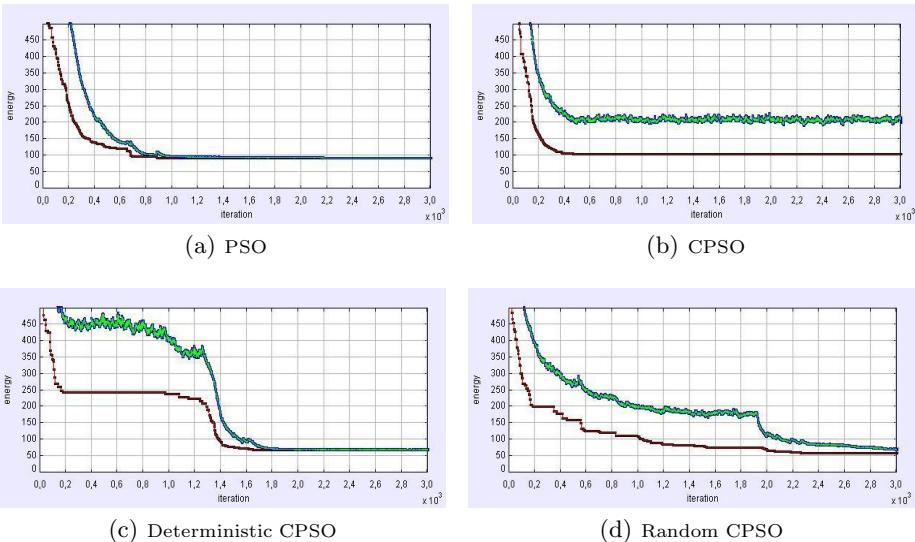


Fig. 2. Convergence Behavior: red - best so far value, green - current value average

Then, the predominant comparison criterion between different approaches is the running time evoked by the different convergence behavior of the different methods. It should be clear that D/RCPSO faces disadvantages in this respect because it is designed to search the state space more in detail before converging. In some cases RCPSO performs better than DCPSO, probably due to a more symmetric charge sum reduction as discussed in section 3.

5 Conclusion

In the paper on hand we introduced a new approach to CPSO. The two considered methods D/RCPSO showed spontaneously self-organizing behavior as observable in natural swarms. During the search process the groups are reorganized incrementally which courses a dynamic convergence to classical PSO. Both

methods provided optimization results for various benchmark functions significantly better compared to classical PSO or CPSO, where RCPSO performed for some instances better than DCPSO.

This gives also hints for future directions in research. Also for DCPSO different other more symmetric merging schedules are imaginable, choosing only two groups per merging iteration or choosing the delay between merging iterations based on the charge reduction. It is also possible to apply different strategies to choose merging groups, considering e.g. group size or spatial proximity. The advantage of the applied schemes is their efficient implementation and RCPSO shows already a good symmetry for charge reduction. More general it would be interesting to introduce some kind of adaptive behavior. Ideally, the method converges relatively fast to classical PSO started on straight forward problems but searches very involved state spaces more in deep before convergence is obtained.

References

1. Kennedy, J., Eberhart, R.: Particle Swarm Optimization. In: Proceedings of IEEE International Conference on Neural Networks, vol. 4, pp. 1942–1948 (1995)
2. Thiem, S.: Swarmintelligence - Simulation, Optimization and Comparative Analysis. Diplom Thesis, Chemnitz University of Technology (2008)
3. Blackwell, M.: Particle Swarms and Population Diversity. *Soft Computing* 9(11), 793–802 (2005)
4. Shi, Y., Eberhart, R.: A Modified Particle Swarm Optimizer. In: Proceedings of the IEEE International Conference on Evolutionary Computation, pp. 69–73 (1998)
5. Clerc, M., Kennedy, J.: The Particle Swarm - Explosion, Stability, and Convergence in a Multidimensional Complex Space. *IEEE Transactions on Evolutionary Computation* 6(1), 58–73 (2002)
6. Spears, W.M., Spears, D.F., Hamann, J.C., Heil, R.: Distributed, Physics-Based Control of Swarms of Vehicles. *Autonomous Robots* 17(2-3), 137–162 (2004)
7. Urfalioglu, O.: Robust Estimation of Camera Rotation, Translation and Focal Length at High Outlier Rates. In: Proceedings of the First Canadian Conference on Computer and Robot Vision, pp. 464–471 (2004)
8. Blackwell, T.M.: Particle Swarm Optimization in Dynamic Environments (2008), <http://igor.gold.ac.uk/mas01tb/papers/PSOdynenv.pdf>
9. Hedar, A.R.: Test Functions (2007), http://www-optima.amp.i.kyoto-u.ac.jp/member/student/hedar/Hedar_files/TestG_files/Page364.htm
10. Carlisle, A., Dozier, G.: An off-the-Shelf PSO. In: Proceedings of the Workshop on Particle Swarm Optimization, vol. 1, pp. 1–6 (2001)

Parallel Ant Colony Optimization for the Quadratic Assignment Problems with Symmetric Multi Processing

Shigeyoshi Tsutsui

Hannan University, Matsubara, Osaka, Japan
tsutsui@hannan-u.ac.jp

Abstract. Recently symmetric multi processing (SMP) has become available at a reasonable cost. In this paper, we propose several types of parallel ACO algorithms with SMP for solving the quadratic assignment problem (QAP). These models include the master-slave models and the island models. We evaluated each parallel algorithm with a condition that the run time for each parallel algorithm and the base sequential algorithm are the same. The results suggest that using the master-slave model with increased iteration of ACO algorithms is promising in solving QAPs.

1 Introduction

Recently, microprocessor vendors supply CPUs which have multiple cores of 2, 4, or more, and PCs or WSs which use such CPUs are available at a reasonable cost. They are normally configured with symmetric multi processing (SMP) architecture. Since the main memory is shared among processors in SMP, parallel processing can be performed efficiently with less communication overhead among processors. Thus, we can say that computing platforms running parallel algorithms using SMP to get higher performance are now ready for end users.

There are two main reasons for using parallel algorithms [1,2]: (i) cases given a fixed time to search, to increase the quality of the solutions found within that time; (ii) cases given a fixed solution quality, to reduce the time to find a solution not worse than that quality.

In this paper, we discuss parallel ACO algorithms mainly applied to the first reason to solve the quadratic assignment problems (QAPs), with SMP type computing platforms, within the same time as is given for computing platforms with a single processor. We discuss several parallel schemes for this purpose. For this study, we use the cunning Ant System (*cAS*) that showed promising performance both for the traveling salesman problems (TSPs) [3] and QAPs [4]. In solving a QAP with an ACO algorithm, local search occupies a major portion of run time. We use this feature when we set up our parallel schemes.

The articles are organized as follows. Section 2 describes related work in brief. Section 3 gives a brief overview of *cAS*. Then, Section 4 describes various schemes of parallel implementation of *cAS* in SMP for solving QAP, and the empirical analysis is given in Section 5. Finally, Section 6 concludes this paper.

2 Related Work

Many parallel ACO algorithms have been studied [1,2,5,6,7,8,9,10]. Brief summaries can be found in [2,9]. In [1], parallel MMAS with k independent runs was studied.

The most commonly used approach to parallelization is island model where multiple colonies exchange information synchronously or asynchronously. In [6], it is reported that the communication of the whole pheromone matrix leads to a decreased solution quality as well as worse run-time. However the exchanges of best-so-far solutions approach leads to good solution quality.

In [9], a scheme in which the whole pheromone matrix is shared with two colonies using SMP is studied on TSPs. The results showed no clear advantage of parallel ACO algorithms over sequential algorithm. In [2], parallel MMAS algorithms using message passing interface (MPI) libraries with various topologies have been studied on TSPs, and a clear advantage of parallel algorithms with independent run is reported.

3 An Overview of cAS [3,4]

cAS uses agents called cunning ants (*c-ants*), which differ from traditional ants in the manner of solution construction. A part of each new solution is taken from one of the previously generated solutions (also called a donor ant; *d-ant*) whereas the remainder of the solution is generated probabilistically from pheromone density $\tau_{ij}(t)$ as usual, where t is iteration counter of the algorithm. In a sense, since this agent in part appropriates the work of others to construct a solution, we named the agent a cunning ant after the metaphor of its cunning behavior.

In cAS, we maintain an archive consisting of m candidate solutions generated in the past; k th solution in the archive at iteration t is denoted by $s_{k,t}$ ($k \in \{1, 2, \dots, m\}$). At iteration t , a new $c\text{-}ant_{k,t+1}$ (solution) is generated for each position k in the archive using the current $s_{k,t}$ i.e., solution in this position, as the donor. Then, the newly generated $c\text{-}ant_{k,t+1}$ is compared with its donor $s_{k,t}$ with respect to the objective function, and the best of the two survives as the next solution in this position of the archive, $s_{k,t+1}$.

The pheromone density $\tau_{ij}(t)$ in QAP correspond to the desirability of assigning a facility i to a location j . This pheromone density is updated with $s_{k,t+1}$ for $k=1, 2, \dots, m$ and $\tau_{ij}(t+1)$ is obtained as in Ant System (AS) [12], keeping all pheromone densities within the interval $[\tau_{min}, \tau_{max}]$ as in MMAS [11].

Let us represent the number of nodes of partial solution that are constructed based on $\tau_{ij}(t)$, by l_s (i.e., l_c , the number of nodes of partial solutions from its donor, is $n - l_s$). Then cAS introduces the control parameter γ which can define $E(l_s)$ (the average of l_s) by $E(l_s) = n \times \gamma$. For each creation of a new candidate solution, we determine l_s so that $E(l_s) = n \times \gamma$ is satisfied. One simple approach is to determine value of l_s deterministically as $l_s = n \times \gamma$. In this research, we used a probabilistic function defined in [3,4] (please see those references for details).

4 Parallel Implementation of *cAS* in SMP for QAP

4.1 Base Model of the Sequential *cAS* for QAP

The QAP is the problem of assigning a set of facilities to a set of locations with given distances between the locations and given flows between the facilities. Given n facilities and n locations, two $n \times n$ matrices $A = [a_{ij}]$ and $B = [b_{rs}]$, where a_{ij} is the distance between locations i and j , and b_{rs} is the flow between facilities r and s , the QAP is the problem to minimize

$$f(\phi) = \sum_{i=1}^n \sum_{j=1}^n b_{ij} a_{\phi(i)\phi(j)}, \quad (1)$$

where ϕ is an arbitrary permutation of the set of integers.

In [11], when MMAS for QAP was combined with the robust taboo search (Ro-TS) by Taillard [13], parameter settings and the methods of applying Ro-TS were carefully designed so that computational time was the same as the computational times of other meta-heuristics to attain a fair comparison. These are as follows: the number of ant (m) was 5 and the evaporation rate (ρ) was 0.8. 250 times of short Ro-TS runs of length $4n$ were applied. Since $m=5$, the maximum iterations are 50 ($= 250/5$). In a previous study in [4], we used the same setup in *cAS* for QAP and got promising results.

For the base model of the sequential *cAS* in this work, we used the same setup with MMAS in [11] except for the value of m and the maximum number of solution constructions. Number of processors in SMP is usually an even number (power of 2), we used m value of 8, which is easy to configure for parallelization. Number of Ro-TS runs of length $4n$ is 248 (fewer than in [11] by 2) and thus the maximum iterations are 31 ($= 248/8$). We found the performance of *cAS* with $m = 8$ is slightly better than that with $m = 5$ in [4].

4.2 Parallel Implementation of *cAS* for QAP with a SMP

In this study, we have implemented two types of parallel *cAS*; master-slave models and island models.

Master-Slave Models with

SMP. Table 1 shows computation time occupied by the local search (Ro-TS) in runs of sequential *cAS* with the base model described in Section 4.1.

Table 1. Time used by the local search

Problem size n	40	50	60	80	100
Local search (%)	99.75%	99.85%	99.89%	99.94%	99.96%

From this table, we can see that more than 99% of computational time of the base *cAS* is occupied by the local search. Taking into account this feature in solving QAP, we can implement a parallel model with a master-slave model as shown in Fig. 1. In this model, P threads perform local search in parallel. Here, P is number of processors of the SMP. In this research, we implemented the following two types of master-slave models.

(1) Master-slave model with longer local search: In this model, we generate P threads which can run in parallel. m solutions are generated in each iteration of *cAS* of the colony thread, we assign m/P solutions to each local search thread. Since we assume run time for this model is the same as the base *cAS*, each local search thread can run local search longer by P times. We identify this model as MS-LSx P .

(2) Master-slave model with longer iteration of *cAS*: Instead of longer run of local search of MS-LSx P , we increase maximum iteration number of *cAS* by P times. We identify this model as MS-ITx P .

Island Models with SMP. The island model is the most popular parallel processing model in evolutionary computation. In our implementation, we exchange the solutions in the archive. There are P colonies as shown in Fig. 2. Each colony is coded as a thread which can run in parallel. They exchange solutions through the control thread every I interval of iteration synchronously.

Although there are many topologies [2], in this study we implemented the three models as described in the following.

(1) Island model with *fully-connected*: In this model, the control thread collects the best solutions in the archive of each colony and chooses the best solution from them, and then distributes it to all colonies except the colony that produced that best solution. In each colony, the worst solution in each archive is replaced with the received best solution. We identify this model as IS-FCx P .

(2) Island model with *replace-worst*: This model is basically the same as (1) but the replacement is performed only in the colony which produced the worst solution among the best solutions. We identify this model as IS-RWx P .

(3) Island model with *independent-runs*: P colonies simply run independently until the end of the run. We identify this model by IS-INx P .

5 Experiments of Parallel *cAS* on QAP

5.1 Experimental Setup

Here we evaluate parallel *cAS* on QAP using SMP. Setup for the base sequential *cAS* is described already in Section 4.1. All experiments were performed 25 times. The machine we used had two Dual Core AMD OpteronTM 280 (2.4GHz) processors with 2.87GB main memory, and 32-bit Windows XP. Thus, this machine has a total of four processors with an SMP architecture.

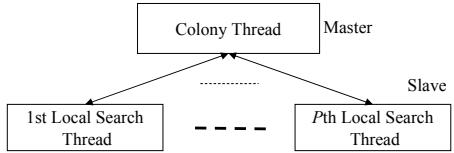


Fig. 1. Master-slave model in SMP

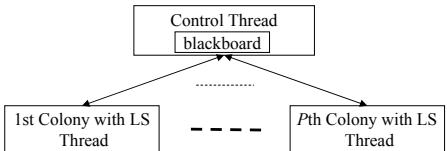


Fig. 2. Island model in SMP

In *cAS*, parameter value of γ (see Section 3) plays an important role. According to [14,11], QAP instances in QAPLIB [15] can be divided into 4 classes; (i) randomly generated instances, (ii) grid-based distance matrix instances, (iii) real-life instances, and (iv) real-lifelike instances. In previous study in [4], we found γ value of 0.4 for QAP classes (i) and (ii), and 0.8 for (iii) and (iv) work well, respectively. Thus, in experiments in Section 5 we used γ values of 0.4 for classes (i) and (ii), and 0.8 for (iii) and (iv), respectively.

cAS was written in Java and the thread programming uses Java Thread class. We used the Ro-TS code which is available at [13], though the code, which is originally written in C, was rewritten in Java.

To attain the same run time between parallel *cAS* on the SMP machine and the base *cAS* on the same machine, we experimented in the following manner.

(1) The base model: The base model uses the setup described in Section 4.1, and was run as a single process. This model becomes the base to see the solution quality and real run time.

(2) Master-slave model: We ran MS-LSxP and MS-ITxP with $P=4$. Thus, these models are indicated as MS-Lx4 and MS-Ix4, respectively.

(3) Island model: We ran IS-INxP, IS-FCxP abd IS-RWxP with $P=4$ with solutions exchange interval $I = 2$. These models are indicated as IS-INx4, IS-FCx4 and IS-RWx4, respectively.

All of these models should have almost the same run time (see Section 4.2) if the machine has a well-designed architecture.

5.2 Analysis of the Results

Results are summarized in Table 2. In the table, the notations are as follows:

Error(%) indicates the solution quality calculated by the average of $(function\ value - known\ best)/known\ best$ over 25 independent runs.

R-time (ratio) indicates average run time of algorithm of each model in seconds. Values in brackets are ratios of the run time of the base model.

#OPT indicates the number of runs in which algorithms succeeded in finding the known best solution.

O-time (ratio) indicates the average time to find the known best solution in those runs where it did find it, where the values in brackets are ratios of the *O-time* of the base model.

B-time (ratio) indicates the average time at which the best functional values of each run was found, where the values in brackets indicate ratios of the *B-time* of IS-INx4.

Before analyzing the results, let us see values of the *R-time (ratio)* of each experiment. We can see that each run time of the parallel model attained almost the same value as the run time of its base model, i.e., the value of *ratio* is very near to 1. Thus, we can say that the SMP platform used in this research executed the algorithms in parallel without significant performance degradation. Overhead caused by the synchronous solutions exchanges with interval $I = 2$ in the island models was also negligible.

All results with the parallel models show increases of solution quality as compared to the quality of their corresponding base models. In the following, we analyze these results for each problem class.

From the results of class (i) instances, we can see the values of *Error* of master-slave models are slightly better than those of the island models. But in QAP instances of this class (randomly generated instances), there are not such significant differences among the five parallel models, though the results of the island model with independent runs (IS-INx4) show the worst *Error* values among island models for all instances in this class. However, if we look at the *B-time*, the master-slave models show smaller *B-time*. This means the master-slave models find their best solutions faster in each run than island models.

From the results of class (ii) instances (grid-based distance matrix instances), we can see the values of *Error* of master-slave models are slightly better than those of the island mod-

Table 2. Results of parallel cAS with SMP

QAP	Measure	Base Model	Master-Slave Model		Island Model		
			MS-1-Sx4	MS-ITx4	IS-INx4	IS-FCx4	IS-RWx4
class (i) with $\gamma=0.4$							
tai40a	<i>Error (%)</i>	0.789	0.495	0.630	0.587	0.566	0.569
	R-time (ratio)	6.81 (1)	(1.00)	(1.04)	(1.00)	(1.03)	(1.01)
	#OPT	0	1	0	0	0	0
	O-time (ratio)	-	0.92	-	-	-	-
	B-time (ratio)	-	(0.85)	(0.93)	3.83 (1)	(0.85)	(1.16)
tai50a	<i>Error (%)</i>	1.064	0.738	0.830	0.832	0.804	0.773
	R-time (ratio)	13.44 (1)	(0.99)	(1.03)	(1.00)	(1.01)	(1.01)
	#OPT	0	1	0	0	0	0
	O-time (ratio)	-	11.56	-	-	-	-
	B-time (ratio)	-	(0.82)	(0.82)	8.92 (1)	(0.92)	(1.04)
tai60a	<i>Error (%)</i>	1.101	0.881	0.930	0.913	0.899	0.886
	R-time (ratio)	23.34 (1)	(0.99)	(1.02)	(1.00)	(1.01)	(1.01)
	#OPT	0	0	0	0	0	0
	O-time (ratio)	-	-	-	-	-	-
	B-time (ratio)	-	(0.92)	(1.00)	16.65 (1)	(0.69)	(1.00)
tai80a	<i>Error (%)</i>	0.815	0.592	0.547	0.672	0.581	0.585
	R-time (ratio)	56.73 (1)	(0.99)	(1.02)	(1.01)	(1.00)	(1.01)
	#OPT	0	0	0	0	0	0
	O-time (ratio)	-	-	-	-	-	-
	B-time (ratio)	-	(0.98)	(0.84)	36.81 (1)	(1.19)	(1.23)
class (ii) with $\gamma=0.4$							
sko49	<i>Error (%)</i>	0.057	0.034	0.024	0.035	0.030	0.031
	R-time (ratio)	12.57 (1)	(0.99)	(1.02)	(1.00)	(1.00)	(1.02)
	#OPT	6	10	14	10	12	11
	O-time (ratio)	6.89 (1)	0.80	(0.91)	(1.19)	(1.01)	(1.04)
	B-time (ratio)	-	(0.82)	(0.88)	7.37 (1)	(1.01)	(1.08)
sko64	<i>Error (%)</i>	0.072	0.005	0.011	0.009	0.005	0.006
	R-time (ratio)	28.47 (1)	(0.98)	(1.01)	(1.00)	(1.00)	(1.00)
	#OPT	1	10	8	13	17	16
	O-time (ratio)	20.20 (1)	(0.89)	0.61	(0.77)	(0.71)	(0.78)
	B-time (ratio)	-	(1.03)	(0.78)	15.47 (1)	(0.96)	(1.00)
sko81	<i>Error (%)</i>	0.129	0.044	0.048	0.076	0.049	0.059
	R-time (ratio)	58.91 (1)	(0.99)	(1.02)	(1.01)	(1.01)	(1.01)
	#OPT	0	0	1	0	0	0
	O-time (ratio)	-	-	33.92	-	-	-
	B-time (ratio)	-	(0.96)	(0.97)	43.53 (1)	(0.93)	(1.09)
sko90	<i>Error (%)</i>	0.155	0.074	0.095	0.085	0.050	0.063
	R-time (ratio)	81.04 (1)	(1.00)	(1.03)	(1.01)	(1.01)	(1.01)
	#OPT	0	0	0	1	2	1
	O-time (ratio)	-	-	-	57.860	39.36	13.25
	B-time (ratio)	-	(0.92)	(0.91)	59.77 (1)	(1.05)	(1.12)
class (iii) with $\gamma=0.8$							
kra30a	<i>Error (%)</i>	0.054	0	0	0	0	0
	R-time (ratio)	2.84 (1)	(1.00)	(1.05)	(1.03)	(1.00)	(0.99)
	#OPT	24	25	25	25	25	25
	O-time (ratio)	0.88 (1)	(0.81)	(0.65)	(0.52)	0.49	(0.52)
	B-time (ratio)	-	(1.55)	(1.25)	0.46 (1)	(0.94)	(0.99)
kra30b	<i>Error (%)</i>	0.009	0	0	0	0	0
	R-time (ratio)	2.84 (1)	(1.00)	(1.05)	(1.01)	(0.99)	(1.01)
	#OPT	22	25	25	25	25	25
	O-time (ratio)	1.46 (1)	(0.51)	0.35	(0.47)	(0.58)	(0.47)
	B-time (ratio)	-	(1.09)	(0.74)	0.69 (1)	(1.24)	(1.00)
ste36a	<i>Error (%)</i>	0.032	0.004	0.010	0.025	0.023	0.004
	R-time (ratio)	4.94 (1)	(1.00)	(1.04)	(1.00)	(1.01)	(1.00)
	#OPT	20	24	24	23	24	23
	O-time (ratio)	2.96 (1)	(0.59)	(0.64)	(0.79)	(0.86)	(0.79)
	B-time (ratio)	-	(0.83)	(0.98)	2.28 (1)	(1.09)	(0.97)
ste36b	<i>Error (%)</i>	0	0	0	0	0	0
	R-time (ratio)	4.94 (1)	(1.00)	(1.04)	(1.01)	(1.01)	(1.00)
	#OPT	25	25	25	25	25	25
	O-time (ratio)	1.42 (1)	(0.50)	0.31	(0.63)	(0.57)	(0.63)
	B-time (ratio)	-	(0.78)	(0.48)	0.89 (1)	(0.90)	(0.98)
class (iv) with $\gamma=0.8$							
tai40b	<i>Error (%)</i>	0	0	0	0	0	0
	R-time (ratio)	6.76 (1)	(1.01)	(1.04)	(1.01)	(1.01)	(1.01)
	#OPT	25	25	25	25	25	25
	O-time (ratio)	3.16 (1)	(0.90)	0.26	(0.50)	(0.40)	(0.44)
	B-time (ratio)	-	(1.79)	(0.53)	1.58 (1)	(0.80)	(0.88)
tai60b	<i>Error (%)</i>	0.071	0.069	0.046	0.009	0.002	0.001
	R-time (ratio)	23.17 (1)	(0.99)	(1.02)	(1.01)	(1.00)	(1.00)
	#OPT	4	9	22	9	23	21
	O-time (ratio)	22.61 (1)	(0.86)	0.35	(0.93)	(0.76)	(0.86)
	B-time (ratio)	-	(0.93)	(0.41)	20.79 (1)	(0.84)	(0.96)
tai80b	<i>Error (%)</i>	0.484	0.671	0.208	0.226	0.262	0.196
	R-time (ratio)	56.58 (1)	(0.99)	(1.03)	(1.00)	(1.01)	(1.01)
	#OPT	0	0	9	0	2	0
	O-time (ratio)	-	-	34.35 (1)	-	(1.67)	-
	B-time (ratio)	-	(1.11)	(0.81)	50.68 (1)	(1.07)	(1.08)
tai100b	<i>Error (%)</i>	0.154	0.138	0.055	0.112	0.075	0.102
	R-time (ratio)	114.89 (1)	(0.98)	(1.00)	(1.01)	(1.01)	(1.01)
	#OPT	0	0	13	0	3	0
	O-time (ratio)	-	-	50.77 (1)	-	(1.99)	-
	B-time (ratio)	-	(0.96)	(0.61)	106.35 (1)	(1.00)	(1.03)

els except for sko90. Again IS-INx4 shows the worst *Error* values among island models for all instances in this class. Comparisons with *O-time* are possible on sko49 and sko64. On these instances, the master-slave models have smaller values of *O-time* than the island models, showing faster finding of the known best solutions.

Results of class (iii) instances (real-life instances) showed a different feature as compared to the results of classes (i) and (ii). In class (iii), values of $\#OPT$ for all parallel models attained 25 except for on ste36a. So, we cannot see differences in *Error* values among parallel algorithms. Instead let us turn our attention to *O-time (ratio)*. For example in ste36b, $\#OPT$ of all models including the base model is 25. On the other hand, looking at the *O-time* values of the parallel model, the ratios of *O-time* against the base model are 0.50 (MS-LSx4), 0.31 (MS-ITx4), 0.63 (IS-INx4), 0.57 (IS-FCx4), and 0.63 (IS-RWx4), respectively. Thus, we can see that the master-slave models perform faster searches than the island models on this instance. Other instances in this class except for kra30a showed similar results. On kra30a, island models find known best solution faster than the master-slave models.

Results of class (iv) instances (real-lifelike instances) showed similar results to that of class (iii), though values of $\#OPT$ in this class are smaller than those of (iii) instances except for tai40b. Results for tai40b are almost the same as instances of class (iii). Although on tai60b, the values of *Error* of all island models are smaller than the values of *Error* of master-slave models, the values of *O-time* of island models are much larger than those of master-slave models. On the all instances in this class, MS-ITx4 has the smallest *O-time* values among parallel models. Thus, the advantage of MS-ITx4 can be observed in this class. We can also observe that the *B-time* of the MS-ITx4 is the smallest among all parallel models.

As a summary of analysis in this section, we can say master slave models, especially MS-ITx4 (master-slave model with longer iteration), showed the clear advantage for instances in classes (iii) and (iv) on the measures of *Error*, *O-time*, and *B-time*. Although we could not observe which parallel models are the best for instances in classes (i) and (ii) by looking at *Error* values, *B-time* analysis suggests that using master-slave models is a better choice than island models.

6 Conclusions

In SMP, parallel run can be performed efficiently with less communication overhead among processors. In this paper, we proposed a total of five parallel ACO algorithms with SMP for solving QAP. We evaluated each parallel algorithm under conditions in which the run time for each parallel algorithm and the base algorithm are about the same.

The results showed all parallel models studied here improved solution quality of the base algorithm. Although detail improvement features were different depending on problem classes of QAPs, our results suggested that using the master-slave models, especially MS-ITxP (master-slave model with longer

iteration) for classes (iii) and (iv), is promising in solution quality and search speed for solving QAPs where computation for local search occupies a large part of the total computation time. To apply these models to problems in other domains remains for future work.

Acknowledgements. This research is partially supported by the Ministry of Education, Culture, Sports, Science and Technology of Japan under Grant-in-Aid for Scientific Research number 19500199.

References

1. Stützle, T.: Parallelization strategies for ant colony optimization. In: 5th International Conf. on Parallel Problem Solving for Nature (PPSN-V), pp. 722–731 (1998)
2. Manfrin, M., Birattari, M., Stützle, T., Dorigo, M.: Parallel ant colony optimization for the traveling salesman problems. In: Ant Colony Optimization and Swarm Intelligence, 5th International Workshop, ANTS 2006, pp. 224–234 (2006)
3. Tsutsui, S.: *cAS*: Ant colony optimization with cunning ants. In: Proc. of the 9th Int. Conf. on Parallel Problem Solving from Nature (PPSN IX), pp. 162–171 (2006)
4. Tsutsui, S.: Cunning ant system for quadratic assignment problem with local search and parallelization. In: Ghosh, A., De, R.K., Pal, S.K. (eds.) PReMI 2007. LNCS, vol. 4815, pp. 269–278. Springer, Heidelberg (2007)
5. Randall, M., Lewis, A.: A parallel implementation of ant colony optimization. Journal of Parallel and Distributed Computing 62(9), 1421–1432 (2002)
6. Benkner, S., Doerner, K., Hartl, R., Kiechle, G., Lucka, M.: Communication strategies for parallel cooperative ant colony optimization on clusters and grids. In: Complimentary Proc. of PARA 2004 Workshop on State-of-the-art in Scientific Computing, pp. 3–12 (2005)
7. Bullnheimer, B., Kotsis, G., Strauss, C.: Parallelization strategies for the ant system. High Performance Algorithms and Software in Nonlinear Optimization, 87–100 (1998)
8. Middendorf, M., Reischke, F., Schmeck, H.: Multi colony ant alorithms. Journal of Heuristics 8(3), 3005–3200 (2002)
9. Lv, Q., Xia, X., Qian, P.: A parallel aco approach based on one pheromone matrix. In: Ant Colony Optimization and Swarm Intelligence, 5th International Workshop, ANTS 2006, pp. 332–339 (2006)
10. Talbi, E., Roux, O., Fonlupt, C., Robillard, D.: Parallel ant colonies for the quadratic assignment problem. Generation Computer System 17, 441–449 (2001)
11. Stützle, T., Hoos, H.: MAX-MIN ant system. Future Generation Computer Systems 16(9), 889–914 (2000)
12. Dorigo, M., Maniezzo, V., Colomi, A.: The ant system: Optimization by a colony of cooperating agents. IEEE Trans. on SMC-Part B 26(1), 29–41 (1996)
13. Taillard, É.D.: The robust taboo search code,
<http://mistic.heig-vd.ch/taillard/>
14. Taillard, É.D.: Robust taboo search for the quadratic assignment problem. Parallel Computing 17, 443–455 (1991)
15. QAPLIB –AQuadratic Assignment Problem Library,
<http://www.seas.upenn.edu/qplib/>

Social Odometry in Populations of Autonomous Robots

Álvaro Gutiérrez¹, Alexandre Campo²,
Francisco C. Santos², Carlo Pincioli², and Marco Dorigo²

¹ ETSIT, Universidad Politécnica de Madrid, Madrid, Spain
aguti@etsit.upm.es

² IRIDIA, CoDE, Université Libre de Bruxelles, Brussels, Belgium
{acampo,fsantos,cpinciro,mdorigo}@ulb.ac.be

Abstract. The improvement of odometry systems in collective robotics remains an important challenge for several applications. In this work, we propose a localisation strategy in which robots have no access to centralised information. Each robot has an estimate of its own location and an associated confidence level that decreases with distance travelled. Robots use estimates advertised by neighbouring robots to correct their own location estimates at each time-step. This simple *online* social form of odometry is shown to allow a group of robots to both increase the quality of individuals' estimates and efficiently improve their collective performance. Furthermore, social odometry produces a successful self-organised collective pattern.

1 Introduction

Many robotics applications require localisation methods to achieve different tasks. Many different solutions to the localisation problem have been implemented. Among these, odometry is probably the most used as it provides easy and cheap real time position information by the integration of incremental motion information over time. Unfortunately, this integration causes an accumulation of errors during the movement of the robot. Many different approaches have been implemented to deal with systematic and non-systematic localisation errors [1]. Some implementations have used Kalman filters [2]. Thrun and colleagues in [3] create a map of indoor environments combining the idea of posterior estimation with incremental map construction using maximum likelihood estimators.

Some applications in multirobot exploration are implemented without using odometry or dead-reckoning techniques. In [4], a group of robots remains stationary while the other team is in motion. In [5], only one robot is allowed to move while the others act as immobile landmarks. In [6], a chain between two specific areas is created, so that the rest of the group can follow it. In [7], the LOST method enables a team of robots to navigate in an indoor environment. Each robot updates the new optimal path to the goal communicating with a central computer.

All these implementations have a number of different limitations: *i*) they are power consuming in terms of computation because of the Kalman filters and use of maps, *ii*) some robots are not allowed to move while others are tracking distance between them, *iii*) robots must maintain visual contact at all times with the rest of the group, and *iv*) in some cases robots have to communicate with a central device to update or download maps, synchronise movements, or update positions.

The solution we propose in this paper exploits self-organised cooperation in a group of robots to reduce each individual location error. In a nutshell, each robot location knowledge consists of an estimate of its own location and an associated confidence level that decreases with the distance travelled. In order to maximise its confidence about its estimate, each individual tries to update it using the information available in its neighbourhood. Estimated locations, confidence levels and actual locations of the robots co-evolve in parallel in order to guide each robot to the correct objective. This simple *online* social dynamics is shown to allow the population of robots to both reduce individual's errors and efficiently reach a common objective.

In the rest of this paper, we present the details of our solution and analyse its performance in a particular task in which collective correction of odometry errors is beneficial.

2 The Task

To study cooperative odometry, we have devised a task of central place foraging [8] in which robots need to explore the environment to find resource sites and bring items back to a central place. The robots can perceive the central place and the resource site only when they are closer than a threshold distance, given by their sensorial capabilities. Initially, robots are scattered in the arena and they explore the environment to locate both areas. Robots rely on odometry to maintain an estimate of the location of each area (central place and resource site). As soon as a robot comes back to an area, the corresponding location estimate is reset and odometry errors affecting this estimate are discarded.

In an ideal case, robots would make no mistake in estimating the location of the two areas. They could travel endlessly from one place to the other without drifting away. As soon as errors are introduced in the odometry system, estimated locations differ from true locations. A robot may not manage to go back to a given area, and may end up lost. In that case, the robot is doomed to explore again the environment to find the area. To reduce the impact of odometry errors, robots keep in memory both estimated area locations and share this information among them.

3 Mobile Robot Positioning

The accuracy of odometry measurements depends on the kinematics of the robot (see [9]). Let the location of a robot at time $k - 1$ be $L_{k-1} = [x_{k-1} \ y_{k-1} \ \theta_{k-1}]^T$

where (x_{k-1}, y_{k-1}) are the Cartesian coordinates and θ_{k-1} is the orientation with respect to a global reference frame. A rotation $\Delta\theta_k$ and a translation $\Delta\rho_k$ move the robot to a new location L_k :

$$L_k = L_{k-1} + \begin{bmatrix} \Delta\rho_k \cos(\theta_{k-1} + \Delta\theta_k/2) \\ \Delta\rho_k \sin(\theta_{k-1} + \Delta\theta_k/2) \\ \Delta\theta_k \end{bmatrix} \quad (1)$$

Equation 1 is not taking into account problems such as slippage, unequal wheels diameters, wheels misalignments, etc. These errors can be classified as either *systematic* or *non-systematic errors* [10]. The first ones can be modelled and corrected [11], while the last ones can not be corrected and many classical techniques have been implemented to cope with them [12].

When modelling non-systematic errors, each computed robot position is surrounded by a characteristic *error ellipse* that represents a region of uncertainty in which the actual location lies. This region grows with the distance travelled, and it is reset to zero when the robot can localise itself exactly thanks to an environmental landmark (entering one of the two areas in our case). The error ellipse model is based on the covariance matrix of the robot's location defined as:

$$\text{Cov}(L_k) = J_{L_{k-1}} f \cdot \text{Cov}(L_{k-1}) \cdot J_{L_{k-1}} f + J_{U_k} f \cdot \text{Cov}(U_k) \cdot J_{U_k} f \quad (2)$$

where $J_{L_{k-1}} f$ and $J_{U_k} f$ are the Jacobians of f with respect to L_{k-1} and U_k , f is the $(x, y, \theta, \Delta U_r, \Delta U_l)$ vector, U_k is the $(\Delta U_l, \Delta U_r)$ vector and ΔU_l and ΔU_r are the displacements of the left and right wheels respectively. The covariance matrix $\text{Cov}(L_0)$ has an initial value of 0.

4 Methods

4.1 Experimental Setup

Our experiments are carried out using a simulation software based on ODE¹. The arena is a bounded square area of 3x3 m² and the robots are randomly scattered in the centre of the arena at the beginning of the experiment. The ground of the arena is white except for the two areas: central place is a black circle of 20 cm radius and the resource site is a grey circle of the same dimensions. Robots differentiate areas using an infrared sensor directed to the ground. A specific simulated range and bearing communication board, based on infrared sensors, allows robots to send messages to each other when their interdistance is less than 25 cm.

We introduce errors to simulate the imperfect response of the range and bearing sensor. Noise is added to the bearing ($\pm 20^\circ$) and range (± 2.5 cm) values. Moreover, each message emitted can be lost with a probability that varies linearly from 1% when the sender-receiver distance is less than 1 cm, to 50% when the two robots are at 25 cm from each other. Errors have also been introduced on the encoder sensors chosen uniformly random in $\pm 20\%$ of the maximum movement at each time step.

¹ <http://www.ode.org>

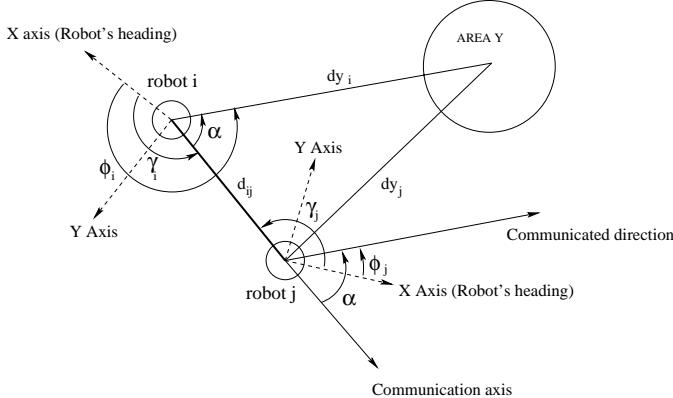


Fig. 1. Robots sharing information about the estimated location of area Y

4.2 Learning from Others

Figure 1 shows how information about the estimated location of area *Y* is transmitted from robot *i* to robot *j*. In a first step, robot *i* transmits its estimate of the distance dy_i and direction ϕ_i of area *Y* to robot *j*. For the direction, the value transmitted is the angle α , obtained from ϕ_i using the communication beam as reference axis: $\alpha = \phi_i - \gamma_i$. In a second step, robot *j* transforms the received data into its own coordinates system. First, it calculates the direction pointed by robot *i* as $\phi_j = \gamma_j + \alpha - \pi$. Then, robot *j* calculates the location $loc_j = (x, y)$ of area *Y* related to its own reference frame using robot *i* information and the simple trigonometric equations $x = d_{ij} \cdot \cos\gamma_j + dy_i \cdot \cos\phi_j$ and $y = d_{ij} \cdot \sin\gamma_j + dy_i \cdot \sin\phi_j$.

At this stage, robot *j* has the opportunity to adopt the estimate of the neighbour, to keep its own or to produce an updated location (loc_up_j) based on both. Given that estimates get worse with distance travelled, the robots use the inverse of the distance travelled as a confidence level of their estimated location. This confidence level, denoted by ϵ_i for robot *i*, respectively ϵ_j for robot *j*, is part of any communicated location and informs about the reliability, or quality, of the information. To calculate loc_up_j , we use the so-called pairwise comparison rule [13] for the social learning dynamics, which makes use of the Fermi distribution:

$$c = \frac{1}{1 + e^{-\beta(\epsilon_i - \epsilon_j)}}, \quad (3)$$

where β measures the importance of the relative confidence levels in the decision making. We use a weighted average to merge locations and confidence levels using the Fermi function: $loc_up_j = c \cdot loc_j + (1 - c) \cdot loc_i$ and $\epsilon_up_j = c \cdot \epsilon_j + (1 - c) \cdot \epsilon_i$

5 Results

The performance of the robots in the foraging task under study is measured as the number of total round trips completed from the central place to the resource site and back during one simulated hour.

Figure 2a shows box-plots of the performance with respect to different β values tested. We observe an optimal value $\beta = 10^{-3}$ showing that robots efficiently rely on imitation to increase their collective performance.

Next, we carry out a comparison of different behaviours:

- **no communication:** robots do not communicate and are affected by odometry errors.
- **no odometry error:** robots communicate and are not affected by odometry errors.
- **covariance knowledge:** robots communicate and robots update new location using their own covariance matrix value and the one offered by their neighbours.
- **global communications:** robots communicate with each other globally. Each robot updates its estimates by averaging the knowledge of the whole group.
- **local communications:** robots communicate with each other locally, β is set to its optimal value previously determined, that is, the behaviour detailed in section 4.

Figure 2b reports the outcome of the comparisons. In the **no communication** behaviour, robots rely solely on error prone odometry to find the areas. Once lost, they have to explore the environment and find the areas by chance, which explains the poor performance. In the **no odometry error** case, robots know

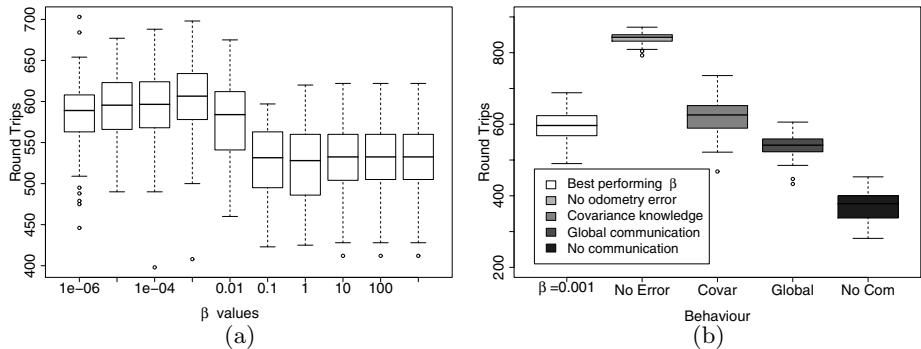


Fig. 2. (a) Task performance using local communications as a function of parameter β . (b) Task performance for the different individual behaviours tested (30 replications for each boxplot). Each box comprises observations ranging from the first to the third quartile. The median is indicated by a horizontal bar, dividing the box into the upper and lower part. The whiskers extend to the farthest data points that are within 1.5 times the interquartile range. Outliers are shown as dots.

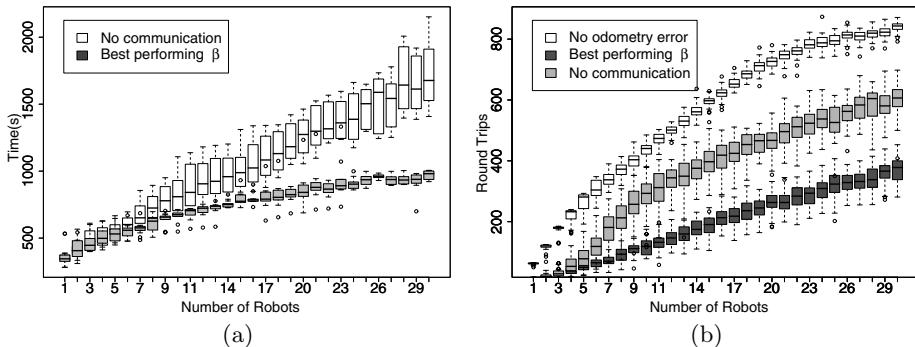


Fig. 3. (a) Time employed for a group of 30 robots to initially localise the central place and the resource site. (b) Task performance as function of group size for three representative cases (30 replications for each boxplot).

accurately the location of both areas. When robots employ **global communication**, they tend to perform worse than with **local communication**, mainly because of the negative influence of lost robots on the knowledge of the group. Interestingly, the **covariance knowledge** behaviour does not show statistically different performance with **local communications** with the best performing β . Not taking into account ideal behaviours that are not implementable in reality, our collective behaviour is therefore exhibiting maximal performance.

We focus now on the comparison between the **no communication** and the **local communication** behaviours. We study the time required to have each robot visit the central place and the resource site at least once. Figure 3a shows these times as a function of the number of robots. With few robots, the two behaviours perform equally well, which is explained by the infrequent encounters of the robots and the consequent low amount of communications. As the number of robots in the experiment increases, we clearly observe that using **local communication** allows the robots to find the areas faster. Using communication, robots are intrinsically carrying out a recruitment process which speeds up the initial exploration phase.

Figure 3b shows the performance of the robots with respect to increasing density, using three different behaviours. The performance of **local communication** is always in between the performances of **no odometry error** and **no communication**. When the number of robots is small (less than 10), the performance of **local communication** is increasing very fast. With more than 10 robots, the performance still increases but at a lower rate. This is due to two counterbalancing effects, namely the improvement of the knowledge of areas' locations through communication versus the disruption of the measure of odometry caused by a higher number of obstacle avoidance events. Results suggest that a high density of robots disrupts performance and there is most likely an optimal density to carry out the foraging task as reported in [14]. We also see that local communication allows the robots to cope with density to some extent and have performance that scales linearly in a wide range of situations.

6 Conclusions

In this paper we have described a social strategy in which robots use pairwise local communication to share knowledge about specific locations to improve their performance in a foraging task. By letting the robots use the estimates of the others, we engineer an efficient and decentralised knowledge sharing mechanism which allows the robots to achieve their goals, both from an individual and group perspective. This simple mechanism drives the system to a successful collective pattern that none of the individuals is able to achieve independently.

We show that local communications are more effective than global communications which would additionally require either more expensive devices or a central system taking care of routing robots' communications. Social odometry does not rely on any internal model provided to each robot, but exclusively on a general collective dynamics of knowledge sharing. This has obvious advantages, given that less effort is needed for environmental dependent parameter tuning.

Preliminary observations suggest that the β parameter depends on the distance between the areas, suggesting that the robots can tune parameter β knowing the distance between the central place and the resource site. The tuning of the parameter can be done off-line, where the designer introduces the value or on-line, where robots update the β parameter once they have located both areas. In the future, we intend to implement and test this strategy on real robots, emphasising the online tuning of the parameter β .

Finally, the performance of the social odometry allows an optimistic forecast concerning the use of online self-organised methodologies in the field of collective robotics.

Acknowledgments. A. Gutiérrez acknowledges support from the Consejo Social of the Universidad Politécnica de Madrid via the Fifth PhD Formation programme. A. Campo, F. C. Santos and M. Dorigo acknowledge support from the F.R.S.-FNRS. C. Pinciroli acknowledges support from COMP2SYS, a Marie Curie Early Stage Research Training Site funded by the European Community's Sixth Framework Programme. This work was partially supported by the SWARMOID project, funded by the Future and Emerging Technologies programme (IST-FET) of the European Commission and by the ANTS project, an *Action de Recherche Concerté* funded by the Scientific Research Directorate of the French Community of Belgium. The information provided is the sole responsibility of the authors and does not reflect the European Commission's opinion. The European Commission is not responsible for any use that might be made of data appearing in this publication.

References

1. Wang, C.M.: Location estimation and uncertainty analysis for mobile robots. *Autonomous Robot Vehicles* 1(1), 90–95 (1990)
2. Larsen, T., Bak, M., Andersen, N., Ravn, O.: Location estimation for autonomously guided vehicle using an augmented Kalman filter to autocalibrate the odometry. In: FUSION 1998 Spie Conference, pp. 33–39. CSREA Press, Las Vegas (1998)

3. Thrun, S., Burgard, W., Fox, D.: A real-time algorithm for mobile robot mapping with applications to multi-robot and 3D mapping. In: Proceedings of the IEEE International Conference on Robotics and Automation, pp. 321–328. Robotics and Automation Society, NJ (2000)
4. Kurazume, R., Hirose, S.: An experimental study of a cooperative positioning system. *Autonomous Robots* 8(1), 43–52 (2000)
5. Grabowski, R., Navarro-Serment, L., Paredis, C., Khosla, P.: Heterogeneous teams of modular robots for mapping and exploration. *Autonomous Robots* 8(2), 293–308 (2000)
6. Nouyan, S., Campo, A., Dorigo, M.: Path formation in a robot swarm: Self-organized strategies to find your way home. *Swarm Intelligence* 2(1), 1–23 (2008)
7. Vaughan, R., Stoy, K., Sukhatme, G., Matarić, M.: LOST: Localization-space trails for robot teams. *IEEE Transactions on Robotics and Automation* 18(5), 796–812 (2002)
8. Balch, T.: Reward and diversity in multirobot foraging. In: IJCAI-1999 Workshop on Agents Learning About, From and With other Agents, pp. 15–21. Morgann Kaufman, San Francisco (1997)
9. Klarer, P.: Simple 2-d navigation for wheeled vehicles. Technical report, Sandia Report SAND88-0540, Sandia National Laboratories, Livermore, CA (1988)
10. Feng, L., Borenstein, J., Everett, H.: Where am I? Sensors and Methods for Autonomous Mobile Robot Positioning. University of Michigan Press, Ann Arbor (1994)
11. Borenstein, J., Feng, L.: Measurement and correction of systematic odometry errors in mobile robots. *IEEE Transactions on Robotics and Automation* 12, 869–880 (1999)
12. Chong, K., Kleeman, L.: Accurate odometry and error modelling for a mobile robot. In: IEEE International Conference on Robotics and Automation, pp. 2783–2788. Robotics and Automation Society, NJ (1997)
13. Santos, F.C., Pacheco, J.M., Lenaerts, T.: Cooperation prevails when individuals adjust their social ties. *PLoS Computational Biology* 2(10), e140 (2006)
14. Goldberg, D., Matarić, M.J.: Interference as a tool for designing and evaluating multi-robot controllers. In: Proceedings of the Fourteenth National Conference on Artificial Intelligence, pp. 637–642. AAAI Press, Menlo Park (1997)

The Architecture of Ant-Based Clustering to Improve Topographic Mapping

Lutz Herrmann and Alfred Ultsch

Databionics Research Group, Dept. of Mathematics and Computer Science
University of Marburg
`{lherrmann,ultsch}@informatik.uni-marburg.de`

Abstract. This paper analyzes the popular ant-based clustering approach of Lumer/Faieta. Analysis of formulae unveils that ant-based clustering is strongly related to Kohonen’s Self-Organizing Batch Map. Known phenomena, e.g. formation of too many and too small clusters, can be explained due to that. Furthermore it is shown how topographic mapping of ant-based methods is substantially improved by means of a modified error function. This is demonstrated on few selected fundamental clustering problems.

1 Introduction

Techniques inspired by flocking behaviour of social insects have attracted a lot of attention in numerous research papers over the last decade due to the ability of simple interacting entities to exhibit sophisticated self-organization abilities. The idea behind ant-based clustering is that autonomous stochastic agents, called ants, move data objects on a low-dimensional regular grid such that similar objects are more likely to be placed on nearby grid nodes than dissimilar ones.

Most popular methods are based on the algorithm proposed by Lumer and Faieta [6]. Lumer/Faieta derivatives are known for at least two flaws: results highly depend on parametrization [1] and have been found to be “not competitive to the established methods of Multi-dimensional Scaling or Self-Organizing Maps” [3] in terms of topographic mapping.

This paper shows how to analyze ant-based clustering methods on basis of Self-Organizing Maps. A unifying representation for Lumer/Faieta and well-known Self-Organizing Batch Maps is introduced. Naive improvements for topographic mappings of ant-based methods are derived and experimentally verified.

2 Ant-Based Clustering

The method proposed by Lumer and Faieta [6] (here after LF algorithm) operates on a fixed regular low-dimensional grid $\mathbb{G} \subset \mathbb{N}^2$. A finite set of input samples X from a vector space with norm $\|\cdot\|$ is projected onto the grid by $m : X \rightarrow \mathbb{G}$. The mapping m is altered by autonomous stochastic agents, called ants, that move input samples $x \in X$ from $m(x)$ to new location $m'(x)$.

Ants might pick input samples when facing occupied nodes and drop input samples when facing empty nodes. Probabilities for picking and dropping are computed using the average similarity $\phi_x(i)$ between $x \in X$ and input samples located on the so-called perceptive neighbourhood around node $i \in \mathbb{G}$. The perceptive neighbourhood usually consists of $\sigma^2 \in \{9, 25\}$ quadratically arranged nodes at which the ant is located in the center. The set of input samples mapped onto the perceptive neighbourhood around $i \in \mathbb{G}$ is denoted with $N_x(i) = \{y \in X : y \neq x, m(y) \text{ neighbouring } i\}$. In this context, ϕ is referred to as *error function* since its minimization determines the ants' probabilistic modifications of mapping m .

$$\phi_x(i) = \frac{1}{\sigma^2} \sum_{y \in N_x(i)} \left(1 - \frac{\|x - y\|}{\alpha} \right) \quad (1)$$

LF-like methods lead to a local sorting of input samples on the grid in terms of similarities. Ants gather scattered input samples into dense crowds. In literature, it has been noticed that LF and its derivatives are prone to produce too many and too small clusters [1] [3]. For illustration see Figure 1.

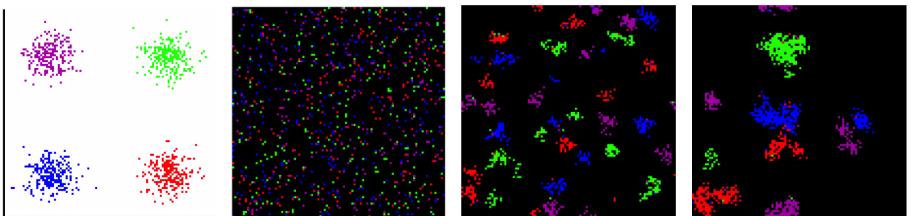


Fig. 1. Typical results [1] of LF algorithm from left to right: gaussian data with 4 clusters, initial mapping of data objects, dense clusters appear, finally too many clusters with topological defects have emerged

3 Analysis of Ant-Based Clustering by Means of Self-Organizing Batch Maps

Self-Organizing Batch Maps (Batch-SOM, [5]) are well-known artificial neural networks that consist of grid \mathbb{G} , codebook vectors $w_i \in \mathbb{R}^n, i \in \mathbb{G}$ and a mapping function $m : X \rightarrow \mathbb{G}$ with $m(x) = \arg \min_{i \in \mathbb{G}} \|x - w_i\|$. The codebook vectors are defined according to Equation 2 at which $h : \mathbb{G} \times \mathbb{G} \rightarrow [0, 1]$ denotes a time-dependent neighbourhood function. An update of $m : X \rightarrow \mathbb{G}$ leads to an update of codebook vectors $w_i, i \in \mathbb{G}$ and vice versa. This is how the Batch-SOM modifies mapping $m : X \rightarrow \mathbb{G}$. For details see [5].

In literature, two main types of Self-Organizing Maps (SOM) can be distinguished: first, SOM in which each codebook vector represents a single cluster. In contrast to that, SOM consisting of several thousands of codebook vectors

visualize structural features of the input space. These SOM are referred to as Emergent Self-Organizing Maps (ESOM, [9]).

$$w_i = \frac{\sum_{x \in X} h(m(x), i) \cdot x}{\sum_{x \in X} h(m(x), i)} \quad (2)$$

A meaningful error function for the Batch-SOM is derived from the quantization error $\|x - w_i\|$ because its minimization determines the update of $m : X \rightarrow \mathbb{G}$. Resolving the quantization error with Equation 2 leads to the error function Φ of the Batch-SOM. Φ_x represents the norm of averaged differences $x - y$ of grid-neighbouring input samples $y \in X$.

$$\Phi_x(i) = \frac{\left\| \sum_{y \in X} h(m(y), i) \cdot (x - y) \right\|}{\sum_{y \in X} h(m(y), i)} \quad (3)$$

In the following, the mechanism of picking and dropping ants is no longer subject of consideration. In [7] it was shown that collective intelligence can be discarded in LF systems, i.e. same results could be achieved without ants but using error function ϕ directly for probabilistic cluster assignments. This simplification is evident: over a period of time, randomly moving ants may select an arbitrary subset of input samples but re-allocation through picking and dropping depends on ϕ only. Probability of selection is the same on all input samples such that ants might be omitted in favor of any other subset sampling technique.

For the LF algorithm a meaningful neighbourhood function $h : \mathbb{G} \times \mathbb{G} \rightarrow [0, 1]$ is defined according to the perceptive neighbourhood of ants, i.e. $h(i, j)$ is 1 if $j \in \mathbb{G}$ is located in the perceptive neighbourhood of node $i \in \mathbb{G}$ and 0 elsewhere. This neighbourhood function allows to restate ϕ as follows:

$$\phi_x(i) = \frac{|N_x(i)|}{\sigma^2} \cdot \left(1 - \frac{\Phi'_x(i)}{\alpha} \right) \text{ with } \Phi'_x(i) = \frac{\sum_{y \in X} h(m(y), i) \cdot \|x - y\|}{\sum_{y \in X} h(m(y), i)} \quad (4)$$

The error function $\phi = \frac{|N|}{\sigma^2} (1 - \frac{\Phi'}{\alpha})$ of the LF algorithm incorporates the term Φ' that is a weighted sum of local input space distances. Obviously, Φ' measures the local stress of topographic mapping $m : X \rightarrow \mathbb{G}$, comparable to the error function Φ of the Batch-SOM. Φ' even acts as an upper limit to Φ since $\forall x \in X, i \in \mathbb{G} : \Phi_x(i) \leq \Phi'_x(i)$. Due to that Φ' is referred to as *topographic term* of the LF algorithm. The term $\frac{|N_x(i)|}{\sigma^2}$ estimates the output space density around grid node $i \in \mathbb{G}$. Therefore, it is referred to as *output density term* of the LF algorithm.

A unifying framework for analysis and assessment of Batch-SOM and LF exists by means of error functions Φ and ϕ . Both error functions are denoted by means of three functions: norm $\|\cdot\|$, neighbourhood $h : \mathbb{G} \times \mathbb{G} \rightarrow [0, 1]$, and mapping $m : X \rightarrow \mathbb{G}$. This leads to the following insights: The LF algorithm uses a fixed neighbourhood function with small radius, whereas Batch-SOM uses shrinking neighbourhood functions with large radiuses. The LF algorithm has a probabilistic update of mapping $m : X \rightarrow \mathbb{G}$, whereas Batch-SOM is

Table 1. Differences of Batch-SOM, LF-algorithm

	Batch-SOM	Lumer/Faieta
neighbourhood $h : \mathbb{G} \times \mathbb{G} \rightarrow [0, 1]$	large, shrinking	small, fixed
update of $m : X \rightarrow \mathbb{G}$	deterministic	probabilistic
searching for update of $m : X \rightarrow \mathbb{G}$	global \mathbb{G}	local $N_{m(x)} \subset \mathbb{G}$
error function	Φ	$\frac{ N }{\sigma^2}(1 - \frac{\Phi'}{\alpha})$
termination	cooling scheme	never

deterministic. The error function of the LF algorithm decomposes into an output density term $\frac{|N|}{\sigma^2}$ and a topographic term $1 - \frac{\Phi'}{\alpha}$. The topographic term is easily identified as a topographic distortion measure because of its relation to Batch-SOM error Φ . Therefore, the LF algorithm is easily convertible into a special case of Batch-SOM, and vice versa. For an overview of differences see Table 1.

4 Assessment and Improvement

Ant-based clustering methods following the LF scheme are prone to produce bad topographic mappings, e.g. too many, too small and topographically distorted clusters. If one regards LF as a derivative of the Batch-SOM, improvement of topographic mapping can easily be achieved.

Maximization of the *topographic term* $1 - \frac{\Phi'}{\alpha}$ corresponds to minimization of Φ' and Φ , too. This is known to produce sufficiently topography preserving mappings $m : X \rightarrow \mathbb{G}$, e.g. when using Batch-SOM [5].

In contrast to that, the *output density term* $\frac{|N|}{\sigma^2}$ has two major flaws. First, maximization of output space densities does not imply any preservation. Obtained mappings are, therefore, not related to the configuration of available clusters in the input space. In addition to that, the LF algorithm is not allowed to assign two or more objects to a single grid node (see Section 2) in order to prevent the mapped clusters from collapsing into a single grid node. Due to that, densities of input data are hardly preservable on grid \mathbb{G} .

In comparison with the topographic term, the output density term is much easier to maximize and, therefore, will distort the error ϕ . Accounting of output densities is prone to distort the formation of correct topographic mappings because it is responsible for additional local optima of ϕ . Future derivatives of the LF algorithm should maximize $1 - \frac{\Phi'}{\alpha}$ and minimize Φ' , respectively, in order to obtain better topographic mappings. For example, Figure 2 illustrates that traditional LF does not preserve looped cluster structures, in contrast to Emergent SOM and modified LF without density term.

The topographic term $1 - \frac{\Phi'}{\alpha}$ of the LF error depends on the shape of the neighbourhood function $h : \mathbb{G} \times \mathbb{G} \rightarrow [0, 1]$ (see Section 3). Usually, the neighbourhoods' sizes are chosen as $\sigma^2 \in \{9, 25\}$, i.e. the immediate neighbours. From the Self-Organizing Batch Map (Batch-SOM) it is known that the cooling scheme

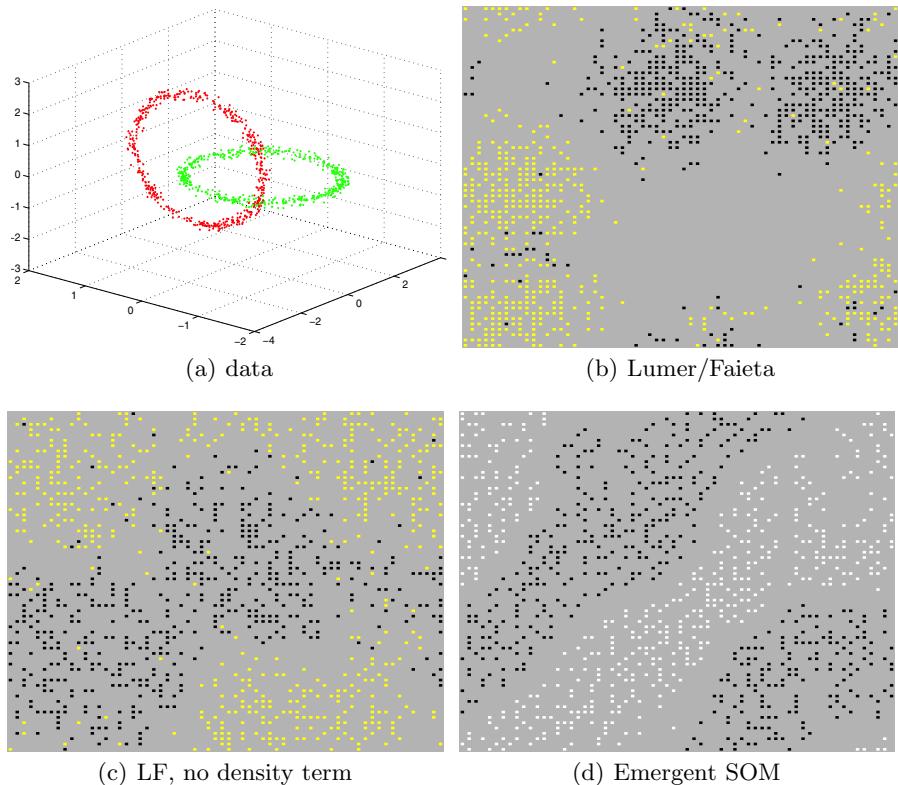


Fig. 2. Looped chainlink data from FCPS [8], data mapped on grid by several methods, only Emergent SOM and “LF without density term” enable formation of looped clusters with little effort

of the neighborhood radius influences the goodness for topographic mapping very strongly (see [4] for details). A bigger radius enables a more continuous mapping in the sense that proximities existing in the original data are visible on the grid. This is evident because smaller neighbourhoods are more likely to exclude parts of a cluster.

In order to prevent future LF derivatives from unsufficient topographic mappings, i.e. too many and too small clusters emerge during the training process, bigger neighbourhood radii are to be chosen. The ideal learning radius, however, remains a data-dependent quality.

5 Advanced Topographic Mapping

LF derivatives that do not account for output densities, usually will produce a SOM-like, equally distributed mapping of input samples (see Figure 2 for illustration). In this case, cluster retrieval cannot be achieved according to sparse regions dividing dense clusters on the grid.

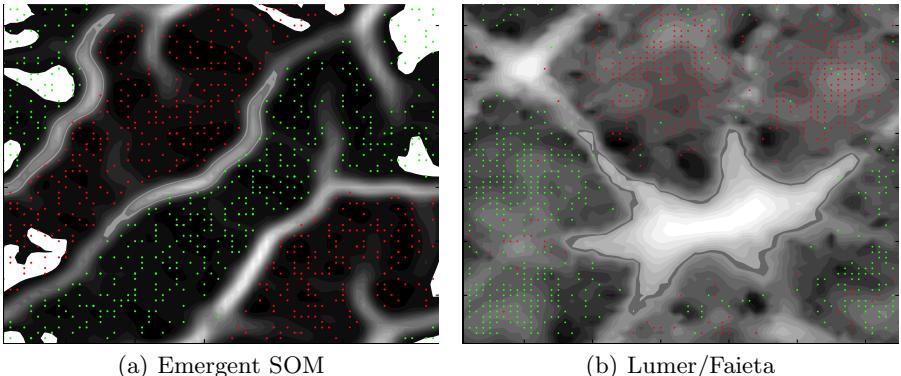


Fig. 3. U-Maps of looped chainlink data, input space distances depicted as gray levels
 (a) looped clusters, large distances occur between clusters (b) no looped clusters, large distances occur even inside clusters due to accounting for output density

A promising technique for cluster retrieval is based on so-called U-Maps [9]. Arbitrary projections from normed vector spaces onto grid \mathbb{G} are transformed into landscapes, so-called U-Maps. The U-Map technique assigns each grid node a height value that represents the averaged input space distance to its' neighbouring nodes and codebook vectors, respectively. Clusters lead to valleys on U-Maps whereas empty input space regions lead to mountains dividing the cluster valleys (see Figure 3 for illustration). The U* C cluster algorithm uses the so-called watershed transformation to retrieve cluster valleys on U-Maps (see [10] for details).

6 Experimental Settings and Results

In order to measure the distortion of a given topographic mapping method, a collection of fundamental clustering problems (FCPS) is used [8]. Each data set represents a certain problem that arbitrary algorithms shall be able to handle when facing unknown real-world data. Here, two versions of the Lumer/Faieta approach are tested on which one delivers the best topographic mapping: with and without accounting for output density.

A comprehensive overview on topographic distortion measurements can be found in [2]. Here, the so-called *minimal path length* (MPL) measurement is used. It is an easy-to-compute measurement that sums up input space distances of grid-neighbouring data objects and codebook vectors, respectively.

$$mpl = \sum_{x \in X} \frac{1}{|N_x|} \sum_{y \in N_x} \|x - y\| \quad (5)$$

Lower MPL values indicate less topographic distortion when moving on the grid and, therefore, a more trustworthy topographic mapping. Each algorithm is run several times with the same parametrization. MLP values indicate if accounting

Table 2. Topographic distortion measured by *minimal path length* method, mean values \pm standard deviation of each 100 experiments, p -values of Kolmogorov-Smirnov test indicate that “LF without density term” produces significantly smaller error values than traditional LF

data set	LF with density term	LF without density term	p-value
atom	161 ± 15.6	>	142 ± 6.2
chainlink	6.33 ± 0.33	>	6.19 ± 0.12
hepta	11.16 ± 0.66	>	9.86 ± 0.54
iris	11.8 ± 0.65	>	10.02 ± 0.57
target	6.69 ± 0.41	>	5.35 ± 0.33
2diamonds	3.86 ± 0.09	>	3.28 ± 0.10
wingnut	5.64 ± 0.32	>	5.07 ± 0.23

for output densities assists the formation of good topographic mappings, or not. All data sets from the FCPS collection were processed with the same parameters established in literature, i.e. $\alpha = 0.5$, $\sigma^2 = 25$, $k_1 = 0.3$ and $k_2 = 0.1$ on a 64×64 grid with 100 ants during 100000 iterations. The results can be found in Table 2. Accounting for output densities leads to increasing MPL values on an average, i.e. worsenings of topographic mappings. Significance has been confirmed using a Kolmogorov-Smirnov test on a $\alpha = 5\%$ level.

7 Discussion

Minimal path lengths (MPL), as proposed in Section 6, are well-known topographic distortion measures. The length of *paths* is normalized by the cardinality $|N_x|$ of the corresponding grid neighbourhood, i.e. the number of objects mapped onto the grid neighbourhood. This is supposed to decrease error values of locally dense mappings, as produced by traditional LF, because small radial neighbourhoods usually do not cover objects of another cluster, since locally dense mappings imply sparse dividing grid regions around clusters. Nevertheless, traditional LF produces bigger MPL errors than the modified LF that is not accounting for densities. We conclude that the topographic mapping quality is improved beyond our empirical evaluation.

8 Summary

To the best of our knowledge, this is the first work that shows how the LF algorithm by Lumer and Faieta [6] is related to Self-Organizing Maps [5]. The mechanism of picking and dropping ants was omitted in favor of a formal analysis of the underlying formulae and comparison with Kohonen’s Batch-SOM. It could be shown that a unifying framework for both methods does exist in terms of closely related topographic error functions. The LF algorithm is to be considered a probabilistic, first-class relative of the Batch-SOM. The behaviour of LF and derivatives becomes explainable on that unifying basis.

Ant-based clustering methods derived from LF exhibit poor clustering abilities because of distorted topographic mappings. Improvements of topographic mapping were derived by means of SOM architecture. Perceptive areas are to be increased, and accounting for density of mapped data is futile. The obtainable methods do not produce dense clusters any more but equally distributed, SOM-like mappings. Due to that, cluster are to be retrieved using U-Map technology. As predicted by our theory, an empirical evaluation showed on few clustering problems that not-accounting for density of mapped data improves the quality of topographic mapping despite of unfavorable settings.

References

1. Aranha, C., Iba, H.: The effect of using evolutionary algorithms on ant clustering techniques. In: Proc. Third Asian-Pacific workshop on Genetic Programming, pp. 24–34 (2006)
2. Goodhill, G.J., Sejnowski, T.J.: Quantifying neighbourhood preservation in topographic mappings. In: Proc. 3rd Joint Symposium on Neural Computation, pp. 61–82 (1996)
3. Handl, J., Knowles, J., Dorigo, M.: Ant-Based Clustering and Topographic Mapping. Artificial Life 12, 35–61 (2006)
4. Nybo, K., Venna, J., Kaski, S.: The self-organizing map as a visual neighbor retrieval method. In: Proc. of the Sixth Int. Workshop on Self-Organizing Maps (2007)
5. Kohonen, T.: Self-Organizing Maps. Springer Series in Information Sciences (2001)
6. Lumer, E., Faieta, B.: Diversity and adaption in populations of clustering ants. In: Proc. Third International Conference on Simulation of Adaptive Behaviour: From Animals to Animats, vol. 3, pp. 501–508 (1994)
7. Tan, S.C., Ting, K.M., Teng, S.W.: Reproducing the Results of Ant-Based Clustering Without Using Ants. In: Proc. IEEE Congress on Evolutionary Computation, pp. 1760–1767 (2006)
8. Fundamental Clustering Problem Suite,
<http://www.uni-marburg.de/fb12/datenbionik/data>
9. Ultsch, A., Mörchen, F.: U-maps: topographic visualization techniques for projections of high dimensional data. In: Proc. 29th Annual Conference of the German Classification Society (2006)
10. Ultsch, A., Herrmann, L.: Automatic Clustering with U*C. Technical Report. Philipps-University of Marburg (2006)

The Small World of Pheromone Trails

Paola Pellegrini and Andrea Ellero

Department of Applied Mathematics, University Ca' Foscari of Venice
Venice, Italy
`paolap@pellegrini.it, ellero@unive.it`

Abstract. In this paper we consider $\mathcal{MAX}\text{-}\mathcal{MIN}$ Ant System and Ant Colony System. They are generally recognized to be the best performing algorithms of the Ant Colony Optimization family. They are characterized by a quite different way for dealing with the pheromone trail. We propose an experimental analysis for observing whether this difference impacts significantly on the characteristics of the pheromone distributions produced during the runs. The results obtained are analyzed by using some concepts derived by the literature on small-world networks. It comes out that ants actually build small-world pheromone graphs during their runs. This behavior is interpreted here as a sort of decomposition of the instances tackled.

1 Introduction

In this paper we consider two Ant Colony Optimization (ACO) algorithms: $\mathcal{MAX}\text{-}\mathcal{MIN}$ Ant System (\mathcal{MMAS}) and ant colony system (ACS). They are recognized as the two best performing procedures of the Ant Colony Optimization family [1]. They are characterized by a pretty different behavior with respect to pheromone distribution and exploitation, as described in the following.

The aim of this paper is analyzing the effect of such a difference in the way the two algorithms deal with the distribution of the pheromone trail on solution components. In order to study this distribution, we use two elements coming from the literature on the small-world phenomenon. This concept originally belongs to the field of social sciences: A social network exhibits the small-world phenomenon if, roughly speaking, any two individuals in the network are likely to be connected through a short sequence of intermediate acquaintances [2,3,4]. Shifting this definition to mathematics, a small-world network is a graph in which nodes are neighbors of few others, but most nodes can be reached from every other in few steps. Moreover, the network is highly clustered, in the sense that if two vertices are neighbors of a third one, with high probability they will be connected. The small-world phenomenon is of particular interest here, since its definition allows us to interpret the evolution of a run (in terms of pheromone distribution) as some kind of splitting an instance in sub-instances. In this sense, each cluster represents a sub-instance: when it is reached, various ways are available for visiting the neighborhood, while when it is left only few paths are likely to be chosen.

In order to make this concept more explicit, and to observe the difference in the pheromone structure imposed by $\mathcal{MAX}\text{-}\mathcal{MIN}$ Ant System and Ant Colony System, the rest of the paper is organized as follows: in Section 2 the two algorithms are described, while in Section 3 some elements on small-world graphs are summarized. Sections 4 and 5 report the experimental setup and results, and Section 6 concludes the paper.

2 $\mathcal{MAX}\text{-}\mathcal{MIN}$ Ant System and Ant Colony System

Let us consider the generic combinatorial optimization problem to be solved as mapped on an edge-weighted graph $G = (N, E)$, with $N = \{1, 2, \dots, n\}$ set of nodes, and $E = \{(i, j) : i, j \in N\}$ set of edges. The graph is such that each solution to the combinatorial optimization problem corresponds to at least one path on the graph itself. The weights associated to the edges are such that the cost of a path (V_s) equals the cost of the associated solution s .

In $\mathcal{MAX}\text{-}\mathcal{MIN}$ Ant System [5], the state transition rule is named *random-proportional rule*. The probability of choosing the generic edge (i, j) is biased by the heuristic information η_{ij} and by the pheromone trail τ_{ij} . The relative weight of these values are controlled by two parameters of the algorithms, namely α and β .

After the activity of m ants, the pheromone evaporates on all the edges of the graph. The amount of this evaporation depends on the value of parameter ρ , $0 < \rho < 1$. Moreover, some pheromone is deposited on those used in a specific solution: One solution is used for the reinforcement. It is either the *iteration best* one, i.e. the best among the last m solutions built, or the *best so far* one, i.e. the best solution constructed. Finally, the pheromone is constrained between a lower and an upper bound τ_{min} and τ_{MAX} .

In ACS [6], the *pseudorandom-proportional rule* is applied: With probability q_0 , the step to perform is the one with the highest combination of the values of the heuristic information η_{ij} and by the pheromone trail τ_{ij} . Only the former is weighted by using an exponent β . With probability $1 - q_0$ the random proportional rule is applied (with $\alpha = 1$). The pheromone update is performed both after each iteration (*global update*) and after each ant has added one component to the solution under construction (*local update*). In the global pheromone update, the pheromone changes only on the edges belonging to the best so far solution. Its evaporation and deposit are controlled by parameter ρ . In the local pheromone update, after ant k has included edge (i, j) in the path, the pheromone level on (i, j) itself is updated, and the role of ρ is taken by another parameter ξ .

The two algorithms are not hybridized with any local search procedure. This choice is owed to the fact that we are interested in the nature of the algorithms themselves rather than in the absolute quality of the results. For a more detailed description of the procedures, we refer the reader to the specific literature.

3 Small-World Graphs

For observing the characteristics and the evolution of the pheromone trails on solution components, we analyze the structure of the graph we obtain considering all the nodes of the original one $G = (N, E)$, and the subset of edges on which the pheromone level is above the average: $G' = (N, E')$, $E' \subseteq E : \tau_{i,j} \geq |E|^{-1} \sum_{(i,j) \in E} \tau_{i,j}, \forall (i, j) \in E'$. G' will be referred to as pheromone graph. We study the structure of the pheromone graphs built during a run of the algorithm. This is done by using two measures that are typical of the literature on small-world networks [7], namely the characteristic path length (L) and the clustering coefficient (C). The former is defined as the average number of edges that must be traversed in the shortest path between pairs of nodes in the graph. The latter, instead, describes the neighborhood structure of the graph. In particular, if a node v is directly connected to k_v vertices, then this neighborhood defines a subgraph in which at most $k_v(k_v - 1)/2$ edges can exist. The clustering coefficient of v (C_v) is the ratio between this maximum and the number of edges that actually compose the subgraph. The average clustering coefficient is the average of this measure over all the nodes of the graph.

In order to state whether a graph G with n vertices exhibits the small-world property, its characteristic path length and clustering coefficient are compared to those of a random graph with the same number of nodes. In the latter $kn/2$ out of all possible $n(n - 1)/2$ edges are chosen at random with equal probability (k indicates the average degree of the vertices of G). In such a random graph, an asymptotic approximation of the two measures we are considering are: $L_{random} \sim \ln(n)/\ln(k)$, $C_{random} \sim k/n$ [8]. If $L \sim L_{random}$ and $C \gg C_{random}$, the graph under analysis is a small-world network. In words, it represents a clustered structure and, in average, a quite short path is sufficient to move from one node to another. In the following we will consider the ratios L/L_{random} (L ratio) and C/C_{random} (C ratio) in order to analyze the presence of the small-world property. Usually a graph is recognized to have this property if L ratio ~ 2.5 or lower and C ratio ~ 5 or higher [9,10,11,12].

To our aim, a small-world pheromone graph may be interpreted as the algorithm splitting an instance in sub-instances: computational resources are used for focusing on the clusters. Once one node of a cluster is reached, the possibility for visiting its neighbors are various, while, when a cluster is left, only few paths have high probability of being selected.

In the analysis proposed here, we use these concept for observing whether the pheromone graphs created during a run by *MAX-MIN* Ant System and Ant Colony System allow to think to this kind of problem decomposition.

4 Experimental Setup

The classical combinatorial optimization problem called traveling salesman problem (TSP) is considered in the experiments. Four sets of instances are tackled, with 200, 300, 400 and 500 nodes respectively.

Table 1. Values tested for the parameters

<i>MAX-MIN</i> Ant System							Ant colony system						
parameter values tested							parameter values tested						
m	50, 100, 150, 200	m	5, 10, 20, 50	β	2, 3, 4, 5, 6	β	2, 3, 4, 5, 6	ρ	0.02, 0.07, 0.12, 0.17, 0.22, 0.27, 0.32	ρ	0.16, 0.25, 0.23, 0.42, 0.5	q_0	0.75, 0.8, 0.85, 0.9
α	1, 2, 3, 4	ξ	0.1, 0.2										

Table 2. Values chosen for the parameters after the tuning procedure

Fairly explorative parameters <i>MMAS</i>							Fairly explorative parameters <i>ACS</i>								
n	m	ρ	β	α	n	m	ρ	β	α	n	m	ρ	β	q_0	ξ
200	50	0.17	4	1	400	50	0.22	4	1	200	20	0.33	4	0.75	0.1
300	50	0.22	4	1	500	50	0.32	4	1	300	20	0.25	4	0.75	0.1
Explorative parameters <i>MMAS</i>							Explorative parameters <i>ACS</i>								
n	m	ρ	β	α	n	m	ρ	β	α	n	m	ρ	β	q_0	ξ
200	150	0.07	3	1	400	150	0.12	3	1	200	50	0.42	4	0.75	0.1
300	150	0.07	3	1	500	150	0.12	4	1	300	20	0.33	4	0.8	0.1

For both *MAX-MIN* Ant System and Ant Colony System, we consider two configurations of parameters for each set of instances: a fairly explorative configuration and an explorative one. In order to fix them, we apply the tuning procedure F-Race [13]. The stopping criteria adopted is the total number of objective function evaluations (which is proportional to computational time). The reasoning is based on the fact that ACO algorithms, in order to get good performances, need to devote resources to both exploration and exploitation. If the available time is short, good parameters should imply a level of exploration such that some time is left to exploitation. On the other hand, if the computational time is longer, the exploration level can be higher, still allowing for exploitation [14]. Therefore, the maximum number of objective function evaluations allowed for a run is varied: we consider first a short run (25000 evaluations) in order to get a fairly explorative configuration, and then a long one (175000 evaluations) in order to get a more explorative one.

All the combinations of the values reported in Table 1 are considered for the tuning procedure. These values are chosen on the basis of the literature on the two algorithms [1]. Five hundred instances are used for the tuning phase. The values selected after the tuning procedure are reported in Table 2.

5 Experimental Results

In the experiments proposed, we apply the two algorithms presented in Section 2 to the traveling salesman problem. Ten TSP instances are used for each set, with 200, 300, 400 and 500 nodes respectively. The number of objective function evaluations considered as stopping criterion is 175000. Beside observing the quality of the results returned for each run, we consider the dynamics of the distribution of pheromone on solution components during the run itself. More in detail, for each run we focus on the pheromone graphs as defined in Section 3: Every ten iterations, we observe the graph having all the nodes of the original one, and

Table 3. Number of objective function evaluations after which the pheromone graph loses the small-world property (results are recorded each 10 iterations)

inst	Fairly explorative parameters											
	n=200			n=300			n=400			n=500		
	$\mathcal{M}\mathcal{M}\mathcal{A}\mathcal{S}$	ACS	$\mathcal{M}\mathcal{M}\mathcal{A}\mathcal{S}$	ACS	$\mathcal{M}\mathcal{M}\mathcal{A}\mathcal{S}$	ACS	$\mathcal{M}\mathcal{M}\mathcal{A}\mathcal{S}$	ACS	$\mathcal{M}\mathcal{M}\mathcal{A}\mathcal{S}$	ACS	$\mathcal{M}\mathcal{M}\mathcal{A}\mathcal{S}$	ACS
1	3500	13800	3500	12000	4000	25600	2500	88800				
2	3000	3200	2500	1200	3500	26800	2500	31200				
3	4000	10200	2500	2600	7500	71400	3000	15800				
4	3500	4300	3000	2600	3500	101400	3500	45600				
5	3000	4400	3000	19600	4000	44200	3000	33600				
6	4000	3000	2500	14000	4000	23200	2500	27500				
7	3500	1600	3500	24600	4500	44800	3000	16700				
8	4000	1200	4000	18600	5000	13200	3000	15200				
9	4000	7800	3000	18400	4000	92600	2500	25000				
10	4000	6000	4000	20000	4500	22000	3500	30000				

inst	Explorative parameters														
	n=200			n=300			n=400			n=500					
	$\mathcal{M}\mathcal{M}\mathcal{A}\mathcal{S}$	ACS	1	24000	43200	55500	19400	24000	56800	40500	20600				
2	27000	2000	31500	30900	18000	35200	22500	15200							
3	30000	27800	45000	22500	21000	32000	30000	14600							
4	34500	45300	37500	32400	22500	42200	24000	21700							
5	36000	1200	46500	35100	30000	15600	21000	15400							
6	36000	200	52500	43600	46500	35400	27000	9700							
7	36000	1200	51000	25000	33000	43400	25500	13900							
8	37500	2000	55500	30200	33000	43000	22500	21400							
9	28500	2700	49500	24000	27000	24100	22500	55800							
10	51000	12200	36000	26000	27000	16700	37500	8000							

only the edges on which the pheromone trail is above the average. For each of these pheromone graphs, we compute the C and L ratios (Section 3), and we verify whether we are in presence of a small-world network.

As for what the quality of the result is concerned, for all the runs, at the beginning Ant Colony System performs better than $\mathcal{M}\mathcal{A}\mathcal{X}\text{-}\mathcal{M}\mathcal{I}\mathcal{N}$ Ant System, while the latter ends up being the best at the end of the process. By the way, the number of objective function evaluations after which the value of the best solution returned by $\mathcal{M}\mathcal{M}\mathcal{A}\mathcal{S}$ is smaller than the one returned by ACS with the explorative parameters, is often higher than the value chosen as stopping criterion for these configurations.

After this observations, let us focus on the presence of small-world pheromone graphs during the runs. Interestingly, the presence of such networks can be observed in all the runs performed. Table 3 reports the number of objective function evaluations after which the small-world phenomenon weakens, *i.e.* after which the relevant values are no more above (or below, respectively) the thresholds reported in Section 3.

For visualizing the small-world presence in the pheromone graphs, we propose the results obtained for a representative instance with 400 nodes (instance number 2 in Table 3). Figure 1 reports the C and L ratios computed on the pheromone graph for the two algorithms and for the different sets of parameters. Moreover, in the upper part of the graphics (above the dashed line), the number and size of the clusters are represented: The size of the bullets is proportional to the average size of the clusters, and their y-coordinates correspond to the number of clusters that are present in the graphs. As it can be observed, clusterization is somehow present during the whole runs, but it is clearly higher at the beginning. A difference can be detected in the behavior of $\mathcal{M}\mathcal{A}\mathcal{X}\text{-}\mathcal{M}\mathcal{I}\mathcal{N}$ Ant System and Ant Colony System, regardless the set of parameters considered: at the very beginning the former works with one or two large clusters,

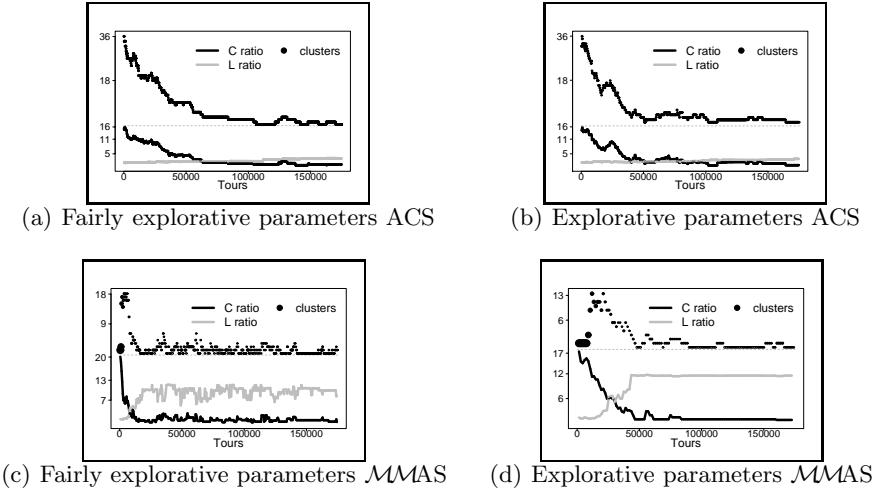


Fig. 1. C and L ratios, and number of clusters in one instance with 400 nodes

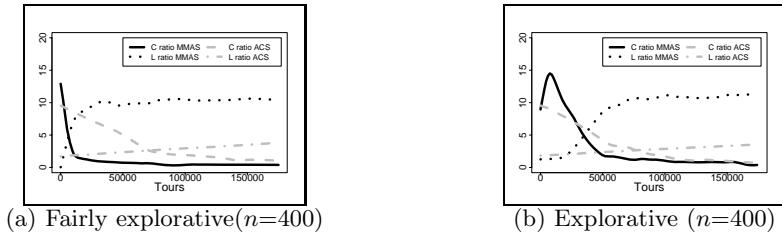


Fig. 2. Friedman super smoother computed on the observations of the C and L ratios in the set of instances with 400 nodes, for ACS and MMAS

which are then decomposed in smaller ones. The latter, instead, immediately creates several small groups and only progressively decreases their number. In some sense, then, MMAS chooses more carefully which nodes are to be grouped together, at the cost that, in some sense, it works for some time without any indication given by the pheromone. ACS, on the other hand, exploits from the first iteration the indirect communication that characterizes ACO algorithms, at the cost of possibly choosing the clusters in an imprecise way. According to the results obtained, ACS's strategy may be more powerful when the computational resources are very limited, while it is in general outperformed by MMAS's one when the time available is longer. For summarizing the trends followed, Figure 2 represents the Friedman super smoother [15] computed on the observations of the C and L ratios in the set of instances with 400 nodes, for ACS and MMAS. The pictures concerning the other sets are qualitatively similar: The difference implied by the sets of parameters chosen is much stronger for MMAS than for ACS. This can be read as a sort of greater ability of MAX-MIN Ant System to adapt to the conditions in which it has to work, and, on the other hand, as

a higher stability of Ant Colony System. Besides the difference in the duration of small-world effect, it is interesting to notice that both algorithms present the same type of behavior: At the beginning of a run, when the need for exploration is higher, the pheromone is distributed in a very particular way. The nodes are grouped in clusters. As a consequence, when an ant reaches one vertex of a cluster, several possibilities for choosing the following vertex have a probability significantly higher than zero. Intuitively, then, the order in which the nodes of a cluster appear in the different solutions will vary. When a cluster has been completely visited, instead, only very few edges are rich of pheromone.

As already mentioned, this characteristic of the runs can be seen as the algorithms splitting the instances in sub-instances, *i.e.* the clusters. This procedure is very effective, as the results on ACO algorithms applied to the TSP reported in the literature show. Moreover, it reproduces the algorithmic idea behind several procedure used for solving, among a large set of problems, the traveling salesman problem [16]: This idea consists in recursively breaking down a problem into two or more sub-problems of the same type, until these become simple enough to be solved directly.

6 Conclusions

In this paper we propose the application of MAX-MIN Ant System and Ant Colony System to the traveling salesman problem. The original contribution of the research consists in observing the characteristics of the pheromone distributions, beside the relative performance of the algorithms.

In the experiments proposed, during each run, we observed the graphs obtained by discarding, at different points in time, the edges with a very low level of pheromone which are less likely to be selected as far as alternative arcs exist.

These pheromone graphs ended up in exhibiting the small-world properties: simplifying, at the beginning of each run each of them appeared as a set of clusters linked by few chains. The presence of the small-world properties in the pheromone graphs at the beginning of each run, when the need of exploration is higher, and their vanishing later on, represents a strong link between the behaviors of the two ACO algorithms. This is particularly surprising given the different ways pheromone is treated in the two procedures.

We read this behavior as ants decomposing the instances in sub-problems, without even the implementer being aware of it. The solutions to the sub-problems are then combined to give a solution to the original problem. One of the main problems of *divide et impera* approaches lies in fixing a rule for selecting the groups to be tackled separately. ACO algorithms solve this problem for us, building their small-world on the instances they are given.

Of course, further experiments need to be done in order to state that these observations can actually be extended to the whole ACO family, or at least to a great part of it. Moreover, we need to deal with some other optimization problem. The next step will consist in extending the analysis to a problem in which ACO algorithms build oriented pheromone graphs, namely the Quadratic Assignment Problem.

References

1. Dorigo, M., Stützle, T.: *Ant Colony Optimization*. MIT Press, Cambridge (2004)
2. Pool, I., Kochen, M.: Contacts and influence. *Social Networks* 1, 1–48 (1978)
3. Milgram, S.: The small world problem. *Psychology Today* 2, 60–67 (1967)
4. Newman, M.E.J., Barabasi, A.L., WattsD., J.: *The Structure and Dynamics of Complex Networks*. Princeton University Press, Princeton (2006)
5. Stützle, T., Hoos, H.H.: Improving the Ant System: A detailed report on the MAX-MIN Ant System. Technical Report AIDA-96-12, FG Intellektik, FB Informatik, Technische Universität Darmstadt, Darmstadt, Germany (1996)
6. Dorigo, M., Gambardella, L.M.: Ant Colony System: A cooperative learning approach to the traveling salesman problem. *IEEE Transactions on Evolutionary Computation* 1(1), 53–66 (1997)
7. Watts, D.: Networks, dyanmics, and the small-world phenomenon. *The American Journal of Sociology* 105(2), 493–527 (1999)
8. Bollobas, B.: *Random Graphs*. Academic Press, London (1985)
9. Watts, D., Strogatz, S.: Collective dynamics of 'small-world' networks. *Nature* 393, 440–442 (1998)
10. Abe, S., Suzuki, N.: Small-world structure of earthquake network. *Physica A* 337, 357–362 (2004)
11. Montoya, J.M., Solé, R.V.: Small wolrd patterns in food webs. *Journal of theoretical biology* 214(3), 405–412 (2002)
12. Stam, C.J.: Functional connectivity patterns of human magnetoencephalographic recordings: a small-world network? *Neuroscience Letters* 355, 25–28 (2004)
13. Birattari, M.: The Problem of Tuning Metaheuristics as Seen from a Machine Learning Perspective. PhD thesis, Université Libre de Bruxelles, Brussels, Belgium (2004)
14. Pellegrini, P.: ACO: parameters, exploration and quality of solutions. PhD thesis, Department of Applied Mathematics, Università Ca' Foscari, Venice, Italy (2007)
15. Friedman, J.H.: A variable span smoother. Technical Report 5, Department of Statistics, Stanford University, Stanford, CA, USA (1984)
16. Halton, J.H., Terada, R.: A fast algorithm for the euclidean traveling salesman problem, optimal with probability one. *SIAM J. Comput.* 11(1), 28–46 (1982)

A Particle Swarm Optimization Algorithm for Multiuser Scheduling in HSDPA

Mehmet E. Aydin¹, Raymon Kwan¹, Cyril Leung², and Jie Zhang¹

¹ University of Bedfordshire, CWIND, Luton, UK

² The University of British Columbia, Vancouver, B.C., Canada

This paper briefs the problem of optimal multiuser scheduling in HSDPA. The modulation and coding schemes (MCSs), numbers of multicodes and power levels for all users are jointly optimized at each scheduling period, given that only limited Channel Quality Indicator (CQI) information, as specified in the HSDPA standard [1], is fed back to the BS. An integer programming model is proposed in order to provide a globally optimal solution to the multiuser scheduling problem. Due to the complexity of the globally optimal method, a swarm intelligence approach, namely particle swarm optimization (PSO), is subsequently proposed. The experimentations suggest that it potentially provides a near-optimum performance with significantly reduced complexity.

The problem is modeled in the following way. The CQI feedback value, q_i , from user i corresponds to the rate index that the user requests from the BS, and is associated with a required number of OVSF codes (multicodes) and downlink transmit power. Since the number of multicodes and transmit power are limited, the BS might not be able to simultaneously satisfy the bit rate requests for all users as described by $\{q_i, i = 1, \dots, N\}$. Thus, given the set $\{q_i, i = 1, \dots, N\}$, the BS must calculate a set of *modified CQIs*, $\{J_i, i = 1, \dots, N\}$, for all users by taking into account the power and number of multicodes constraints. This can be achieved by obtaining an approximated downlink Signal-to-Noise Ratio, $\hat{\gamma}_i$ based on the value of q_i [2]. The optimal scheduling problem **P1** can be expressed as

$$\mathbf{P1} : \quad \max_{\mathbf{A}, \overline{\phi}} \sum_{i=1}^N \sum_{j=0}^{J_i} a_{i,j} r_{i,j} \quad (1)$$

subject to $\sum_{j=0}^{J_i} a_{i,j} = 1$, $\sum_{i=1}^N \sum_{j=0}^{J_i} a_{i,j} n_{i,j} \leq N_{max}$, and $\sum_{i=1}^N \phi_i \leq N$, where $a_{i,j} \in \{0, 1\}$, and N_{max} is the maximum number of multicodes available for HSDPA at the BS. By making use of the appropriate mappings in system level model [3], the set of modified CQIs, $\{J_i, i = 1, \dots, N\}$, can be obtained as $J_i = \min(\max(\eta_i(\tilde{\gamma}_i^*, \phi_i), 0), q_{i,max})$ by assigning a power adjustment factor ϕ_i to user i , i.e. $\hat{\gamma}_i \mapsto \phi_i \hat{\gamma}_i$, where $q_{i,max}$ is the maximum rate index level due to the mobile capability of user i , $\tilde{\gamma}_i^* = \log_{10}(\hat{\gamma}_i)$, $\eta_i(\tilde{\gamma}_i^*, \phi_i) = \lfloor c_{i,1} (\tilde{\gamma}_i^* + 10 \log_{10} \phi_i) + c_{i,2} \rfloor$, and $0 \leq \phi_i \leq 10^{\left(\frac{q_{i,max} - (c_{i,1} \tilde{\gamma}_i^* + c_{i,2})}{10 c_{i,1}}\right)}$.

The terms $r_{i,j}$ and $n_{i,j}$ correspond to the achievable bit rate and the required number of multicodes associated with CQI value j , $j \in \{0, 1, \dots, K\}$, for user i , and are given in [1]. Note that J_i is the maximum allowable CQI value for

user i after the power adjustment. Depending on code availability, the assigned combination of MCS and the number of multicode may correspond to a bit rate that is smaller than that permitted by J_i . The objective of the above optimization problem is to select the values of the decision variables $\mathbf{A} = \{a_{i,j}\}$ and $\bar{\phi} = \{\phi_i\}$ at each TTI in order to maximize (1), subject to the above-mentioned constraints. Further information regarding the model and the implementation can be found in [4].

PSO has been applied in various continuous and discrete problems with good record of achievements [5]. This implementation is based on a standard PSO, but without the use of the velocity vector. The reason is to avoid double adaptation in computation process. Instead, the rule $x_{i,k}(t+1) = x_{i,k}(t) + \Delta x_{i,k}(t)$ is used, where t is the time index and as $\Delta x_{i,k}(t) = \delta w_{t+1}(c_1 r_1(y_{i,k}(y) - x_{i,k}(t)) + c_2 r_2(g_k(t) - x_{i,k}(t)))$.

This maximization model is to optimize two decision variables; $\bar{\phi}$ and \mathbf{A} , where $\bar{\phi}$ is a set of positive real numbers identifying the relative power level for each user, and \mathbf{A} is a set of binary variables which identifies the appropriate rate index, j , assigned to user, i . The aim is to obtain the most appropriate values for both set of variables such that the throughput of the system is maximized as described in (1).

For illustration purposes, we have simulated different scenarios for evaluating the performance of the proposed algorithms. We first consider $N = 2$, assuming that the mobiles are of category 10 and the values for $n_{i,j}$ and $r_{i,j}$ are obtained from [1]. We solved the problem instances generated as 3 different scenarios with a global optimization algorithm, which we denote as Joint Global Optimum (JGO). Then, a simple greedy (SG) algorithm is used to draw a minimum level of solution quality. This SG simply allocates resources to the users in decreasing order of their estimated SIR values. Finally, the same problem instances are solved using a PSO algorithm as summarized above. Simulation results indicate that the performances of PSO and JGO are very similar, and are consistently more superior than that of SG. The results show that a good throughput improvement can be achieved with JGO and PSO approaches over SG, and that the performances of JGO and PSO are very similar.

References

1. Universal Mobile Telecommunications Systems (UMTS); Physical Layer Procedures (FDD) (2007)
2. Kwan, R., Leung, C., Zhang, J.: A Power Assignment Scheme for Improving Outage Probability in HSDPA. In: IEEE Vehicular Technology Conference, Singapore (2008)
3. Motorola, Nokia: Revised CQI Proposal. Technical Report R1-02-0675, 3GPP RAN WG1 (2002)
4. Kwan, R., Aydin, M.E., Leung, C., Zhang, J.: Multiuser Scheduling in HSDPA using Simulated Annealing. In: Proc. of IEEE International Wireless Communications and Mobile Computing Conference, IWCMC 2008 (2008)
5. Kennedy, J., Eberhart, R., Shi, Y.: Swarm Intelligence. Morgan Kaufmann, San Mateo (2001)

AntLib v1.0: A Generic C++ Framework for Ant Colony Optimization

Francisco Javier Diego Martín, José Ángel González Manteca,
Ruth Carrasco-Gallego, and Javier Carrasco Arias

Escuela Técnica Superior de Ingenieros Industriales
Universidad Politécnica de Madrid, Spain

javier.diego@upm.es, jagonzalez@etsii.upm.es, ruth.carrasco@upm.es

This paper introduces AntLib, an Ant Colony Optimization (ACO) framework. C++ developed; it is a generic framework that can be applied with almost no adaptations to any combinatorial optimization problem. AntLib is a reusable object oriented framework, based on several well known design patterns [2].

AntLib is an extensible framework with a great amount of code ready to use, is efficient on execution leading to a high-speed performance, and it is very robust with loads of code checking. But the essential goal kept in mind in the AntLib design and development has been to provide a tool to the researcher that helps in the development phase of a combinatorial optimization system solved by ACO. AntLib offers a re-builder that reproduces a solution building process in order to achieve an easy debug and bug's repair tasks, a verifier to check whether the solution is properly built (to send it back to the rebuilder), and an analyst that studies the evolution of an algorithm, enables de parameters adjustment values and stagnation detection. AntLib is a generic framework, so almost no adaptation has to be done to solve different kinds of problems. Besides the object oriented paradigm, to achieve all the things mentioned before, AntLib makes use of templates, that can significantly reduce source code size and increases code flexibility without reducing safety type.

Architecture

The AntLib architecture is divided in four horizontal layers and two vertical layers as shown in Figure 1. The horizontal layers give more specialty as we climb, starting at the bottom with a technology layer (STL)[3], to the top layer that implements the ACO algorithms. The intermediate layers are the Library Foundations that defines the basic behaviours of the entire system, and ACO Framework that builds the basic framework to work with different algorithm types.

The vertical layers, services and monitoring, include the log files, exception processing, statistical performance, configuration files and utilities.

The basic classes for the framework are defined in 'Library Foundations' layer. The classes defined in this base layer are templates, what means the design is very generic, and can be applied to different problems and ACO algorithms without making any adjustment. The solver is a central class that manages the course

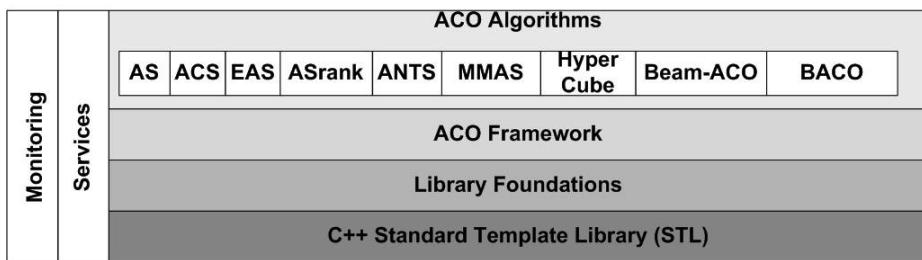


Fig. 1. Architecture in layers of AntLib

of an algorithm. An algorithm is composed by two operator types: structure operators, that performs all initialization steps, and iterative operators, which are executed once per iteration. There is also a context that encapsulates all the information that is needed on any algorithm step.

The 'ACO Framework' layer defines the structures that an ACO algorithm uses: the graph, the pheromone trails, and the total information. It also defines the data structures that ants use to construct the solutions. These structures are generic and can be easily adapted to different combinatorial optimization problems.

The 'ACO Algorithms' layer contains classes that implements many different published ACO algorithms. It is done mainly by pheromone trails updating classes. Due to its design, it is easy to change them for a given problem and compare their results.

Future Developments

New ACO algorithms will be added to AntLib, to enhance its genericity, making it able to be applied to more problems. AntLib will evolve towards a framework for hybrid metaheuristics development.

References

1. Dorigo, M., Stützle, T.: *Ant Colony Optimization*. MIT Press, Cambridge (2004)
2. Gamma, E., Helm, R., Johnson, R., Vlissides, J.: *Design Patterns: Elements of Reusable Object-Oriented Software*. Pearson Education, London (1995)
3. Musser, D.R., Saini, A.: *STL Tutorial and Reference Guide: C++ Programming with the Standard Template Library*. Addison-Wesley, Reading (1996)

Applying a Distributed Swarm-Based Algorithm to Solve Instances of the RCPSP

Paulo R. Ferreira Jr.^{1,2} and Ana L.C. Bazzan¹

¹ Instituto de Informática, UFRGS - Porto Alegre, RS, Brasil

{prferreiraj,bazzan}@inf.ufrgs.br

² Instituto de Ciências Exatas e Tecnológicas, Feevale - Novo Hamburgo, RS, Brasil

This paper addresses distributed task scheduling problems generalized as a distributed version of the Resource-Constrained Project Scheduling Problem (RCPSP) [1]. We apply and evaluate a novel approach for the RCPSP that is distributed and based on theoretical models of division of labor in social insects.

Our approach uses a probabilistic decision model, based on paradigms from swarm intelligence such as the tendency social insects have for performing certain tasks [2]. It has been implemented as an algorithm called Swarm-DRCPSP and was experimented in an abstract simulation environment. We show that Swarm-DRCPSP performs better than a distributed greedy algorithm, and that this performance is not much far from the best known solutions for the RCPSP, with the advantage of being computed in a distributed way.

Swarm-DRCPSP

In [3] the authors present a model where the interactions among members of the colony and the individual perception of local needs result in a dynamic distribution of tasks. Using our approach, agents decide which task to schedule based on that model. Furthermore, worker ants in several species retrieve preys or food items larger and heavier than a single individual capability using a mechanism of cooperative transport. We use this mechanism to handle simultaneous task allocation.

Experiments and Results

Empirical evaluations of Swarm-DRCPSP were conducted using three RCPSP instances available in the “j120” benchmark data set of the *Project Scheduling Problem Library* library [4]. Contrarily to our algorithm, the best known solution for those instances is not distributed. Thus, to analyze the performance of our algorithm in a fair way we have implemented a distributed greedy algorithm. This algorithm works almost as Swarm-DRCPSP, except that all perceived tasks are scheduled if resources are available. As mentioned, both Swarm-DRCPSP and the greedy algorithm are compared with the best approximate solution available for those three instances of the PSPLIB.

Figure 1 shows the comparison regarding the total time makespan achieved by using the Swarm-DRCPSP, the greedy algorithm, and the best heuristic solution, for three experimentation instances from the PSPLIB. The greedy algorithm

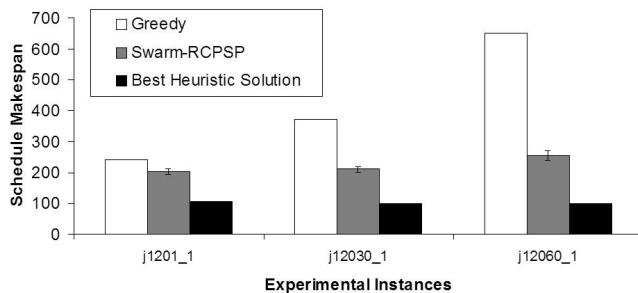


Fig. 1. Comparing the total time makespan of Swarm-DRCPSP, a greedy algorithm and the best heuristic solution, for 3 instances of the PSPLIB

achieves the worst result, obtaining schedules that are two to six times larger than the best approximate solution. Swarm-DRCPSP is much more efficient than the greedy algorithm, especially in the case of the most complex instance. The schedule makespans obtained by Swarm-DRCPSP are at worst twice that of the best known solution. However this performance is achieved despite its distributed computation and has a further advantage of not needing specific heuristics used by the best known solutions for the RCPSP.

Conclusion

Our approach for the DRCPSP is simple and effective. Empirical results show that the theoretical models of the division of labor in social insects may be successfully applied to the distributed task scheduling.

References

- Brucker, P., Drexl, A., Mohring, R., Neumann, K., Pesch, E.: Resource-constrained project scheduling: Notation, classification, models, and methods. *European Journal of Operational Research* 112(1), 3–41 (1999)
- Robison, G.E.: Regulation of division of labor in insect societies. *Annual Review of Entomology* 37, 637–665 (1992)
- Theraulaz, G., Bonabeau, E., Deneubourg, J.: Response threshold reinforcement and division of labour in insect societies. *Royal Society of London Series B - Biological Sciences* 265, 327–332 (1998)
- Kolisch, R., Sprecher, A.: Psplib – a project scheduling problem library. *European Journal of Operational Research* 96(1), 205–216 (1997)

bicACO: An Ant Colony Inspired Biclustering Algorithm*

Fabrício O. de França, Guilherme P. Coelho, and Fernando J. Von Zuben

Laboratory of Bioinformatics and Bioinspired Computing (LBiC)

University of Campinas (Unicamp), Campinas, SP, Brazil

{olivetti,gcoelho,vonzuben}@dca.fee.unicamp.br

A recent proposal developed to avoid some of the drawbacks presented by standard clustering algorithms is the so-called *biclustering* technique [1], which performs clustering of rows and columns of the data matrix simultaneously, allowing the extraction of additional information from the dataset. Since the biclustering problem is combinatorial, and ant-based systems present several advantages when dealing with this kind of problems [2], in this work we propose an ant-inspired algorithm for biclustering, which was named bicACO.

In order to adapt ACO to the biclustering problem some modifications must be made. First of all, each ant will have a separate pheromone table, mainly because the biclustering is a multimodal problem and requires that several different solutions be provided at the same time. In this way, indirect communication on bicACO occurs only among ants related to *the same bicluster*, but in different iterations of the algorithm. Additionally, each one of the k ants (the desired number of biclusters) starts from the full matrix and decides upon which row/column to remove, according to the probabilistic equation given in [2]. Each ant will repeat this step while the residue of the bicluster (defined in [1]) is above a given threshold δ or the volume is above a minimum value. The pheromone table will represent the probability (probabilistic factor of the algorithm) of removing a row or column from the full data matrix of dimension $n \times m$, so that the pheromone table will have a size of $n+m$. The heuristic term of the probabilistic equation is defined as the average residue of a given row/column. The probability of removing a given row/column will become higher if the corresponding average residue is high when compared to the average residue of other rows/columns.

After each ant has built a unique bicluster, the pheromone table must be updated (iterative improvement step) proportionally to the obtained results (as in [3]) and, in order to avoid stagnation, the pheromone value of the rows/columns that *remained* on the bicluster is reduced until $\tau_{ij} < \Delta\tau_i$ (Eq. 1). Once this lower limit is reached, the pheromone will automatically start to increase, in order to favor the exploration of the search space.

$$\begin{cases} \tau_{ij} = \tau_{ij} + \rho \cdot (\Delta\tau_i - \tau_{ij}) & \text{for all } j \in B_i \\ \Delta\tau_i = \frac{Vol_i/r_i}{\sum_i Vol_i/r_i} \end{cases}, \quad (1)$$

where Vol_i is the volume and r_i is the mean residue value of bicluster B_i .

* The authors would like to thank CAPES and CNPq for the financial support.

Table 1. Comparison of bicACO, CC and FLOC. The bicACO results are shown in the format (average \pm std. deviation), taken over 5 independent runs.

Algorithm	Avg.	Residue	Avg.	Volume
bicACO	176.15 ± 1.37		2725.61 ± 105.53	
CC		204.29		1576.98
FLOC		187.543		1825.78

The bicACO algorithm was compared to FLOC [4] and CC [1] algorithms on the Yeast microarray dataset [5], that contains 2,884 genes under 17 experimental conditions. The results of the latter two algorithms were taken from [4]. The proposed algorithm was executed over ten iterations to generate 100 biclusters ($k = 100$ ants) with a residue threshold of 180 and a pheromone decay rate (ρ) of 0.2. As reported in [4], the CC and FLOC algorithms were run in order to find 100 biclusters with a residue value smaller than 300. This difference on the adopted residue threshold is justified because bicACO stops immediately after reaching this objective, while FLOC and CC continue the optimization of the residue until they are not capable of improving the solutions anymore.

Table 1 presents the average results obtained by bicACO, over 5 independent runs, together with the results from the other two algorithms. In each run, the average residue and volume of the ants in the population were taken. It can be seen from this table that bicACO could find biclusters with an average residue smaller than the ones proposed by CC and FLOC, and with a much higher volume. This means that the generated biclusters present a much better coherence (given by the smaller residue) and may be more useful on post analysis (due to their higher volume). This fact is not a surprise, since the construction heuristic adopted in this work is based on the one adopted in CC, with the addition of the probabilistic factor (pheromone) and the iterative improvement.

References

- Cheng, Y., Church, G.M.: Biclustering of expression data. In: Proc. of the 8th Int. Conf. on Intelligent Systems for Molecular Biology, pp. 93–103 (2000)
- Dorigo, M.: Optimization, Learning and Natural Algorithms. PhD thesis, Politecnico di Milano, Italy (1992)
- de Fran  a, F.O., Von Zuben, F.J., de Castro, L.N.: Max min ant system and capacitated p-medians: Extensions and improved solutions. *Informatica* 29(2), 163–172 (2005)
- Jiong, Y., Haixun, W., Wei, W., Yu, P.S.: Enhanced biclustering on expression data. In: Proc. of the Third IEEE Symposium on Bioinformatics and Bioengineering, pp. 321–327 (2003)
- Cho, R., Campbell, M., Winzeler, E., Steinmetz, L., Conway, A., Wodicka, L., Wolfsberg, T., Gabrielian, A., Landsman, D., Lockhart, D., Davis, R.: A genome-wide transcriptional analysis of the mitotic cell cycle. *Molecular Cell* 2, 65–73 (1998)

Dynamic Routing and Travel Time Prediction with Ant Based Control

Bogdan Tatomir¹, Adriana-Camelia Suson², and Leon Rothkrantz²

¹ Quintiq, 's-Hertogenbosch, The Netherlands
bogdan.tatomir@quintiq.com

² Delft University of Technology, Delft, The Netherlands
{A.C.Suson,L.J.M.Rothkrantz}@ewi.tudelft.nl

At this moment the capacity of the highways is not sufficient to transport all car drivers without delay. Especially in the rush hours there are enormous traffic jams. In case of special events such as traffic accidents car drivers can be delayed by hours. The losses in time and money are enormous, so the problem of traffic congestion has a high priority.

To increase the capacity of the road network is not an option. The construction of new freeways in some areas is blocked by ecological, financial or political reasons. On the short term the only solution is to use the road network in an optimal way. Usually, there are many alternatives to travel from A to B. In case one route is blocked or delayed car drivers should be informed about alternatives routes. Nowadays, information about traffic jams is broadcasted via news on radio and TV, or via mobile phones, but normally only big congestions are reported and no alternative routes are suggested. Most current route planner devices have TMC (Traffic Message Channel) integrated but their information is updated only every 30 minutes which is not so fast. For extra payment special services provide personalized information which gets updated every few minutes. But all these services use the situation of the roads on a given moment and are unable to take the future into account. If all the cars are advised to take the same alternative route, soon also this one will get congested.

To compute alternatives routes it is necessary to have good prediction models of expected congestions and fast algorithm to compute the shortest path while being able to react to dynamic changes in the network caused by special incidents. In this paper we present a dynamic routing system founded on Ant Based Control (ABC). Starting from historical traffic data, ants are used to compute and predict the travel times along the road segments. They are finding the fastest routes not only looking to the past and present traffic conditions but also trying to anticipate and avoid future congestions.

Various shortest path algorithms are available for computing the optimal route. The most popular algorithm is Dijkstra's algorithm that has a runtime complexity of $O(n^2)$, where n is the number of nodes in the network. Many variations to the Dijkstra's algorithm such as bidirectional search and binary heap implementation have been proposed to improve its response time. An improved version of the Dijkstras algorithm is the A* algorithm [1], which is widely used in vehicle navigation.

Static routing algorithms like Dijkstra's algorithm only apply to central routing. To minimize the traveling time we need a decentralized routing algorithm which is able to adapt to the dynamic changes that take place in the traffic network. We found the answer in the real life behaviour of ants. Despite they are very simple insects, together, in a colony, they show what is called emergent behaviour and are able to accomplish complex tasks. Using the laying of pheromone ants are able to find the shortest path from their nest to a food source and vice-versa.

In [2] a dynamic vehicle routing system was introduced which uses the Ant Based Control algorithm (ABC-algorithm) for car navigation in a city. But the algorithm proved to be suitable for small networks and showed scalability problems on big networks as the one of the streets of a city. This problem was solved in [3] where the H-ABC, a scalable ant colony optimization algorithm was presented. Similar with the current navigation systems the previous algorithms looks only at the past and present and no future is considered.

To predict travel time, historical data (recorded from the ANWB) can be used, especially when the prediction horizon becomes further away. Apparently the load on the freeway network is almost the same on compatible days (same day of the week). Unfortunately, this has one considerable drawback: when unexpected events (like accidents) happen, a travel time prediction based on historical data will be completely wrong. Our approach is to use an Ant Based Control algorithm to approximate how many cars are expected to travel between two nodes in a specific time interval. Knowing the traffic flow, based on the speed/density relation of the traffic, we can make an estimate of the expected travelling time on a road segment. To test our concepts we modeled a part of the Dutch highway network. It consists of 58 nodes (highway intersections) and 84 bidirectional roads characterized by the number of lanes and the maximum allowed speed. Because of the big amount of data, we focused only on the morning period between 5:00 and 12:00. For almost 57% of the 3306 possible routes, faster alternatives were found by our routing system compared with a static one using with the Dijkstra's algorithm. A difference from 10 to 20 minutes was noticed in 12.16% of the situations. If we consider that The Netherlands is a small country and most of the selected routes are shorter than 150 km, using the dynamic routing system is a real benefit.

References

- Chabini, L., Lan, S.: Adaptations of the A* Algorithm for the Computation of Fastest Paths in Deterministic Discrete-Time Dynamic Networks. *IEEE Transactions on Intelligent Transportation Systems* 3(1), 60–74 (2002)
- Tatomir, B., Rothkrantz, L.J.M.: Dynamic traffic routing using Ant Based Control. In: International Conference on Systems, Man and Cybernetics IEEE SMC, pp. 3970–3975 (2004)
- Tatomir, B., Rothkrantz, L.J.M.: Hierarchical routing in traffic using swarm intelligence. In: The 9th International IEEE Conference on Intelligent Transportation systems, pp. 228–235 (2006)

Network Formation Using Ant Colony Optimization

Steven C. Oimoen, Gilbert L. Peterson, and Kenneth M. Hopkinson

Air Force Institute of Technology, Wright Patterson AFB, OH, USA
`steven.oimoen@afit.edu`

A significant area of research in the field of hybrid communications is the Network Design Problem (NDP) [1]. The NDP is an NP complete problem [1] that focuses on identifying the optimal network topology for transmitting commodities between nodes, under constraints such as bandwidth, limited compatible directed channels, and link and commodity costs. The NDP focuses on designing a flexible network while trying to achieve optimal flow or routing. If a link (or arc) is used, then an associated fixed cost of the edge is incurred. In addition, there is a cost for using the arc depending on the flow. The solution is a network topology connecting all of the nodes that minimizes the total system cost.

The specific topology control problem in this work is the Multi-Commodity Capacitated Network Design Problem (MCNDP) [1,2]. The MCNDP adds capacity limits for each arc to the uncapacitated NDP. A formal description of the MCNDP can be found in [1,2]. This paper presents a novel approach to solving this problem using the Ant Colony System (ACS) to construct the network topology and several heuristics for ACS to utilize to reduce computation time.

MCNDP Learning Using Ant Colony System (ACS)

Ant Colony Optimization (ACO) is a meta-heuristic technique that has been shown to be quite successful in solving many combinatorial optimization problems [3]. ACO mimics the foraging behavior of real ants, where ants deposit pheromone and over time identify the shortest paths from their nest to food.

The MCNDP solver uses ACS [4] to learn a graph similar to ACO learning of Bayesian Networks [5]. A network object contains lists of nodes and commodities. In addition, each node object contains a list of regular edges and potential edges. After all data is initialized, the list of edges used in the topology is empty. Each ant in building its solution uses the ACS selection strategy to select an edge from the list of all potential edges. The selected potential edges form the network topology. Ants iteratively select edges based on the pheromone on each edge, and the heuristic evaluation of the edge. Once all available potential edges have been explored, the ant stops searching.

The ACS algorithm was applied using two approaches, these are ACS Standard for the MCNDP (ACSS-MCNDP) and ACS Estimated for the MCNDP (ACSE-MCNDP). ACSS-MCNDP constructs a network topology and performs a full routing of commodities for each ant solution to evaluate the objective score of the proposed network. ACSS-MCNDP produces near optimal networks, at the

Table 1. Algorithm Comparative Analysis Summary

Number of Nodes	Max Flow	ACSS-MCNDP	ACSE-MCNDP
10	1083.39 (+/- 60.48)	870.06 (+/- 8.11)	920.61 (+/- 3.63)
15	5575.94 (+/- 6628.40)	2066.75 (+/- 22.90)	2377.14 (+/- 16.38)

cost of high run times due to performing routing. ACSE-MCNDP uses heuristics that replace the routing process in constructing the network. Routing only occurs at the end of the algorithm to evaluate the final network. Four heuristics were tested; a) fixed edge cost, b) sum of the fixed and variable edge costs, c) weighted sum of the fixed edge cost and capacity, and d) a weighted sum of the fixed edge cost and the edge value. The fixed edge cost is the cost associated with using a particular edge in the network (regardless of commodities using it). The variable edge cost is an average commodity cost associated with using an edge. The edge capacity is equivalent to its bandwidth and the edge value is the value of the commodity that would flow from that edge's source to its destination. Several combinations of the weighted cost heuristics were tested.

Both ACS algorithms found solutions with no dropped commodities for both 10-node and 15-node networks. For our ACSE approach we experienced mixed results. However, we did find that the 80 percent fixed cost and 20 percent edge capacity heuristic, although not consistent across all routing algorithms, closely approximated the ACSS solutions. Table 1 shows a comparative assessment of a Maximum Flow solution (1), our ACSS-MCNDP solution and our ACSE-MCNDP solution. The numbers in bold identify the best overall cost solutions.

References

1. Ahuja, R.K., Magnanti, T.L., Orlin, J.B.: Network Flows: Theory, Algorithms, and Applications. Prentice Hall, Upper Saddle River (1993)
2. Garner, R.: Heuristically Driven Search Methods for Topology Control in Directional Wireless Networks. Masters Thesis Air Force Institute of Technology, Wright-Patterson AFB, OH (2007)
3. Dorigo, M., Maniezzo, V., Colorni, A.: The Ant System: Optimization by a Colony of Cooperating Agents. IEEE Transactions of Systems, Man and Cybernetics Part B 26, 1–13 (1996)
4. Dorigo, M., Gambardella, L.M.: Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem. IEEE Transactions on Evolutionary Computation 1, 53–66 (1997)
5. de Campos, L., Fernandez-Luna, J., Gamez, J., Puerta, J.: Ant Colony Optimization for Learning Bayesian Networks. International Journal of Approximate Reasoning, 291–311 (2002)

On the Stability and the Parameters of Particle Swarm Optimization

Keiichiro Yasuda, Nobuhiro Iwasaki, and Genki Ueno

Department of Electrical and Electronic Engineering, Tokyo Metropolitan University
Tokyo, Japan
yasuda@eei.metro-u.ac.jp

In this paper, swarm activity is defined as the root mean square velocity of the particles in Particle Swarm Optimization (PSO). A new method for determining the numerical stability of PSO based on swarm activity was developed. Using the results of a numerical stability analysis of PSO, the search of conventional PSO methods is examined. From this analysis, issues related to diversification and intensification during the search can be explored.

Definition of Swarm Activity

In order to obtain a quantitative assessment of the search's progress, it is necessary to define a new index that quantitatively assesses the extent of diversification and intensification during a PSO search [1]. Swarm activity, Act , is defined in Eq.(1) as the root mean square velocity of particles, which can be used as an index of diversification and intensification during the PSO search.

$$Act = \sqrt{\frac{1}{mn} \sum_{i=1}^m \sum_{j=1}^n v_{ij}^2} \quad (1)$$

Numerical Stability Analysis of PSO

The algorithm for activity-based numerical stability analysis of PSO with parameters w and c ($= c_1 = c_2$) is given below.

Step 0: [Preparation]

Select the number of particles $2 \leq m \in R^1$, upper and lower limits of PSO parameters $0 < w_{\min} \in R^1$, $0 < w_{\max} \in R^1$, $0 < c_{\min} \in R^1$, $0 < c_{\max} \in R^1$, step width $0 < \Delta w \in R^1$, $0 < \Delta c \in R^1$, activity threshold $0 < Act_s \in R^1$, maximum iteration number T_{\max} .

Step 1: [Initialization]

$w = w_{\min}$, $c = c_{\min}$, $c_1 = c_2 = c$.

Step 2: [Assessment of Stability]

PSO is applied to the selected optimization problem. The swarm activity at $k = T_{\max}$ is set at

$$Act^{T_{\max}} = \sqrt{\frac{1}{mn} \sum_{i=1}^m \sum_{j=1}^n (v_{ij}^{T_{\max}})^2}.$$

If $Act^{T_{\max}} > Act_s$, the parameter values $\{w, c\}$ are considered unstable. Otherwise, they are considered stable.

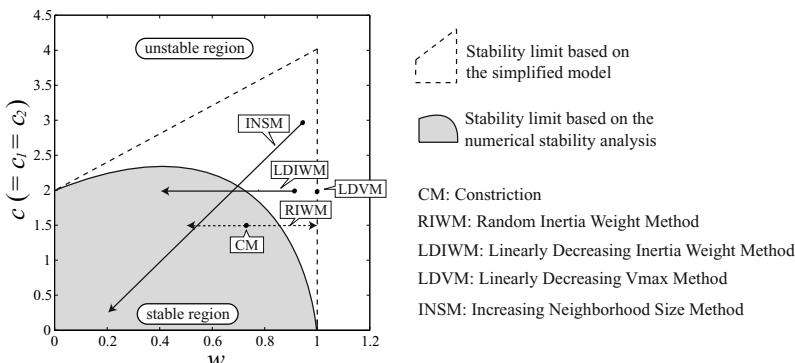


Fig. 1. Strategies for setting and adjusting the parameters in typical PSO methods

Step 3: [Inner Loop Termination Criterion]

If $c = c_{\max}$, proceed to Step 4. Otherwise, set $c = \min\{c + \Delta c, c_{\max}\}$, $c_1 = c_2 = c$, and return to Step 2.

Step 4: [Outer Loop Termination Criterion]

If $w = w_{\max}$, terminate. Otherwise, set $w = \min\{w + \Delta w, w_{\max}\}$, $c = c_{\min}$, $c_1 = c_2 = c$, and return to Step 2.

Based on a numerical stability analysis in the parameter plane presented in Fig.1, it is possible to make some observations about strategies for setting and adjusting the parameters in typical PSO methods with respect to stability.

Conclusions

The results presented in this paper can be summarized as follows:

- (1) Swarm activity was defined as the root mean square velocity of particles, and a new activity-based numerical stability analysis method was developed.
- (2) General and systematic strategies based on the activity-based numerical stability analysis were developed for setting and adjusting the parameters.

Reference

1. Yasuda, K., Ide, A., Iwasaki, N.: Stability Analysis of Particle Swarm Optimization. In: Proceedings of The Fifth Metaheuristics International Conference, MIC 2003 (2003)

Regional Traffic Assignment by ACO

Preliminary Results

Vittorio Maniezzo¹, Matteo Roffilli¹, Roberto Gabrielli², Alessandra Guidazzi²,
Manuel Otero³, and Rolando Trujillo³

¹ Dept. Computer Science, University of Bologna, Bologna, Italy
vittorio.maniezzo@unibo.it

² Province of Forlì - Cesena, Forlì and Cesena, Italy
³ University of Havana, Havana, Cuba

An established research line in ACO systems supports the intuition that ant algorithms are particularly fit for dynamic optimization problems because of their ability to construct an internal representation of the essential elements of the problem to solve, a representation which needs to be updated and not reconstructed when the instance changes.

Our work is about one such case arising in public authorities control and planning functions, where the availability of modeling tools which support both queries on the current state and simulation and optimization when forecasting is essential for territorial processes management and control. In the framework of a research line on road traffic simulation and forecast we implemented an ACO model targeted at real-world traffic flow simulation at a regional scale.

Traffic flow simulation is a well-known research topic, which has lead to a significant theoretical corpus and to effective marketed packages. Nevertheless, the current state of the art is not yet fully satisfactory for local government agencies, especially for medium sized ones, because of operational constraints and inherent rigidity of currently available models and software.

The core problem to face is the so-called *Traffic Assignment Problem* (TAP), which determines traffic flows on the roads given an Origin - Destination (OD) matrix describing the vehicle movements. The TAP can be modeled as a combinatorial optimization problem with a nonlinear objective function, and is particularly tricky because the flow levels on the roads are a function of the flows themselves (traffic flows divert from congested, i.e. high flow, roads).

ACO modeling, meaning with this the ability of ACO systems to construct a model of the instance to solve by means of trail distribution, is particularly suited for the TAP. The general flow distribution is dictated by Wardrop's principles [1], but physical road parameters are not enough to fully determine all chosen paths. This is where ACO trail become central: physical road properties are considered in the attractiveness computations, but the resulting congestion level is implicitly modeled in the trail distribution.

We report results obtained at a province (regional in the literature) level, in a setting which is rather common for this type of application, featuring a GIS-based road network, with a superimposed region zoning decomposing the area of interest into about 60 zones, each of which corresponds to a row/column of an OD

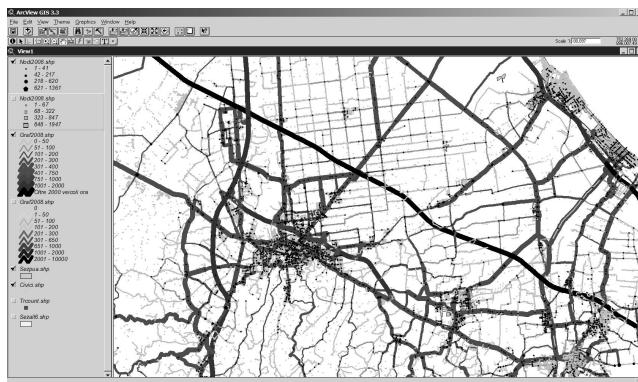


Fig. 1. Validated simulation results

matrix. Moreover, we had over 100 geocoded points, where actual traffic counts are available. The ants are called to determine flows so to minimize a global cost, which is given by the sum of the costs on arcs, where costs are a function of the flows themselves. Along the iterations, trail is added to divert flows from the paths that appear to be the least time ones, but that turn out not to be such once they are chosen by too many travelers. The general workings of our solution bears strong similarities with the Internet packet-routing applications [2], but a number of structural differences make a direct translation of packet-routing codes infeasible.

First results are available on a road network of the Italian province of Forlì - Cesena. To get actual data, we interfaced our code with the GIS of the Forlì-Cesena province, directly accessing the shapefiles it is based upon, thereby obtaining interoperability with the province's GIS. The network, along with validated simulation results, is shown in figure 1. Results validation is made against the data available as forecasts of traffic flows following the infrastructural scenarios for the next 20 years. The model implicit in the trail distribution permits to extend the assessment of non/physical road arcs, which can be determined when calibrating against known data, also to future scenarios, where calibration data is obviously unavailable. Thus, trail distribution does not only support a warm start on modified instances, but truly enables forecasting for future scenarios.

References

1. Wardrop, J.G.: Some theoretical aspects of road traffic research, vol. PART II, Vol.1, pp. 325–378. Institute of Civil Engineers, Palo Alto (1952)
2. Caro, G.D., Dorigo, M.: Antnet: Distributed stigmergetic control for communications networks. Journal of Artificial Intelligence Research 9, 317–365 (1998)

SwarmClass: A Novel Data Clustering Approach by a Hybridization of an Ant Colony with Flying Insects

Amira Hamdi^{1,2}, Nicolas Monmarché¹,
M. Adel Alimi², and Mohamed Slimane¹

¹ Laboratoire d'Informatique, Université François Rabelais de Tours, France
amira.hamdi7@yahoo.fr, {nicolas.monmarche,mohamed.slimane}@univ-tours.fr

² Department of Electrical Engineering, National School of Engineers (ENIS)
University of Sfax, Tunisia
adel.alimi@ieee.org

Swarm behaviors contribute to the resolution of very large number of difficult tasks thanks to simplified models and elementary rules [1]. This work claims a new swarm based behavior used for unsupervised classification. The proposed behavior starts from the ants collective sorting behavior as initially proposed by Lumer and Faieta [2] and overwrites it with additional behaviors inspired from birds and spiders. Our algorithm is then based on the existing work of [3], [4] and [2]. The proposed approach, called SwarmClass, outperforms previous ant-based clustering methods and resolve all its drawbacks by the introduction of simple swarm techniques and without the need of complex parameters configuration and prior information on classes' partition and distribution. Our proposed algorithm uses ants' segregation behavior to group similar objects together; birds' moving behavior to control next relative positions for a moving ant; and spiders' homing behavior to manage ants' movements when conflicting situations occur.

In SwarmClass, each object is placed in a cell on a discrete grid, which represents the environment of the ants. Each ant may pickup or drop an object according to a similarity function that measures the degree of similarity of an object with others neighboring objects. In SwarmClass, ants are moving in a new way: ants will cross and will follow a local behavioral rule that make them become closer or roll away, go on the same direction or not depending on the similarity of carried objects. More precisely, the ant can perform two kinds of move: a classical random move where the ant selects a random direction among the neighboring cells and an intelligent move (we should call it a swarm move). From this rules, ant groups will appear and will move together allowing to define hopefully better groups in data. Experimental results on synthetic and real data sets demonstrated the ability of SwarmClass to extract the correct number of clusters and to give better clustering quality compared to those obtained by a classical clustering algorithm like K-means and several ant based clustering algorithms [5,6,7].

With SwarmClass, we have demonstrated that it can be possible to improve stochastic algorithms for clustering by using hybridization of various swarm techniques.

Table 1. Comparative results obtained with 10-means and several ant-based clustering algorithms (values are averaged number of classes obtained over 50 runs)

database name	# of real classes	K-means	AntClass	AntTree	V-AntClust	FlyingInsects	SwarmClass
Art1	4	8.58	4.22	2.36	4.66	4.5	4.2
Art2	2	8.52	12.32	1.94	3.76	3.6	2.54
Art3	4	8.28	14.66	2.26	3.54	3.3	5.5
Art4	2	6.38	1.68	3.80	2.16	5.2	2.15
Iris	3	7.12	3.52	2.36	2.28	4.1	3.22
Thyroid	3	9.56	5.84	2.76	11.66	2.2	3.67
Soybean	4	8.82	1.60	3.90	4	5.6	4.1

This work suggests many perspectives. For the next step, we plan to use SwarmClass algorithm to tackle the image segmentation problem. Here, image segmentation can be viewed as a clustering problem which aims to partition the image into clusters such that the pixels within a cluster are as homogenous as possible whereas the clusters among each other are as heterogeneous as possible with respect to a similarity. The swarm behavior of the ants moving on the grid can be of interest in this context where many pixels should be manipulated together.

References

1. Bonabeau, E., Dorigo, M., Theraulaz, G.: *Swarm Intelligence: From Natural to Artificial Systems*. Oxford University Press, New York (1999)
2. Lumer, E., Faieta, B.: Diversity and Adaptation in Populations of Clustering Ants. In: Cliff, D., Husbands, P., Meyer, J., Wilson, S.W. (eds.) *Proceedings of the Third International Conference on Simulation of Adaptive Behavior (SAB)*, pp. 501–508. MIT Press, Cambridge (1994)
3. Bourjot, C., Chevrier, V., Thomas, V.: A new swarm mechanism based on social spiders colonies: from web weaving to region detection. *Web Intelligence and Agent Systems: An International Journal - WIAS* (2003)
4. Reynolds, C.W.: Flocks, herds, and schools: A distributed behavioral model. *Computer Graphics (SIGGRAPH 1987 Conference Proceedings)* 21(4), 25–34 (1987)
5. Monmarché, N.: On data clustering with artificial ants. In: Freitas, A. (ed.) *AAAI-1999 & GECCO-1999 Workshop on Data Mining with Evolutionary Algorithms: Research Directions*, Orlando, Florida (July 18 1999), pp. 23–26 (1999)
6. Labroche, N., Monmarché, N., Venturini, G.: AntClust: Ant Clustering and Web Usage Mining. In: Cantu-Paz, E. (ed.) *GECCO 2003. LNCS*, vol. 2723, pp. 25–36. Springer, Heidelberg (2003)
7. Azzag, H., Monmarché, N., Slimane, M., Venturini, G., Guinot, C.: AntTree: A new model for clustering with artificial ants. In: *IEEE Congress on Evolutionary Computation*, Canberra, 8-12 december 2003, vol. 4, pp. 2642–2647. IEEE Press, Los Alamitos (2003)

The Differential Ant-Stigmergy Algorithm for Large Scale Real-Parameter Optimization

Peter Korošec and Jurij Šilc

Computer Systems Department, Jožef Stefan Institute, Ljubljana, Slovenia
{peter.korosec,jurij.silc}@ijs.si

The optimization problem treated here is to find \mathbf{x} , which optimizes cost function $F(\mathbf{x})$, where $\mathbf{x} = \{x_1, x_2, \dots, x_D\}$ is a set of real parameters and D represents the dimension of the cost function. Domains of the real parameters are defined by their lower and upper bounds: $x_j^{low}, x_j^{upp}; 1 \leq j \leq D$. In this paper we consider only high-dimensional cost functions with real parameters; D up to 1000.

Proposed Approach

In [3] our ACO-based continuous optimization method, so-called Differential Ant-Stigmergy Algorithm (DASA), is presented. So far, the DASA was already tested on low-dimensional cost functions and compared to the state-of-the-art continuous optimization algorithms. It has proved to be an effective and efficient algorithm for this kind of problems [4].

The optimization consists of an iterative improvement of the temporary best solution, \mathbf{x}^{tb} , by constructing an appropriate path \mathbf{p} . By applying \mathbf{p} to \mathbf{x}^{tb} new solutions are produced.

First a solution \mathbf{x}^{tb} is randomly chosen and evaluated. Then a search graph is created and an initial amount of pheromone is deposited on search graph according to the Cauchy probability density function $C(z) = \frac{1}{s\pi(1+(\frac{z-l}{s})^2)}$, where l is the location offset and $s = s_{global} - s_{local}$ is the scale factor. For an initial pheromone distribution the standard Cauchy distribution ($l = 0$, $s_{global} = 1$, and $s_{local} = 0$) is used and each parameter vertices are equidistantly arranged between $z = [-4, 4]$.

There are m ants in a colony, all of which begin simultaneously from the *start* vertex. Ants use a probability rule to determine which vertex will be chosen next. The rule is the same as in Simple-ACO with $\alpha = 1$. The ants repeat this action until they reach the ending vertex. For each ant, path \mathbf{p} is constructed. If for some predetermined number of tries we get $\mathbf{p} = \mathbf{0}$ the search process is reset by randomly choosing new \mathbf{x}^{tb} and pheromone re-initialization. New solution \mathbf{x} is constructed and evaluated with a calculation of $F(\mathbf{x})$.

The current best solution, \mathbf{x}^{cb} , out of m solutions is compared to the temporary best solution \mathbf{x}^{tb} . If \mathbf{x}^{cb} is better than \mathbf{x}^{tb} , then \mathbf{x}^{tb} values are replaced with \mathbf{x}^{cb} values. In this case s_{global} is increased (in our case for 1%) and pheromone amount is redistributed according to the associated path. Furthermore, if new \mathbf{x}^{tb} is better than \mathbf{x}^{cb} , then \mathbf{x}^{cb} values are replaced with \mathbf{x}^{tb} values. So, global

best solution is stored. If no better solution is found s_{global} is decreased (in our case for 3%).

Pheromone evaporation is defined by some predetermined percentage ρ . The probability density function $C(z)$ is changed in the following way: $l \leftarrow (1 - \rho)l$ and $s_{\text{local}} \leftarrow (1 - \rho)s_{\text{local}}$.

The whole procedure is then repeated until some ending condition is met.

Performance Evaluation

The DASA algorithm was tested on six CEC'2008 special session benchmark functions [5]. Functions were optimized for three dimensions $D = 100$, $D = 500$, and $D = 1000$ with 25 runs on each function.

The DASA is compared to state-of-the-art algorithm for global optimization over continuous space. A differential evolution (DE) turned out to be one of the best metaheuristic for such problems. There are many variations of DE. We decided to use the algorithm jDEdynNP-F introduced in [1,2]. The jDEdynNP-F is a self-adaptive Differential Evolution algorithm where control parameters are self-adaptive and a population size reduction method.

The experimental results show that the DASA is capable of solving large scale real-parameter optimization problems. It outperformed the included differential evolution type algorithm in convergence on all test functions and also obtained better solutions on some test functions.

These results confirm that the DASA is effective to small and large scale optimization problems.

References

1. Brest, J., Sepesy Maučec, M.: Population size reduction for the differential evolution algorithm. *Appl. Intell.* 29 (2008)
2. Brest, J., Zamuda, A., Bošković, B., Sepesy Maučec, M., Žumer, V.: High-dimensional real-parameter optimization using self-adaptive differential evolution algorithm with population size reduction. In: Proc. IEEE World Congress on Computational Intelligence, Hong Kong (2008)
3. Korošec, P.: Stigmergy as an approach to metaheuristic optimization. PhD Thesis, Jožef Stefan International Postgraduate School, Ljubljana, Slovenia (2006)
4. Korošec, P., Šilc, J., Oblak, K., Kosel, F.: The differential ant-stigmergy algorithm: An experimental evaluation and a real-world application. In: Proc. IEEE Congress on Evolutionary Computation, Singapore, pp. 57–164 (2007)
5. Tang, K., Yao, X., Suganthan, P.N., MacNish, C., Chen, Y.P., Chen, C.M., Yang, Z.: Benchmark functions for the CEC 2008, special session and competition on large scale global optimization. Technical Report NCL-TR-2007012. Nature Inspired Computation and Applications Laboratory, University of Science and Technology of China (2007), <http://nical.ustc.edu.cn/cec08ss.php>

Author Index

- Adams, Carl 323
Al-Ani, Ahmed 1
Al-Jumaily, Adel 1
Albert, Patrick 84
Alimi, M. Adel 411
AlSukker, Akram 1
Aydin, Mehmet E. 395

Barman, Sarah 251
Bautista, Joaquín 331
Bazzan, Ana L.C. 399
Benedettini, Stefano 179
Birattari, Mauro 307, 347
Blum, Christian 25, 299
Brocco, Amos 275
Bruckstein, Alfred M. 72
Brutschy, Arne 96

Campo, Alexandre 307, 371
Carmona, Pablo 13
Carrasco-Gallego, Ruth 397
Carrasco Arias, Javier 397
Castro, Juan Luis 13
Çelikkanat, Hande 108
Cheung, Eugene 267
Chica, Manuel 331
Christensen, Anders Lyhne 259
Coelho, Guilherme P. 401
Cordón, Óscar 331

Damas, Sergio 331
Decugnière, Antoine 307
de França, Fabrício O. 401
del Rey Zapatero, Marco 347
Di Caro, Gianni A. 211
Diego Martín, Francisco Javier 397
Di Gaspero, Luca 155, 179
Dobata, Shigeto 283
Dorigo, Marco 259, 307, 347, 371
Ducatelle, Frederick 211

Elor, Yotam 72
Ellero, Andrea 387
Ermetici, Andrea 155

Farooq, Muddassar 315
Farrelly, Caroline 120
Fernandes, Carlos 339
Ferreira Jr., Paulo R. 399
Forestiero, Agostino 291
Francès, Guillem 25
Frapolli, Fulvio 275
Freitas, Alex A. 48
Fujisawa, Ryusuke 283

Gabrielli, Roberto 409
Gambardella, Luca M. 211
Gökçe, Fatih 108
González Manteca, José Ángel 397
Gordon, Noam 72
Greene, Casey S. 37
Guesgen, Hans W. 243
Guidazzi, Alessandra 409
Gutiérrez, Álvaro 371

Hamdi, Amira 411
Hernández, Hugo 25
Herrmann, Lutz 379
Hirsbrunner, Béat 275
Hopkinson, Kenneth M. 405
Huepe, Cristián 108

Imamura, Hikaru 283
Iwasaki, Nobuhiro 407
Izzo, Dario 347

Jacob, Christian 191
Johnson, Colin G. 48

Kell, Douglas B. 120
Kentzoglaniakis, Kyriakos 323
Khayam, Syed Ali 315
Khemka, Namrata 191
Khichane, Madjid 84
Khushaba, Rami N. 1
Knowles, Joshua 120
Köchel, Peter 355
Korošec, Peter 413
Kriesel, David M.M. 267
Kronfeld, Marcel 203

- Kubota, Daisuke 283
Kwan, Raymon 395
Laredo, Juan Luís 339
Lässig, Jörg 355
Leung, Cyril 395
Li, Li 219
Lipson, Hod 267

Makowski, Armand M. 167
Maniezzo, Vittorio 409
Mastroianni, Carlo 291
Matsuno, Fumitoshi 283
Merelo, Juan Julián 339
Merkle, Daniel 96, 299
Middendorf, Martin 96, 299
Monekosso, Dorothy 251
Monmarché, Nicolas 411
Moore, Jason H. 37
Mora, Antonio Miguel 339
Moscardini, Alfredo 235
Mullen, Robert J. 251

Neumann, Frank 132

O'Grady, Rehan 259
Oimoen, Steven C. 405
Osée, Michel 307
Otero, Fernando E.B. 48
Otero, Manuel 409

Pellegrini, Paola 387
Peña, Jorge 144
Pereira, Jordi 331
Peterson, Gilbert L. 405
Pincioli, Carlo 307, 347, 371
Poole, Matthew 323
Poulain, Benjamin 307

Qiao, Fei 219

Rabanal, Pablo 60
Ramos, Vitorino 339
Ranon, Roberto 155
Remagnino, Paolo 251
Riddle, Patricia J. 243
Rodríguez, Ismael 60

Roffilli, Matteo 409
Roli, Andrea 179
Rosa, Agostinho 339
Rothkrantz, Leon 403
Rubio, Fernando 60

Şahin, Erol 108
Saleem, Muhammad 315
Santos, Francisco C. 371
Scheidler, Alexander 96, 299
Šilc, Jurij 413
Sitti, Metin 267
Slimane, Mohamed 411
Solnon, Christine 84
Spezzano, Giandomenico 291
Sudholt, Dirk 132
Suson, Adriana-Camelia 403

Tartini, Bruno 307
Tatomir, Bogdan 403
Thiem, Stefanie 355
Trujillo, Rolando 409
Tsutsui, Shigeyoshi 363
Tuci, Elio 347
Turgut, Ali Emre 108

Ueno, Genki 407
Ultsch, Alfred 379
Uthus, David C. 243

Venables, Harry 235
Vinko, Tamas 347
Von Zuben, Fernando J. 401

Weiss, Christian 203
White, Bill C. 37
Wilkin, Paul 251
Witt, Carsten 132
Wu, Qidi 219

Yasuda, Keiichiro 407

Zell, Andreas 203
Zhan, Zhi-hui 227
Zhang, Jie 395
Zhang, Jun 227