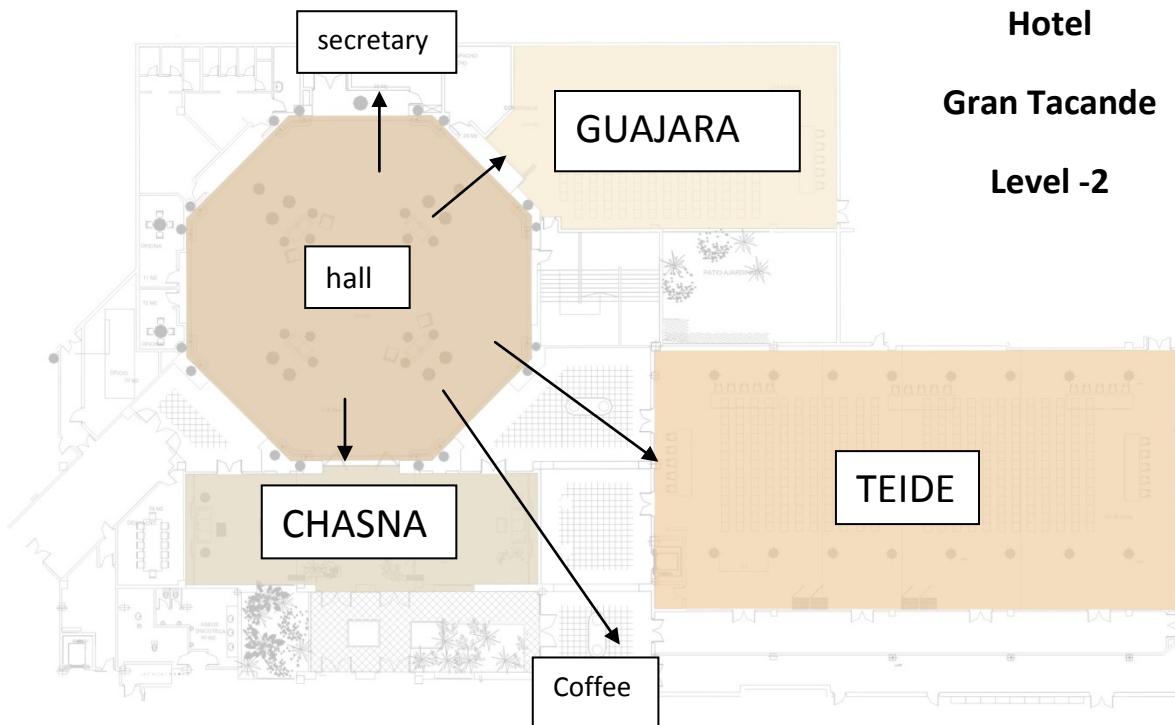


International Network Optimization Conference

May 20-22, 2013
Tenerife, Spain

Program

	Monday May 20 th	Tuesday May 21 st	Wednesday May 22 nd
8:45 – 9:00	Opening		
9:00 – 9:50	Plenary Talk	Plenary Talk	Plenary Talk
10:00 – 11:30	Session M1	Session T1	Session W1
11:30 – 12:00	Coffee Break	Coffee Break	Coffee Break
12:00 – 13:30	Session M2	Session T2	Session W2
13:30 – 15:30	Lunch	Lunch	Lunch
15:30 – 17:00	Session M3	Excursion	Session W3
17:00 – 17:30	Coffee Break		Coffee Break
17:30 – 19:00	Session M4		Session W4
19:00 – 20:00			
20:00 – 22:00			Social Dinner



There is a secretary office close to the conference rooms. Participants can get the badges and other material on

- Sunday, 18:00 - 20:00,
- Monday, 08:15 - 13:30 and 15:30 - 18:00
- Tuesday and Wednesday, 09:00 - 13:30.

Each session contains 4 talks. To allow attending talks in parallel sessions, we kindly request each speaker to use the assigned slot of 22.5 minutes. The chairperson of a session should start each talk in the expected time according to this program. Hence, there will be an idle slot in a session when a speaker does not show up.

Each conference room has a computer running Windows, with Acrobat reader and with PowerPoint. The speakers of a session should bring their slides in a USB device and save the file with the slides in this computer before the beginning of the session.

Coffee-breaks, lunches and social dinner are in the conference hotel (Gran Tacande):

- Coffee-breaks are near the conference rooms (level -2)
- All lunches are in Restaurant “Zurrón”
- The social dinner is in Restaurant “Bocana”

The INOC2013 badge with the identification provided by the organization is mandatory in the activities, even for accompanying persons.

The excursion will leave from outside the conference hotel (Gran Tacande) in three green buses (by TITSA) between 15:15 and 15:30. Please, be ready at the lobby of the hotel by 15:15. Since the Observatory Izaña is at 2390 meters above sea level, the weather could be cold. People under year old are not allowed in the Observatory.

Monday May 20th

8:45 – 9:00 Opening

9:00 – 9:50 Plenary Talk [Room TEIDE] (Chairman: Walid Ben-Ameur)

Flows Over Time: Modeling Network Routing Problems Including a Temporal Dimension

Martin Skutella

10:00 – 11:30 Session M1a [Room TEIDE] (Chairman: Michael Poss)

Computing the probabilities of small component sizes in large networks

Paolo Serafini

A polynomial formulation for the stochastic maximum weight forest problem

Pablo Adasme, Rafael Andrade, Marc Letournel, Abdel Lisser

Stochastic Survivable Network Design Problems

Bernd Zey, Ivana Ljubić, Petra Mutzel

Robust combinatorial optimization with uncertain cost

Michael Poss

10:00 – 11:30 Session M1b [Room GUAJARA] (Chairman: Andrei V. Orlov)

Dantzig-Wolfe Reformulation for the Network Pricing Problem with Connected Toll Arcs

Alessia Violin, Bernard Fortz, Martine Labb  

Pricing techniques for self regulation in Vehicle Sharing Systems

Ariel Waserhole, Vincent Jost, Nadia Brauner

(k, c) - coloring via Column Generation

E.Malaguti, I. M  ndez-D  az, J.J. Miranda-Bront, P.Zabala

Pricing in telecommunication networks and bilevel optimization

Andrei V. Orlov

10:00 – 11:30 Session M1c [Room CHASNA] (Chairman: Maria Grazia Scutell  )

Strategic planning in LTL logistics - increasing the capacity utilization of trucks

J. Fabian Meier, Uwe Clausen

A planning and routing model for patient transportation in health care

Jessica Raffaelli, Alberto Coppi, Paolo Detti

Optimal Client-Server Configuration of Mobile Ad-Hoc Networks

Luigi De Giovanni, Claudio E. Palazzi

Home Care optimization: impact of pattern generation policies on scheduling and routing decisions

Maria Grazia Scutell  , Paola Cappanera

Monday May 20th

11:30 – 12:00 Coffee Break

12:00 – 13:30 Session M2a [Room TEIDE] (Chairman: Martin Tieves)

Checking Feasibility in Stationary Models of Gas Transportation Networks - Methodological Foundation

Claudia Stangl, Ralf Gollmer, Rüdiger Schultz

Checking Feasibility in Stationary Models of Gas Transportation Networks - Case Study

Ralf Gollmer, Rüdiger Schultz, Claudia Stangl

A Study of the Connectivity Over Time Behavior of On-Demand Ad Hoc Path Selection Schemes

David Raz, Shira Levy

Extended Cutset Inequalities for the Network Power Consumption Problem

Martin Tieves, Arie Koster, Truong Khoa Phan

12:00 – 13:30 Session M2b [Room GUAJARA] (Chairman: Hande Yaman)

Routing and Scheduling Decisions in the Hierarchical Hub Location Problem

Okan Dükkancı, Bahar Yetiş Kara

Hub Location Problem under Competition: ($r|Xp$) hub-medianoid and ($r|p$) hub-centroid problems

A.İrfan Mahmutoğulları, Bahar Y. Kara

Heuristics for solving a capacitated multiple allocation hub location problem with application in German wagonload traffic

Julia Sender, Uwe Clausen

A Branch-and-cut Algorithm for the Hub-Cycle Location Problem

Hande Yaman, Inmaculada Rodríguez Martín, Juan José Salazar González

12:00 – 13:30 Session M2c [Room CHASNA] (Chairman: Walid Ben-Ameur)

The cut condition for robust network design

Sara Mattia

Solving the Two-Stage Robust FTTH network design Problem under Demand Uncertainty

Cédric Hervet, Alain Faye, Matthieu Chardy, Marie-Christine Costa, Stanislas Francfort

Distributionally robust stochastic shortest path problem

Jianqiang Cheng, Abdel Lisser, Marc Letournel

Multipolar routing: where dynamic and static routing meet

Walid Ben-Ameur, Mateusz Żotkiewicz

Monday May 20th

13:30 – 15:30 Lunch

15:30 – 17:00 Session M3a [Room TEIDE] (Chairman: Alexandre Salles da Cunha)

Steiner tree network scheduling with opportunity cost of time

Fabien Cornillier

A branch-and-cut algorithm for the Multiple Steiner TSP with Order constraints
R. Taktak, S. Borne, A.R. Mahjoub

Approximating the capacitated facility location problem with length bounded trees
Jannik Matuschke, Andreas Bley, Benjamin Muller

The Degree Preserving Spanning Tree Problem: Valid Inequalities and Branch-and-cut method

Alexandre Salles da Cunha, Bernard Gendron, Abilio Lucena, Luidi Simonetti

15:30 – 17:00 Session M3b [Room GUAJARA] (Chairman: Elena Fernández)

The Two-Level Diameter Constrained Spanning Tree Problem

Markus Leitner, Luis Gouveia, Ivana Ljubić

Dynamic trees in exterior-point Simplex-type algorithms for network flow problems
Angelo Sifaleras, George Geranis

Stronger Lower Bounds for the Quadratic Minimum Spanning Tree Problem with Adjacency Costs

Dilson Lucas Pereira, Michel Gendreau, Alexandre Salles da Cunha

A Flow Formulation for the Optimum Communication Spanning Tree

Elena Fernández, Carlos Luna-Mota, Achim Hildenbrandt, Gerhard Reinelt, Stefan Wiesberg

15:30 – 17:00 Session M3c [Room CHASNA] (Chairman: Grit Claßen)

Multi-Band Robustness I: Model and Theory

Christina Büsing, Fabio D'Andreagiovanni

Multi-Band Robustness II: Constructing the Uncertainty Set

Fabio D'Andreagiovanni, Christina Büsing, Arie M.C.A. Koster, Manuel Kutschka

Multi-Band Robustness III: Application to Network Design Problems

Manuel Kutschka, Fabio D'Andreagiovanni, Arie M.C.A. Koster, Christina Büsing

Traffic Node Assignment in Wireless Networks: A Multi-Band Robust Knapsack Approach

Grit Claßen, Arie M.C.A. Koster, Manuel Kutschka, Anke Schmeink

Monday May 20th

15:30 – 17:00 Coffee Break

17:30 – 19:00 Session M4a [Room TEIDE] (Chairman: S. Raghavan)

Intelligent variable neighbourhood search for the minimum labelling spanning tree problem

S. Consoli, J. A. Moreno Pérez, N. Mladenović

Enhanced dynamic programming algorithms for the constrained subtree of a tree problem

Cristina Requejo, Agostinho Agra, Luidi Simonetti

Disjunctive combinatorial branch in a subgradient tree algorithm for the DCMST problem with VNS-Lagrangian bounds

Rafael Castro de Andrade, Adriano Tavares de Freitas

Local Search for the Reload Cost Spanning Tree Problem

S. Raghavan, Mustafa Sahin

17:30 – 19:00 Session M4b [Room GUAJARA] (Chairman: Andreas Bley)

2-InterConnected Facility Location: Specification, Complexity, and Exact Solutions

Maren Martens, Markus Chimani, Maria Kandyba

On the Two-Architecture Connected Facility Location Problem

Axel Werner, Markus Leitner, Ivana Ljubić, Markus Sinnl

An incremental algorithm for the uncapacitated facility location problem

Ashwin Arulselvan, Olaf Maurer, Martin Skutella

IP Modeling of the survivable hop constrained connected facility location problem

Andreas Bley, M. Rezapour, S. M. Hashemi

17:30 – 19:00 Session M4c [Room CHASNA] (Chairman: Edoardo Amaldi)

A three step decomposition approach for a transportation network design problem with non-linear costs

Lehuédé Fabien, Péton Olivier

An Extension of a Balanced Flow in a Network

Wataru Kishimoto

Load balancing optimization of telecommunication networks with two differentiated services

Amaro de Sousa, Carlos Lopes, Dorabella Santos

On single-path network routing subject to max-min fair flow allocation

Edoardo Amaldi, Stefano Coniglio, Luca G. Gianoli, Can Umut Ileri

Tuesday May 21th

9:00 – 9:50 Plenary Talk [Room TEIDE] (Chairman: Bernard Fortz)

Network design problems in phylogenetics
Martine Labbé

10:00 – 11:30 Session T1a [Room TEIDE] (Chairman: Leonardo Lamorgese)

Branch-and-Price for a European variant of the Railroad Blocking Problem
Robert Voll, Uwe Clausen

An integer fixed-charge multicommodity flow (FCMF) model for train unit scheduling
Zhiyuan Lin, Raymond S. K. Kwan

A fast heuristic approach for train timetabling in a railway node
Martin Philip Kidd, Fabio Furini

The track formulation for the Train Dispatching problem
Leonardo Lamorgese, Carlo Mannino

10:00 – 11:30 Session T1b [Room GUAJARA] (Chairman: Walid Ben-Ameur)

Edge Coloring by Total Labelings of 4-regular Circulant Graphs
Hamida Seba, Riadh Khennoufa

Intersecting a simple mixed integer set with a vertex packing set
Mahdi Doostmohammadi, Agostinho Agra, Cid Carvalho de Souza

Capacitated Network Design using Bin-Packing
Amal Benhamiche, A. Ridha Mahjoub, Nancy Perrot, Eduardo Uchoa

An interval Assignment Problem for Resource Optimization in LTE Networks
Faiz Hamid, Walid Ben-Ameur, Mohamad Assaad

10:00 – 11:30 Session T1c [Room CHASNA] (Chairman: Abilio Lucena)

Multi-period Reverse Logistics Network Design
Sibel A. Alumur, Stefan Nickel, Francisco Saldanha-da-Gama, Vedat Verter

A Network Approach to a Geometric Packing Problem
Bradley Paynter, Douglas R. Shier

Formulations for the Minimum 2-Connected Dominating Set Problem
Vinicius Leal do Forte, Abilio Lucena, Nelson Maculan

Formulating and Solving the Minimum Dominating Cycle Problem
Abilio Lucena, Alexandre Salles da Cunha, Luidi Simonetti

11:30 – 12:00 Coffee Break

Tuesday May 21th

12:00 – 13:30 Session T2a [Room TEIDE] (Chairman: Valentina Cacchiani)

Fleet sizing for offshore supply vessels
Yauhen Maisiuk, Irina Gribkovskaia

Integrated Airline Robust Scheduling and Fleet Assignment under Demand Uncertainty
Luis Cadarso, Ángel Marín

A matheuristic approach for solving a home health care problem
Hanane Allaoua, Sylvie Borne, Lucas Létocart and Roberto Wolfler Castro

A heuristic approach for an integrated fleet-assignment, aircraft-routing and crew-pairing problem
Valentina Cacchiani, Juan José Salazar González

12:00 – 13:30 Session T2b [Room GUAJARA] (Chairman: Hipólito Hernández-Pérez)

The One Commodity Pickup and Delivery Traveling Salesman Problem with Demand Intervals
Güneş Erdoğan, Gilbert Laporte, Roberto Wolfler Calvo

A Tabu Search Approach for the Prize Collecting Traveling Salesman Problem
Odivaney Pedro, Rodney Saldanha, Ricardo Camargo

Team Orienteering Problem with Decreasing Profits
Nacima Labadie, H. Murat Afsar

Heuristics procedures to solve the multi-commodity Pickup-and-Delivery Traveling Salesman Problem
Hipólito Hernández-Pérez, Juan José Salazar González

12:00 – 13:30 Session T2c [Room CHASNA] (Chairman: Bauguion Pierre-Olivier)

Modeling Earthquake Vulnerability of Highway Networks
İdil Arşik, F. Sibel Salman

k-Edge Failure Resilient Network Design
Richard Li-Yang Chen, Cynthia A. Phillips

Efficient algorithms for the 2-Way Multi Modal Shortest Path Problem
Marie-José Huguet, Dominik Kirchler, Pierre Parent, Roberto Wolfler Calvo

A new model for multicommodity flow problems, and a strongly polynomial algorithm for single-source Maximum Concurrent Flow
Bauguion Pierre-Olivier, Ben Ameur Valid, Gourdin Eric

13:30 – 15:30 Lunch

15:30 – 20:00 Excursion

Wednesday May 22th

9:00 – 9:50 Plenary Talk [Room TEIDE] (Chairman: Luis Gouveia)

Benders Decomposition for Hub Location
Jean-François Cordeau

10:00 – 11:30 Session W1a [Room TEIDE] (Chairman: Ibrahima Diarrassouba)

Lexicographical Minimization of Routing Hops in Telecommunication Networks
Pedro Patrício, Luís Gouveia, Amaro de Sousa

Design of Fixed Charge Multi-commodity Networks using k-Partition Facets
Yogesh K. Agarwal, Yash P. Aneja

A polyhedral study of the Hamiltonian p-median problem
Lena Hupp, Frauke Liers

Two Node-Disjoint 3-Hop-Constrained Survivable Network Design and Polyhedra
Ibrahima Diarrassouba, Hakan Kutucu, A. Ridha Mahjoub

10:00 – 11:30 Session W1b [Room GUAJARA] (Chairman: Giuliana Carello)

The final answer to the complexity of a basic problem in resilient network design
Artur Tomaszewski

Integer programming models for maximizing parallel transmissions in wireless networks
Yuan Li, Michał Pióro

On the use of multiple sinks to extend the lifetime in connected wireless sensor networks
F. Castaño, A. Rossi, M. Sevaux, N. Velasco

Energy-aware operations management in telecommunication network with shared protection mechanism
Giuliana Carello, Bernardetta Addis, Sara Mattia

10:00 – 11:30 Session W1c [Room CHASNA] (Chairman: Andreas Bärmann)

Generating conditional uniform random networks by optimization procedures
Stefano Nasini, Jordi Castro

P-Hub Maximal Covering Problem and Extensions for Gradual Decay Functions
Meltem Peker, Bahar Y. Kara

On Debris Removal During the Response Phase
Halenur Şahin, Oya Ekin Karaşan, Bahar Yetiş Kara

Solving Network Design Problems via Iterative Graph Aggregation
Andreas Bärmann, Frauke Liers, Alexander Martin, Maximilian Merkert, Christoph Thurner, Dieter Weninger

Wednesday May 22th

11:30 – 12:00 Coffee Break

12:00 – 13:30 Session W2a [Room TEIDE] (Chairman: Javier Faulin)

Solving the Single Vehicle Routing Problem with Variable Capacity
F.V. Louveaux, J.J. Salazar-Gonzalez

Minimizing waiting times in a route design problem with multiple use of a single vehicle

Iris Martínez-Salazar, Francisco Angel-Bello, Ada Alvarez

On the approximability of two capacitated vehicle routing problems

Adrian Bock, Laura Sanità

A Multi-Start Approach for Optimizing Routing Networks with Vehicle Loading Constraints

Javier Faulin, Angel A. Juan, Oscar Domínguez

12:00 – 13:30 Session W2b [Room GUAJARA] (Chairman: S.P. vanLoggerenberg)

Fiber Optical Network Design Using Passive Splitters
Başak Yazar, Bahar Yetiş Kara, Oya Ekin Karaşan

WDM Fiber Replacement Scheduling

Daniel Karch, Andreas Bley, Fabio D'Andreagiovanni

Iterated Local Search for Fiber Installation in Optical Network Optimization

Thiago F. Noronha, Daniel Morais dos Reis, Sérgio R. de Souza

Solving the Passive Optical Network with Fiber Duct Sharing Planning Problem Using Discrete Techniques

S.P. van Loggerenberg, M.J. Grobler, S.E. Terblanche

12:00 – 13:30 Session W2c [Room CHASNA] (Chairman: Luis Gouveia)

Multi-dimensional layered graphs models for routing problems

Mario Ruthmair, Luis Gouveia

On the Generalized Elementary Shortest Path Problem: A heuristic approach

Guerrero, W.J. Velasco, N. Prodhon, C. Amaya, C.A.

Benders Decomposition for a Location-Design Problem in Green Wireless Local Area Networks

Bernard Gendron, Rosario Garoppo, Gianfranco Nencioni, Maria Grazia Scutellà, Luca Tavanti

Polynomial-time separation of Enhanced Reverse Multistar Inequalities

Luis Gouveia, Juan José Salazar González

Wednesday May 22th

13:30 –15:30 Lunch

15:30 – 17:00 Session W3a [Room TEIDE] (Chairman: Jorge Riera-Ledesma)

Earthwork Optimization Models for the Construction of a Large Highway System
Manuel Iori, Stefano Novellani, Barbara Panicucci, Christian Bogenberger, Mauro Dell'Amico, Gerhard Hoefinger

Solving the bi-objective prize-collecting Steiner tree problem with the epsilon-constraint method

Markus Sinnl, Markus Leitner, Ivana Ljubić

On the Biobjective Adjacent Only Quadratic Spanning Tree Problem
Sílvia M. D. M. Maia, Elizabeth F. G. Goldbarg, Marco C. Goldbarg

The Double Vehicle Routing Problem with Multiple Stacks: Exact Approaches
Jorge Riera-Ledesma, María Batista-Galván, Manuel Iori

15:30 – 17:00 Session W3b [Room GUAJARA] (Chairman: Federico Perea)

Mathematical Analysis of caching policies and cooperation in YouTube-like services
Pablo Romero, Franco Robledo, Pablo Rodríguez-Bocca, Claudia Rostagnol

Optimizing Toll Enforcement in Transportation Networks: a Game-Theoretic Approach

Guillaume Sagnol, Ralf Borndörfer, Julia Buwaya, Elmar Swarat

Fitness Landscape Analysis for Scalable Multicast RRM Problem in Cellular Network
Frédéric Lassabe, Qing Xu, Hakim Mabed, Alexandre Caminada

Introducing new stations on a high-speed railway corridor competing with another transportation mode

Federico Perea, Juan A. Mesa, Gilbert Laporte

Wednesday May 22th

17:00 – 17:30 Coffee Break
17:30 – 19:00 Session W4a [Room TEIDE] (Chairman: I. Diarrassouba)
The Least Cost Influence Problem Dilek Gun nec, S. Raghavan, Rui Zhang
On the Hardness of Equal Shortest Path Routing Issam Tahiri, Frédéric Giroire, Stéphane Pérennes
Exact solution for branch vertices constrained spanning problems Merabet Massinissa, Sylvain Durand, Miklos Molnar
Integer Programming Formulations for the k-Edge-Connected 3-Hop-Constrained Network Design Problem I. Diarrassouba, V. Gabrel, L. Gouveia, A. R. Mahjoub, P. Pesneau
17:30 – 19:00 Session W4b [Room GUAJARA] (Chairman: Monia Giandomenico)
Lagrangian Decompositions of the Two-Level FTTx Network Design Problem Olaf Maurer, Andreas Bley, Ivana Ljubić
Effective MIP formulation for optimal 1-for-N diversity coding Mateusz Żotkiewicz
A New Lower Bound for the Kirchhoff Index using a numerical procedure based on Majorization Techniques Alessandra Cornaro, Gian Paolo Clemente
Approximating the Lovász theta Function with the Subgradient Method Monia Giandomenico, Adam N. Letchford, Fabrizio Rossi, Stefano Smriglio
20:00 – 22:00 Social Dinner

Martin Skutella, Technische Universität Berlin (Germany)

“Flows Over Time: Modeling Network Routing Problems Including a Temporal Dimension”

In many real-world routing problems time plays a vital role. Network flows over time include a temporal dimension and therefore provide a more realistic modeling tool than classical “static” network flow models. While flows over time have already been introduced by Ford and Fulkerson in the 1950s, several more sophisticated models have been studied later on. After giving a general introduction to network flows over time, we are going to present recent work in this area.

Martine Labbé, Université Libre de Bruxelles (Belgium)

”Network design problems in phylogenetics”

A phylogeny is an unrooted (generally binary) tree that represents the evolutionary relationships of a set of n species. Phylogenies find applications in several scientific areas ranging from medical research, to drug discovery, to epidemiology, to systematics, and to population dynamics. In such applications, the available information is usually restricted to the leaves of a phylogeny and is represented by molecular data extracted from the analyzed species, such as DNA, RNA, amino acid, or codon fragments. The phylogeny can be determined by solving a network design problem, called the phylogeny estimation problem (PEP) which consists in determining a tree whose leaves are the species under study. Several versions of the problem exist depending on the criterion used to select a phylogeny from among plausible alternatives. This talk presents an overview of several such network design problems as well as recent results concerning them.

Jean-François Cordeau, HEC Montréal (Canada)

“Benders Decomposition for Hub Location”

Hub location problems are hard combinatorial optimization problems arising in the design of transportation and telecommunication networks that rely on the popular hub-and-spoke architecture. Many of these problems combine binary decisions for the selection of hub nodes with continuous decisions representing flows of commodities in the network. For this reason, they are often good candidates for primal decomposition approaches, such as Benders decomposition, which can take advantage of the separability of the problem. This talk will cover recent progress made in the exact solution of large-scale hub location problems by Benders decomposition. The talk will focus on the classical uncapacitated problem with multiple assignments, but extensions to problems with multiple capacity levels or with stochastic transportation costs will also be discussed.

Generating conditional uniform random networks by optimization procedures

STEFANO NASINI

JORDI CASTRO

stefano.nasini@upc.edu

jordi.castro@upc.edu

Dept. of Statistics and Operations Research
Universitat Politècnica de Catalunya
Jordi Girona 1–3
08034 Barcelona

Abstract

Complex networks is a recent area of research motivated by the empirical study of real-world networks, such as social relations, protein interaction, neuronal connections, etc. As closed-form probabilistic models of networks are often not available, the ability of randomly generating networks verifying specific constraints might be useful. The purpose of this work is to develop optimization-based procedures to randomly generate networks with structural constraints, within the probabilistic framework of conditional uniform models. Based on the characterization of families of networks by means of systems of linear constraints, polynomial-time methods to generate networks with specified structural properties are constructed.

Key words: Conditional Random Networks, Linear Programming, Interior Point Methods, Block-angular problems, Social Networks.

1 Extended Abstract

After observing a real-world network, a conditional uniform model [1] might be used to check whether empirical features (for example, the number of closed triangles, or the number of connected paths) are significantly unlikely under a uniform distribution of all networks having fixed and well known structural properties (for example, uniform probability distribution of all networks with fixed density). Comparing the observed network with an ensemble of such networks allows one to detect deviations from randomness in network properties.

Let x_{ij} be the (i, j) component of the adjacency matrix (AM, from now on) of a simple graph. Then, the set G_n of all simple graphs with n nodes is characterized by the integer points of the polytope $\varphi = \{(x_{12}, \dots, x_{n-1n}) \in [0, 1]^{n(n-1)} : x_{ij} - x_{ji} = 0, i = 1 \dots n-1, i < j \leq n\}$, i.e., $\varphi \cap \mathbb{Z}^{n(n-1)}$. Adding nonnegative slacks, the representation of φ in standard matrix form is

$$\left[\begin{array}{c|c|c|c} I & -I & & \\ \hline I & & I & \\ \hline & I & & I \end{array} \right] \begin{bmatrix} \mathbf{x} \\ \mathbf{s} \end{bmatrix} = \begin{bmatrix} 0 \\ \mathbf{e} \\ \mathbf{e} \end{bmatrix} \quad \begin{bmatrix} \mathbf{x} \\ \mathbf{s} \end{bmatrix} \geq \mathbf{0}. \quad (1)$$

Matrix I is the $n(n-1)/2 \times n(n-1)/2$ identity matrix, $\mathbf{e} = (1, \dots, 1)^T \in \mathbb{R}^{n(n-1)/2}$, $\mathbf{x}^T = [x_{12}, x_{13}, \dots, x_{1n}, x_{23}, x_{24}, \dots, x_{2n}, \dots, x_{45}, \dots, x_{(n-1)n}, x_{21}, x_{31}, x_{32}, x_{41}, \dots, x_{n(n-1)}]$ and $\mathbf{s}^T = [s_{12}, s_{13}, \dots, s_{1n}, s_{23}, s_{24}, \dots, s_{2n}, \dots, s_{45}, \dots, s_{(n-1)n}, s_{21}, s_{31}, s_{32}, s_{41}, \dots, s_{n(n-1)}]$.

The coefficient matrix of (1) is totally unimodular (TU, from now on), such that all extreme points of φ are integer. Moreover, since φ is a subset of the $n(n - 1)$ -dimensional unit cube, there couldn't be any integer point in the interior, so that each graph with n nodes, whose AM is given by an integer solution of (1), must be associated to a unique extreme point of φ and, conversely, the latter is by construction associated to a unique AM. Sufficient conditions for a matrix to be totally unimodular are shown in the appendix of [4].

We proved that, due to the total unimodularity of their matrix structure, many interesting families of networks could be characterized by extreme points of polytopes:

- undirected networks conditioned to the total density;
- directed networks conditioned to the total density;
- undirected networks conditioned to the within & between group densities;
- undirected networks conditioned to the within group densities;
- undirected networks conditioned to the between group densities;
- directed networks conditioned to the within & between group densities;
- directed networks conditioned to the within group densities;
- directed networks conditioned to the between group densities;
- undirected networks conditioned to the within group densities and the total density;
- undirected networks conditioned to between group densities and the total density;
- directed networks conditioned to the within group densities and the total density;
- directed networks conditioned to the between group densities and the total density;
- directed networks conditioned to the in-&-out-degree sequences;
- undirected networks conditioned to the lower bound of the degree sequence range and density;
- directed networks conditioned to the diad count;
- directed networks conditioned to the number of mutual links;

The fact that we are able to associate to each network verifying specified constraints an extreme point of a polytope, allows to generate random networks by solving linear programs.

It may be seen that the systems of linear constraints associated to the families of networks listed above have a primal block-angular structure, as the following

$$\begin{aligned} \min \quad & \sum_{i=0}^k c^T x^i && (2a) \\ \text{subject to} \quad & \begin{bmatrix} N_1 & & & & \\ & N_2 & & & \\ & & \ddots & & \\ & & & N_k & \\ L_1 & L_2 & \dots & L_k & I \end{bmatrix} \begin{bmatrix} x^1 \\ x^2 \\ \vdots \\ x^k \\ x^0 \end{bmatrix} = \begin{bmatrix} b^1 \\ b^2 \\ \vdots \\ b^k \\ b^0 \end{bmatrix} && (2b) \\ & \mathbf{0} \leq x^i \leq \mathbf{1} \quad i = 0, \dots, k. && (2c) \end{aligned}$$

Matrices $N_i \in \mathbb{R}^{m_i \times n_i}$ and $L_i \in \mathbb{R}^{l \times n_i}$, $i = 1, \dots, k$, respectively define the block-diagonal and linking constraints, k being the number of blocks. Vectors $x^i \in \mathbb{R}^{n_i}$, $i = 1, \dots, k$, are the variables for each block. $x^0 \in \mathbb{R}^l$ are the slacks of the linking constraints. $b^i \in \mathbb{R}^{m_i}$, $i = 1, \dots, k$, is the right-hand-side vector for each block of constraints, whereas $b^0 \in \mathbb{R}^l$ is for the linking constraints. The upper bounds for each group of variables are defined by u^i , $i = 0, \dots, k$.

Specialized versions of the simplex method, interior point methods and decomposition techniques have been proposed for these particular structures [2, 3].

Let $A \in \mathbb{R}^{m' \times n'}$ be the coefficient matrix associated to one of the polytopes whose extreme points are bijectively related to the aforementioned families of networks. By the Fundamental Theorem of Linear Programming, given a $\mathbf{c} \in \mathbb{R}^{|V|(|V|-1)}$, there exists a network g belonging to the specified family of networks such that $g \in \operatorname{argmax}_{\mathbf{x}} \mathbf{c}^T \mathbf{x}$, s.t. $\mathbf{x} \in \varphi$. This means that, if \mathbf{c} is a properly defined random vector, the probability distribution of a network in the specified family can be obtained by associating to a given $\mathbf{c} \in \mathbb{R}^{|V|(|V|-1)}$ the set $\{\mathbf{x} \in \varphi : \mathbf{x} \in \operatorname{argmax}_{\mathbf{x}} \mathbf{c}^T \mathbf{x}\}$.

Clearly, if each network is an optimal solution of a LP with randomly generated objective function, we cannot state that every network has the same probability of being selected, i.e. we are not authorized to regard the sample of networks generated this way as draw from a uniform distribution. After fixing a random objective function the probability of obtaining a network depends on geometrical properties of the associated polytope (the polytope whose extreme points characterize the family). It seems in any case reasonable to assume that such probability does not depend on any structural property of the network, except for the ones defined by the specified constraints.

If we have a given extreme point \mathbf{x}_0 of φ , we might obtain another extreme point \mathbf{x}_1 of φ by fixing r variables and optimizing, with a given objective cost vector $\mathbf{c} \in \mathbb{R}^{N-r}$, the remaining $N - r$ variables.

To clarify this method, consider a primal block-angular problem with R blocks and let $\mathbf{r} \subset \{1, 2, \dots, R\}$. Let $\mathbf{x}_{(\mathbf{r})}$ be the vector of decision variables associated with blocks indexed by \mathbf{r} and $\mathbf{x}_{(\mathbf{r}^c)}$ be the vector of decision variables associated with the $|\mathbf{r}^c| = R - |\mathbf{r}|$ remaining blocks. Similarly, let $A_{(\mathbf{r})}$ represent the submatrix of A , whose columns corresponds to the blocks indexed by \mathbf{r} and $A_{(\mathbf{r}^c)}$ the submatrix of A , whose columns corresponds to the blocks indexed by \mathbf{r}^c .

Algorithm 1 illustrates the iterative procedure to generate a sequence of networks by randomly selecting $\mathbf{c} \in \mathbb{R}^{N-r}$ and the blocks to be optimized \mathbf{r} .

Algorithm 1 R-blocks chain

- 1: Let $k = 0$ \mathbf{x}^k be an initial extreme point
 - 2: **repeat**
 - 3: Randomly select $\mathbf{c} \in \mathbb{R}^{N-r}$ and $\mathbf{r} \subset \{1, 2, \dots, R\}$;
 - 4: Construct $A_{(\mathbf{r}^c)}$, $A_{(\mathbf{r})}$, $\mathbf{x}_{(\mathbf{r}^c)}$ and $\mathbf{x}_{(\mathbf{r})}$;
 - 5: Solve the associated subproblem;
 - 6: Update $\mathbf{x}_{(\mathbf{r})}^{k+1}$ using the optimal solution;
 - 7: $k = k + 1$;
 - 8: **until** $k \geq K$
-

Another way to generate extreme points of a polytope is to produce a sequence of nondegenerate bases. Consider a partition of the set of columns of A into two groups: the basic columns \mathcal{B} (linearly independent, $|\mathcal{B}| = \operatorname{rank}(A)$), and the nonbasic ones \mathcal{N} . The components of \mathbf{x} associated to the non-basic columns ($\mathbf{x}_{\mathcal{N}}$) are fixed at their limits, namely either 0 or 1, and the variables associated to basic columns ($\mathbf{x}_{\mathcal{B}}$) are computed as $\mathbf{x}_{\mathcal{B}} = B^{-1}(\mathbf{b} - N\mathbf{x}_{\mathcal{N}})$, where B and N are the matrices of columns of A associated to \mathcal{B} and \mathcal{N} . Denoting as e_q the q -th column of the identity matrix of dimension $|\mathcal{N}|$, $q = 1, \dots, |\mathcal{N}|$, if $|\varphi \cap \mathbb{Z}^{n(n-1)}| > 1$ (i.e, there is more than one extreme point in φ), from an extreme point $x_k \in \varphi$ we can compute another extreme point as

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \lambda \begin{bmatrix} B^{-1}Ne_q \\ -e_q \end{bmatrix}. \quad (3)$$

Indeed (3) is equal to the standard simplex iteration. Using (3) we can obtain a finite sequence

of K extreme points $\{\mathbf{x}_k\}_{k=0}^K$. (From now on we shall use $\delta_{\mathcal{B}}(q)$ to denote in a more compact notation the direction of movement of (3).)

It can be shown for the polytopes described in this paper (subsets of a finite-dimensional hypercube with integer extreme points) that the step length $\lambda \in \{-1, 0, 1\}$.

The sequence (3) requires the repeated selection of $q \in \{1, \dots, |\mathcal{N}|\}$ and $\lambda \in \{-1, 0, 1\}$. Thus, if we let q be a random variable, (3) gives rise to a Markov Chain over the extreme points of φ . Algorithm 2 provides an iterative procedure to simulate using that Markov chain.

Algorithm 2 Pivoting sequence

```

1: Let  $\mathbf{x}_0$  be an initial extreme point
2: repeat
3:   Randomly select  $q \in \{1, \dots, |\mathcal{N}|\}$ ;
4:   compute the feasible step-length  $\lambda \in \{-1, 0, 1\}$  associated with  $q$ ;
5:   if  $\lambda \neq 0$  then
6:     Update  $\mathcal{B}^{-1}$  and  $\mathcal{N}$  and compute  $\mathbf{x}_{k+1} = \mathbf{x}_k + \lambda\delta_{\mathcal{B}}(q)$ ;  $k = k + 1$ ;
7:   end if
8: until  $k \geq K$ 
```

The two procedures to generate extreme points have been implemented, for most of the aforementioned families of networks. The comparison of our computational results with the few available analytical solutions confirms that both procedures provide coherent numerical results, matching the known theoretical values. When analytical results are not available, the two procedures give rise to the same results for large sample sizes and differ only in terms of efficiency, since many bases obtained by the pivoting method do not produce new networks due to the high degeneracy of the problems. Finally, the method based on the sequential random selection of K blocks is (1) efficient, and (2) allows to obtain unbiased information of real-world network properties.

References

- [1] Bollobas, B., (1985), *Random Graphs*. Cambridge University Press.
- [2] Castro, J., (2000), A specialized interior-point algorithm for multicommodity network flows, *SIAM J. Optim.*, 10, 852-877.
- [3] Castro, J., (2003), Solving difficult multicommodity problems through a specialized interior-point algorithm, *Ann. Oper. Res.*, 124, 35-48.
- [4] Heller, I., Tompkins, C.B., (1956), 'An Extension of a Theorem of Dantzig's', in K. Kuhn, A. Tucker, *Linear Inequalities and Related Systems*, Princeton University Press, Princeton, 247-254.

Polynomial-time separation of Enhanced Reverse Multistar Inequalities

Luis Gouveia (University of Lisbon, Portugal)

Juan José Salazar González (University of La Laguna, Spain)

The Vehicle Routing Problem with a minimum number of customers per route (LVRP) concerns a vehicle routing problem with one depot, all the customers with a unit demand, a homogeneous fleet of vehicles, and with upper and lower bounds on the number of customers visited by each vehicle. The LVRP differs from the “usual” Capacitated Vehicle Routing Problem (CVRP) in the sense that LVRP also includes an additional constraint on the minimum number of customers per route. Gouveia, Riera and Salazar (2012) introduced a straightforward flow-based model for this problem as well as several cut-like inequalities defined in the space of the design variables, including the Enhanced Reverse Multistar (ERMS) inequalities. Since these inequalities are not implied by the linear programming relaxation of the flow-based model, the natural question is to know whether there exists a compact formulation for the problem such that the corresponding LP relaxation implies the ERMS inequalities, as well as finding a polynomial-time separation algorithm for this class of inequalities. The two questions were open in the previously cited article.

In this work we answer the two questions. In particular we present a new two-flow based model for the LVRP which implies the ERMS inequalities, and also we describe a polynomial-time algorithm that finds a violated ERMS inequality by a given fractional solution (when such inequality exists).

Multi-dimensional layered graphs models for routing problems

Luis Gouveia¹ and Mario Ruthmair²

¹ Centro IO & Faculdade de Ciencias,
University of Lisbon, Lisbon, Portugal
legouveia@fc.ul.pt

² Institute of Computer Graphics and Algorithms,
Vienna University of Technology, Vienna, Austria
ruthmair@ads.tuwien.ac.at

1 Abstract

Recently, the literature has shown that layered graph approaches have been successful for solving several network design problems involving hop or general resource constraints [7–9, 11, 16] as well as some routing problems [1, 4–6, 12].

In particular, the most successful layered graph models in terms of computation times are based on a modified Picard-Queyranne [13] model and a straightforward cutting plane approach that separates connectivity cuts on an adequate layered graph. In general, the size of the layered graphs is greater than the size of the original graph or substantially greater if problems with non-unit resource demands are being solved; however, recently Ruthmair and Raidl [14, 15] proposed an approach to handle these difficulties: It approximates the linear programming (LP) relaxation bound of layered graph models by solving lower and upper bounds in significantly reduced graphs, iteratively strengthening the bounds by appropriately extending these graphs.

In this talk we consider variants of routing problems where the layered graph is extended from the typical layered graph with one resource dimension (1D) to a layered graph with two resource dimensions (2D). The reason for this additional dimension is due to the consideration of two different commodities or resources. We compare models based on the 2D graph with models based on two 1D layered graphs, each one modeling one of the resources, with adequate linking constraints.

The first problem we consider is the well-known vehicle routing problem with time windows [3]. Here, on the one hand we have to ensure that the vehicle capacity is not exceeded and on the other hand the customer time windows have to be respected. Thus, either we model the two resource dimensions – the vehicle load and the travel time – in two separated 1D layered graphs and link them via the original graph, or we model both resources in one combined 2D graph. In this problem the two types of resources are not related to each other.

Next, we consider the family of one-to-many-to-one pickup and delivery problems [2]. Here, one commodity has to be delivered from the depot to the customers while another commodity has to be picked up from the customers and returned to the depot. The vehicle load with respect to each of these two commodities can be modeled in separate resource dimensions. However, there is a relation between the two commodities, i.e. we have to ensure that the sum of the two loads does not exceed the vehicle capacity. While it is not obvious how to model this combined-load constraint in two separate 1D layered graphs it is quite easy to consider it in the 2D graph: We simply do not include any arc leading to an infeasible load combination. Also other possibly more complicated constraints relating different commodities are imaginable which may be difficult to model in the natural space or in the extended space defined by two separated 1D graphs but probably in an easy way in 2D graphs.

Additionally, we also consider the single-commodity pickup and delivery traveling salesman problem (TSP) [10]. In this variant there is only one commodity and the single vehicle has to pickup up given numbers of items from supply nodes and deliver them to demand nodes. Since the vehicle load can both increase and decrease on a route a corresponding layered graph modeling this load may in general be not acyclic anymore. Thus, solution connectivity can not be ensured by a (pseudo-)polynomial number of flow conservation constraints. If we model the TSP part separately e.g. with the strong compact formulation in [6], connectivity in the 1D load graph is automatically forced by appropriate linking constraints. On the other hand if we use a combined 2D graph with sequence and load dimension we obtain an acyclic graph again.

The aim of this work is to gain insight in the modeling on multi-dimensional layered graphs with respect to the theoretically achievable LP relaxation bounds as well as to computational aspects and practical applicability.

References

1. Abeledo, H., Fukasawa, R., Pessoa, A., Uchoa, E.: The Time Dependent Traveling Salesman Problem: Polyhedra and Branch-Cut-and-Price. In: *Experimental Algorithms*. vol. 6049, pp. 202–213 (2010)
2. Berbeglia, G., Cordeau, J.F., Gribkovskaia, I., Laporte, G.: Static pickup and delivery problems: a classification scheme and survey. *Top* 15(1), 1–31 (2007)
3. Desaulniers, G., Desrosiers, J., Spoerrendonk, S.: The Vehicle Routing Problem with Time Windows: State-of-the-Art Exact Solution Methods. In: *Wiley Encyclopedia of Operations Research and Management Science*, pp. 1–11. Wiley (2010)
4. Godinho, M.T., Gouveia, L., Magnanti, T.L.: Combined route capacity and route length models for unit demand vehicle routing problems. *Discrete Optimization* 5(2), 350–372 (2008)
5. Godinho, M.T., Gouveia, L., Pesneau, P.: Natural and extended formulations for the Time-Dependent Traveling Salesman Problem. *Discrete Applied Mathematics* (2011)
6. Godinho, M.T., Gouveia, L., Pesneau, P.: On a Time-Dependent Formulation and an Updated Classification of ATSP Formulations. *Progress in Combinatorial Optimization* (2011)
7. Gouveia, L., Leitner, M., Ljubić, I.: On the Hop Constrained Steiner Tree Problem with multiple Root Nodes. In: Mahjoub, A.R., Others (eds.) *Proceedings of the 2nd International Symposium on Combinatorial Optimization*. LNCS, vol. 7422, pp. 201–212. Springer (2012)
8. Gouveia, L., Leitner, M., Ljubić, I.: The Two-Level Diameter Constrained Spanning Tree Problem. Tech. Rep. TR 186–1–12–02, Institute of Computer Graphics and Algorithms, Vienna University of Technology (2012)
9. Gouveia, L., Simonetti, L.G., Uchoa, E.: Modeling hop-constrained and diameter-constrained minimum spanning tree problems as Steiner tree problems over layered graphs. *Mathematical Programming* 128(1), 123–148 (2011)
10. Hernández-pérez, H., Salazar-González, J.J.: The One-Commodity Pickup-and-Delivery Traveling Salesman Problem: Inequalities and Algorithms. *Networks* 50(4), 258–272 (2007)
11. Ljubić, I., Gollowitzer, S.: Layered Graph Approaches to the Hop Constrained Connected Facility Location Problem. *INFORMS Journal on Computing* (2012)
12. Pessoa, A., de Aragão, M.P., Uchoa, E.: Robust Branch-Cut-and-Price Algorithms for Vehicle Routing Problems. In: Golden, B., Raghavan, S., Wasil, E., Sharda, R., Voß, S. (eds.) *The Vehicle Routing Problem: Latest Advances and New Challenges*, Operations Research/Computer Science Interfaces Series, vol. 43, pp. 297–325. Springer (2008)
13. Picard, J.C., Queyranne, M.: The time-dependent traveling salesman problem and its application to the tardiness problem in one-machine scheduling. *Operations Research* 26(1), 86–110 (1978)
14. Ruthmair, M.: On Solving Constrained Tree Problems and an Adaptive Layers Framework. Ph.D. thesis, Vienna University of Technology, Institute of Computer Graphics and Algorithms, Vienna, Austria (May 2012), supervised by G. R. Raidl and U. Pferschy
15. Ruthmair, M., Raidl, G.R.: A Layered Graph Model and an Adaptive Layers Framework to Solve Delay-Constrained Minimum Tree Problems. In: Günlük, O., Woeginger, G. (eds.) *Proceedings of the 15th Conference on Integer Programming and Combinatorial Optimization (IPCO XV)*. LNCS, vol. 6655, pp. 376–388. Springer (2011)
16. Ruthmair, M., Raidl, G.R.: On Solving the Rooted Delay- and Delay-Variation-Constrained Steiner Tree Problem. In: Mahjoub, A., et al. (eds.) *Proceedings of the 2nd International Symposium on Combinatorial Optimization*. LNCS, vol. 7422, pp. 225–236. Springer (2012)

Robust combinatorial optimization with uncertain cost

Michael Poss

UMR CNRS 7253 Heudiasyc, Université de Technologie de Compiègne,
Centre de Recherches de Royallieu, 60200 Compiègne, France.
Email : mjposs@gmail.com

We present in this work a new model for robust combinatorial optimization with cost uncertainty that generalizes the classical budgeted uncertainty set [BS04]. We suppose here that the budget of uncertainty is given by a function of the problem variables, yielding an uncertainty multifunction [Po13]. The new model is less conservative than the classical model and approximates better Value-at-Risk objective functions, especially for vectors with few non-zero components. Examples of budget functions are constructed from the probabilistic bounds computed by Bertsimas and Sim in [BS04]. We provide an asymptotically tight bound for the cost reduction obtained with the new model. We turn then to the tractability of the resulting optimization problems and provide a mixed-integer linear reformulation. We show that when the budget function is affine, the resulting optimization problems can be solved by solving $n+1$ deterministic problems, extending to our model the result proven in [BS03]. We propose combinatorial algorithms to handle problems with more general budget functions. We also adapt existing dynamic programming algorithms to solve faster the robust counterparts of optimization problems, which can be applied both to the traditional budgeted uncertainty model and to our new model. We evaluate numerically the reduction in the price of robustness obtained with the new model on the shortest path problem and on a survivable network design problem.

References:

- [BS03] D. Bertsimas and M. Sim: Robust discrete optimization and network flows. *Math. Program.* 98(1-3): 49-71 (2003).
- [BS04] D. Bertsimas and M. Sim: The price of robustness. *Operations Research*, 52:35–53 (2004).
- [Po13] M. Poss: Robust combinatorial optimization with variable budgeted uncertainty, *4OR*, in press, doi: 10.1007/s10288-012-0217-9.

Lexicographical Minimization of Routing Hops in Telecommunication Networks

Luís Gouveia

CIO and DEIO, Faculdade de Ciências da Universidade de Lisboa, legouveia@fc.ul.pt

Pedro Patrício

CIO and Departamento de Matemática, Universidade da Beira Interior, pedrofp@ubi.pt

Amaro de Sousa

Instituto de Telecomunicações, Universidade de Aveiro, asou@ua.pt

Given a telecommunications network with known link capacities and a set K of commodities with known demands, each of which is supported by D hop-constrained node-disjoint service and backup paths such that total demand protection is ensured in case of both single and double failure scenarios, in this work we address the following traffic engineering problem: how to route small deterministic modifications of the original demands, maintaining network survivability and QoS, while minimizing the number of hops of either i) all service paths or ii) the worst service path for each commodity, in a lexicographical sense.

In previous papers, two hop related objective functions were considered: the minimization of the average number of hops and the minimization of the worst case number of hops. The first objective gives the best overall average performance but it may be unfair for some of the commodities. One way of optimizing the fairness criterion is to consider the minimization of the worst case number of hops. Nevertheless, this objective does not account for the minimization of the number of hops of the routing paths that can have fewer hops than its worst value.

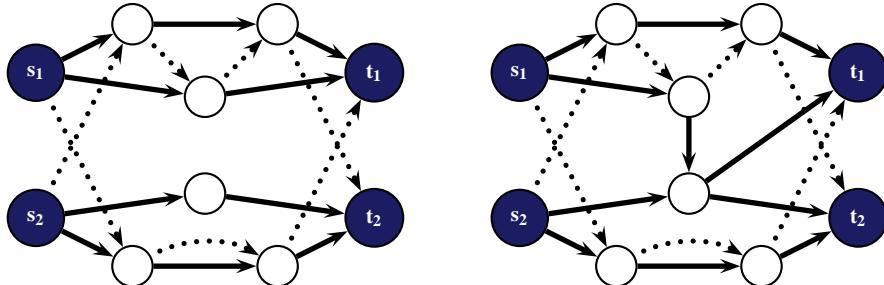
In another previous paper, we considered the case with $D = 1$ and generalized the “worst case” objective function by minimizing the number of routing hops in a lexicographical sense, *i.e.*, minimizing the number of paths with H hops; then, among all such solutions, minimizing the number of paths with $H - 1$ hops; and so on. We presented two approaches, an iterative approach and a single model approach, using two classes of Integer Linear Programming models. We concluded that the single hop-indexed model was more efficient and showed that it was also preferable in theoretical terms.

In this work, we address the cases with $D = 2, 3, 4$ and further generalize the aforementioned “worst case” objective function. It is important to note that we are interested in ensuring a degree of survivability such that total demand protection (Path Protection) is guaranteed under single ($n = 1$) or double ($n = 2$) failure scenarios. Denoting by Δ the number of service paths for each commodity, we thus have $\Delta = D - n$ (and $D - \Delta$ is the number of backup paths). Our aim is to lexicographically minimize the number of routing hops of either i) all service paths of each commodity (variant A) or ii) the worst service path of each commodity (variant B). In other words, we wish to minimize the number of either all or the worst service paths with H hops (where H represents the

maximum number of allowable hops); then, among all such solutions, minimizing the number of either all or the worst service paths with $H - 1$ hops; and so on.

Regarding the lexicographical minimization of all service paths, consider the vector of routing hops $(h_m)_{m \in \{1, \dots, \Delta|K|\}}$ where h_m is the number of hops of a service path d , with $d = 1, \dots, \Delta$, of some commodity $k \in K$. Let $[h_m]_{m \in \{1, \dots, \Delta|K|\}}$ be the vector obtained by rearranging (h_m) in a non-increasing order. Given two vectors $[a_m]$ and $[b_m]$, the first vector is said to be lexicographically smaller than the latter if either $a_1 < b_1$ or there exists an index $l \in \{1, \dots, \Delta|K| - 1\}$ such that $a_i = b_i$ for all $i \leq l$ and $a_{l+1} < b_{l+1}$.

Figure 1 illustrates two feasible solutions considering an instance with $|K| = 2$, $D = 3$, $\Delta = 2$ and $H = 3$, where the service paths (those with fewer hops) of each commodity are represented by thick lines, whereas dotted lines represent backup paths. Clearly, the solution represented on the left is the best one in lexicographic terms, since it has a vector $[a_m] = [3 \ 3 \ 2 \ 2]$, while the solution represented on the right has a vector $[b_m] = [3 \ 3 \ \overline{3} \ \overline{2}]$. Overlined numbers represent the number of hops of the service paths of commodity from s_2 to t_2 .



As for the lexicographical minimization of the worst service paths, consider the vector of routing hops is $(h_k)_{k \in K}$, where h_k is the number of hops of the worst service path of commodity $k \in K$, and also $[h_k]_{k \in K}$, by rearranging (h_k) in a non-increasing order. In the example pictured in Figure 1, we have $[a_k] = [b_k] = [3 \ \overline{3}]$.

In this work, we present and discuss three classes – disaggregated, mixed and aggregated – of Integer Programming hop-indexed models for both variants A and B of this problem, comparing them in theoretical terms, and also present computational results considering $D = 2, 3, 4$ and several traffic demand settings.

Steiner tree network scheduling with opportunity cost of time

Fabien Cornillier

Universidad del Pacífico, Graduate School of Business, Lima, Peru

Interuniversity Research Centre on Enterprise Networks, Logistics and Transportation (CIRRELT), Canada

Keywords: Steiner tree network design, Project scheduling, Resource-constrained scheduling

Whenever significant opportunity cost of time exists (discount rate, rate of time preference, time-dependent profits, etc.) a time-cost tradeoff maybe considered in designing real-world structure like electricity, gas, water or telecommunication networks. For instance, in public procurement, contracts are most often awarded to the lowest qualified bidder complying with a predetermined delivery date. Because social welfare often depends on completion times, a competitive advantage is generally given to the contractor who proposes an aggressive schedule. Furthermore, an increasing amount of contracts nowadays includes explicit time incentives through scoring auction systems such as A+B bidding where offers are scored using both the cost (A) and the completion time (B). The need for a cost-time tradeoff may arise in the telecommunications industry. Telephone and cable telecommunications companies are competing in a rapidly evolving business and technological conditions. Moreover, when no regulated infrastructure monopoly exists, national regulatory authorities tend to foster infrastructure-based competition which seems to lead to a higher level of nationwide coverage and social welfare. In such context, telecommunications companies continuously need to optimize their network infrastructure to deploy the latest technologies on short time-to-market schedules. However, expanding an existing network in a race for subscribers involves high investment costs while time tends to erode the expected profits of connecting new geographical areas. The time dimension becomes a key issue for an effective network modeling. In such projects the network design decisions and the activity scheduling decisions are typically made sequentially: project managers and planners determine the minimal length or cost Steiner tree network, then schedule each selected activities so as to minimize the completion time. When opportunity cost of time exists, we demonstrate that taking a more holistic perspective by intertwining both decisions can result in a more effective network design.

The paper presents a new problem in which a Steiner tree network has to be designed and the selected activities conjointly scheduled while considering a limited amount of renewable resources, so as to optimize an objective function, i.e., a distance, a completion time, a cost, or a present value. We propose a formal definition and a mixed-integer programming formulation of the Steiner Tree Scheduling Problem (STSP), and we describe a heuristic procedure to solve it.

The Steiner Tree Scheduling Problem. The Steiner Tree Scheduling Problem is formulated as follows. Let $G = \{V, A\}$ be a digraph with a vertex set $V = \{1, \dots, n_v\}$, and an arc set $A =$

[☆]Extended abstract

Email address: fyp.cornillier@up.edu.pe (Fabien Cornillier)

Preprint submitted to International Network Optimization Conference, 2013

February 28, 2013

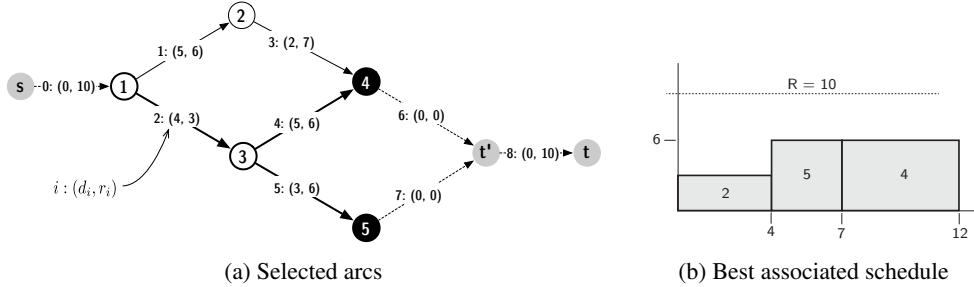


Figure 1: Optimal solution of the Steiner Tree Problem

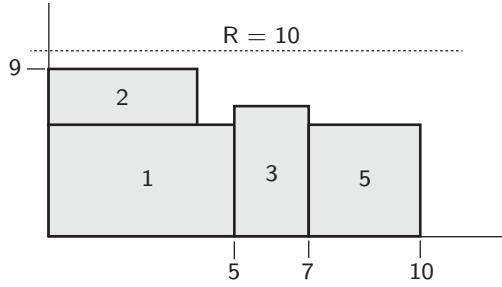


Figure 2: Optimal solution of the Steiner Tree Scheduling Problem

$\{1, \dots, n_a\}$ such that $A \subseteq V^2$. The vertex set V is partitioned into three subsets: the root vertex $\{1\}$, the set V_S of Steiner vertices, and the set V_T of terminal vertices. Let K be a set of renewable resources where each resource $k \in K$ has a total capacity R_k for any time interval. Each arc i corresponds to an activity associated with a duration d_i , and a non-negative amount r_{ik} of required resource k . The STSP is the problem of determining a directed simple path between the root vertex and every terminal vertices, and of scheduling each activity belonging to the selected paths under the precedence and resource constraints, so as to minimize the completion time, i.e., the *makespan*. Because it is a generalization of two strongly NP-hard problems, the Resource-Constrained Project Scheduling Problem and the Steiner Tree in a Graph Problem, the STSP is strongly NP-hard.

Numerical illustration. Figures 1 and 2 depict a single-resource instance of the STSP with $|V| = 5$, $|A| = 5$, and $R = 10$. The set of vertices V is partitioned into a root vertex 1, a set of Steiner vertices $\{2, 3\}$ and a set of terminal vertices $\{4, 5\}$. The pair of values depicted on each arc corresponds to the duration d_i and the resource request r_i . In Figure 1a, arcs in bold correspond to the solution of the classical Steiner Minimal Tree Problem. In this solution, activities 4 and 5 cannot be scheduled simultaneously, and both the minimum total length and the best makespan are equal to 12 (Figure 1b). Figures 2 shows that a better makespan can be found in spite of a longer Steiner tree: the total length is equal to 14, and the makespan to 10.

P-HUB MAXIMAL COVERING PROBLEM AND EXTENSIONS FOR GRADUAL DECAY FUNCTIONS

Meltem Peker, Bahar Y. Kara

Department of Industrial Engineering,
Bilkent University, Ankara, Turkey

meltem.peker@bilkent.edu.tr, bkara@bilkent.edu.tr

Hubs are special facilities which are serving as switching, transshipment and sorting nodes in many large transportation and in telecommunication networks. In “*many to many*” distribution systems, instead of direct links for each origin-destination (O-D) pairs, flows are consolidated at specialized facilities and they are distributed to their destination points through them. Due to the consolidated flow at hubs, by exploiting economies of scale, reduced cost can be achieved for transferring flows between the hubs.

The hub location problem includes selection of the location of hub facilities and assignment of the demand nodes to these hubs in order to route the flow for each O-D pair. For routing of flows, two types of assignments are defined: In *single assignment*, each demand node is assigned to exactly one hub; all the incoming and outgoing flows are sent through that hub, in *multiple assignment*, the flows can be sent and received through more than one hub.

The hub location problem is proposed by O’Kelly [1]. Later, Campbell [2] introduces the variants of the hub location problem such as p -hub median, the uncapacitated hub location, p -hub center and hub covering problems. Among these problems, hub covering problems have not received enough attention so far in the literature [3]. Thus, in this research, we study on the p -hub maximal covering problem which maximizes covered demand with predetermined number of hubs. As well as defining the problem, Campbell [2] gives the mathematical formulations of single and multiple assignment versions of p -hub maximal covering problem. Later, Hwang and Lee [4] also develop a new mathematical formulation for the single assignment version.

In the two models given in [2] and [4], the value of coverage is considered as binary: any O-D pair is covered if the length of the path is within the maximum service distance (β_{ij}) and it is not covered at all if the length exceeds β_{ij} . This is a deficiency for the p -hub maximal covering problem because if the length of a path is $\beta_{ij} - \varepsilon$ it is considered as “*fully covered*” but if length is $\beta_{ij} + \varepsilon$ it is considered as “*not covered*”. However, in real life cases, there may be some situations where the coverage is not strict as given. Therefore, instead of binary coverage, “*partial coverage*” that changes with the distance sometimes may yield more realistic solutions. Partial coverage is first defined by Church and Roberts [5] for maximal covering location problem and later the formulation is improved by Berman and Krass [6]. However, there is no research related with this shortage about the maximal coverage hub location problem. Thus, we define the partial coverage hub location problem for the p -hub maximal covering problem where the “coverage” can be defined via a gradual decay function.

We propose a new mathematical formulation for the single assignment p -hub maximal covering problem which can also be used for the partial decay functions. The proposed formulation for single allocation has $O(n^2)$ binary variables and requires $O(n^3)$ constraints whereas the models proposed in [2] and [4] have $O(n^4)$ binary variables and $O(n^4)$ constraints.

Mathematical Analysis and Computational Results

The p -hub maximal covering problem for single assignment case can be formally stated as follows. Let N be the set of points for demand nodes and potential hub locations. C_{ij} is the distance from origin i to destination j , $\forall i, j \in N$. For every O-D pair, each path length from i to j is calculated by preprocessing with the equation $C_{ij}^{km} = C_{ik} + \alpha C_{km} + C_{mj}$, where α is the economies of scale parameter to be used to discount the cost for transferring flow between hubs, where $0 \leq \alpha \leq 1$. Also, there exists a maximum allowable service distance for each O-D pair denoted by β_{ij} . Then we define a_{ijkm} as a binary parameter to decide whether $i-j$ pair is covered by using hubs k and m or not:

$$a_{ijkm} = \begin{cases} 1 & \text{if } C_{ijkm} \leq \beta_{ij} \\ 0 & \text{otherwise} \end{cases} \quad \forall i, j, k, m \in N \quad (1)$$

The developed formulation can also be applicable to the case where coverage is defined via a gradual decay function. Instead of (1), the following can be used for the partial coverage:

$$a_{ijkm} = \begin{cases} 1 & \text{if } C_{ijkm} \leq \beta_{ij} \\ f(C_{ijkm}) & \text{if } \beta_{ij} \leq C_{ijkm} \leq \gamma_{ij} \\ 0 & \text{otherwise} \end{cases} \quad \forall i, j, k, m \in N \quad (2)$$

where f is any nonincreasing decay function and γ_{ij} is the upper bound for the partial coverage. Although the applicability of the formulations to the partial coverage is not defined in [2] and [4], those formulations can also be applied to the new problem with (2) for nonincreasing decay functions.

Computational experiments are performed to show the effectiveness of the new formulation. For preliminary analysis, we used *Civil Aeronautics Board* (CAB) data set given in [1]. Since CAB data set does not include any maximum service distance (β_{ij}), we used optimal objective function values (β) for the p -hub center problem given in [7]. For the maximum service distance, we used $\beta_{ij} = \beta * 0.75 \quad \forall i, j$. Then a_{ijkm} can be defined as:

$$a_{ijkm} = \begin{cases} 1 & \text{if } C_{ijkm} \leq \beta * 0.75 \\ 0 & \text{otherwise} \end{cases}$$

The results show that, [4], the most up to date formulation in the literature performs worse than the basic formulation proposed by Campbell [2]. To compare the results, we set a time limit of two hours and obtain the result using CPLEX. Based on 8 cases, on the average the gap between the best integer solution and the best bound is 927% for [4] and 9.26% for [2]. Therefore, both of the models fail to prove the optimality within two hours. However, with the proposed formulation, optimum results are achieved within 100 seconds.

For the partial coverage of the p -hub maximal covering problem, we use a step function for CAB data set for gradual decay function. Our proposed formulation again performs better than the other formulations. Even though the solution times are higher than the CPU of the binary coverage, with new developed formulation the problem is still solved within 6 minutes. Models given in [4] and [2] still find suboptimal results and on the average the gap is 1134.25% and 7.44% respectively.

The proposed model will also be tested on large data sets such as TR Data [8]. We expect that the proposed formulation with $O(n^2)$ structure will again perform significantly better than the ones in the literature. We also will analyze and discuss the multi allocation version of the problem.

References

- [1] O'Kelly, M.E (1987), “A quadratic integer program for the location of interacting hub facilities”, *European Journal of Operation Research* 302, 393-404.
- [2] Campbell, J. F. (1994), “Integer programming formulations of discrete hub location problems”, *European Journal of Operational Research* 72, 387–405.
- [3] Alumur, S. and Kara, B.Y. (2008), “Network hub location problems: The state of the art”, *European Journal of Operational Research* 190, 1–21.
- [4] Hwang, Y. H. And Lee, Y.H. (2012), “Uncapacitated single allocation p-hub maximal covering problem”, *Computers and Industrial Engineering*, 63, 382-389.
- [5] Church, R.L. and Roberts, K. L. (1983), “Generalized Coverage Models and Public Facility Location”, *Papers of the Regional Science Association*, 53, 117-135
- [6] Berman, O. and Krass, D. (2002), “The generalized covering location problem”, *Computers and Operations Research*, 29, 563-581.
- [7] Kara, B.Y. and Tansel, B.C. (2003), “The single assignment hub covering problem: Models and linearizations”, *Journal of the Operational Research Society*, 54, 59-64.
- [8] Tan, P. and Kara, B.Y. (2007) “A hub covering model for cargo delivery systems”, *Networks*, 49 (1), 28-39.

Approximating the capacitated facility location problem with length bounded trees

Andreas Bley,* Jannik Matuschke, Benjamin Müller

TU Berlin, Institute of Mathematics
Straße des 17. Juni 136, D-10623 Berlin, Germany

We consider a generalization of the Capacitated Facility Location problem, where clients may connect to open facilities via trees shared by multiple clients. The total demand of clients served by a single tree must not exceed a given tree capacity. Furthermore, the length of the path between a client and its facility within the corresponding tree must not exceed a given length bound. The task is to choose open facilities and shared service trees in such a way that the sum of the facility costs and the tree costs is minimal. This problem arises, for example, in the planning of optical access networks in telecommunications, where multiple clients may share a fiber tree if both fiber capacity and signal attenuation on the resulting connection paths permit.

We show that the problem is as hard as Set Cover in general and approximable with a constant factor for metric edge lengths that individually do not exceed the length bound. For the latter case, we present a general approximation algorithm that combines techniques for the simultaneous approximation of shortest path and minimum spanning trees with a Steiner tree based clustering approach and a greedy techniques to ensure short path lengths between facilities and clients. We also discuss some modifications of this algorithmic framework that yield better approximation ratios or additional solution properties in special cases.

* Supported by the DFG Research Center MATHEON – Mathematics for key technologies.

Routing and Scheduling Decisions in the Hierarchical Hub Location Problem

Okan Dükkanç, Bahar Yetiş Kara

Department of Industrial Engineering, Bilkent University, Ankara, Turkey

E-mail: okan.dukkanç@bilkent.edu.tr

Hubs are facilities that consolidate and disseminate demand flow in many-to-many distribution systems. Instead of having direct link between each origin-destination (o-d) pair, hubs provide the connection between each o-d pair by using fewer links and also concentrate demand flows to allow economies of scale. Hub Location problem considers decisions including locations of hubs on a network and also allocation of the demand points on the network to these hubs in order to ship the demand from its origin to its destination [1, 2, 3]. Multimodal version of hub location problem is related to hub location problem with more than one transportation mode [4]. Hierarchical multimodal hub location problem consider dividing the network into layers and each layer can have different network structure [5]. Customarily, each layer has a different mode of transportation.

The standard hub location problem is two-level with complete-star structure (complete for hub networks and star for allocations). In this study, we propose a hierarchical multimodal hub location problem with three-level (ring-star-star) structure (Figure 1). On the top layer, we also allow more than one ring.

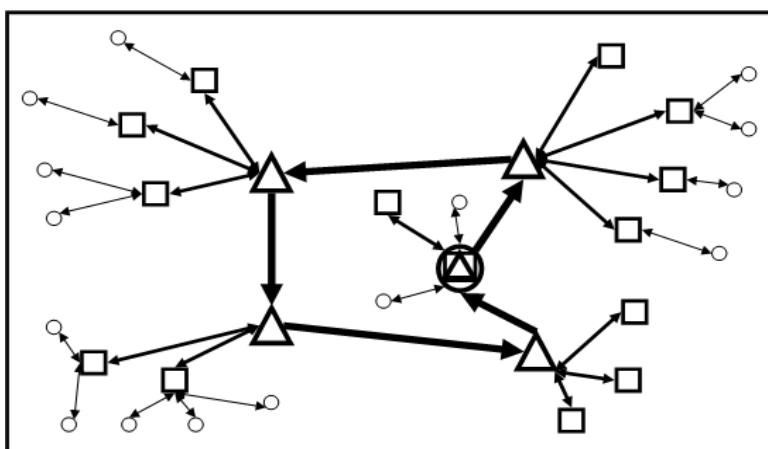


Figure 1: Representation of a network; small circles as demand points, squares as ground hubs, triangles are air hubs and big circle, square, triangle as central airport hub

The motivation behind the ring(s) structure on the top level is mainly to increase the utilization of the “best” vehicle on hand. Customarily, the arcs of the top layer hub network are served with faster vehicles, airplanes if possible. Having a complete hub network with airplanes will be very costly. Thus, we aim to utilize the top layer vehicles by incorporating scheduling decisions.

The aim of the proposed problem is, then, to cover all o-d pairs in a given time bound with minimum number of airplanes. The time bound in this problem has a very crucial role since according to the given time bound, the problem on the top layer can transform to Vehicle Routing Problem (VRP). Therefore, in this study, routing decisions are also included into hub location problem and thus a new problem is introduced to the literature. In this research, we consider “pick then deliver” type of service which means all demands are picked from origins and send to central airport hub, and then all demands are delivered to final destinations.

We propose a linear mixed integer mathematical model for the proposed problem. Given the node set, potential ground hub and air hub locations, the model outputs the network configuration with ground hubs, air hubs, the allocations and the required airline segments with the minimum number of airline links while obeying the time bound. We carried out preliminary analysis on the Turkish data set. Figure 2 shows 81 cities and 22 potential hub locations on the map of Turkey. Except three cities (3, 68 and 81), remaining 19 cities are considered as potential air hub locations and Ankara is considered as the central airport hub since its geographical location and all major companies have main sorting center in Ankara.

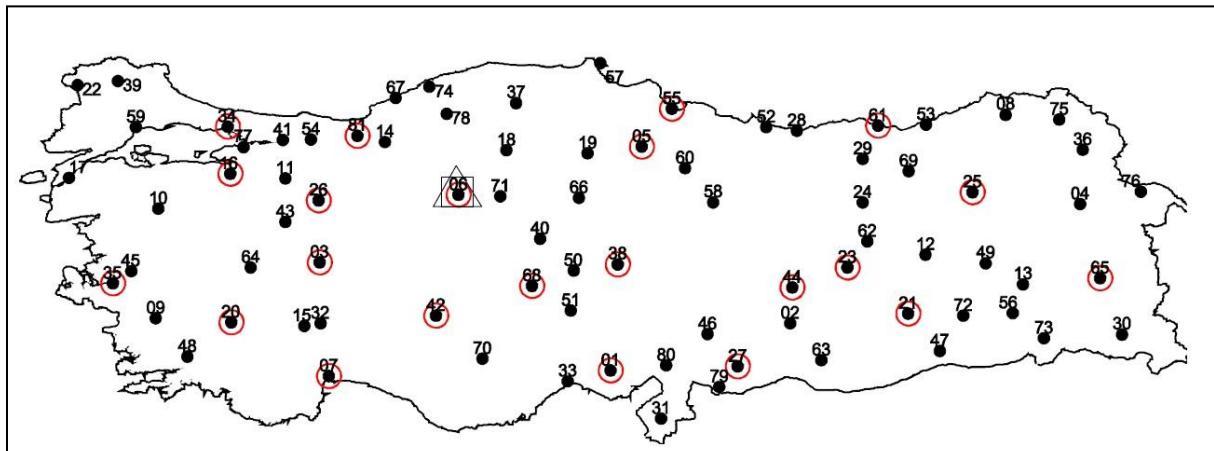


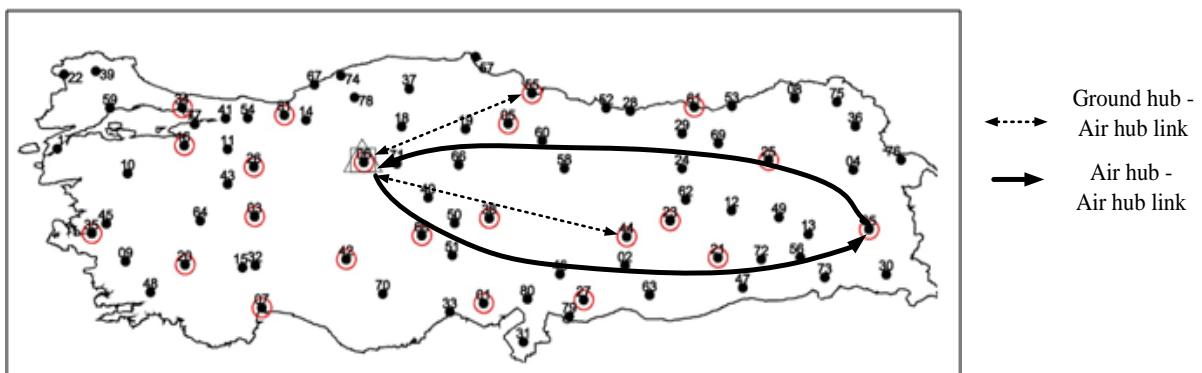
Figure 2: Turkey Map; 81 cities, 22 potential hub locations and Central Airport hub on the Turkish Network

We carried out our computational studies on a server with Intel Xeon E2-1220 processor and 16 GB of RAM and we used the optimization software Gurobi version 5.0.1. Given time bound (T) and total number of hubs (both ground and air) (p) as parameters, Figure 3 shows some of the results.

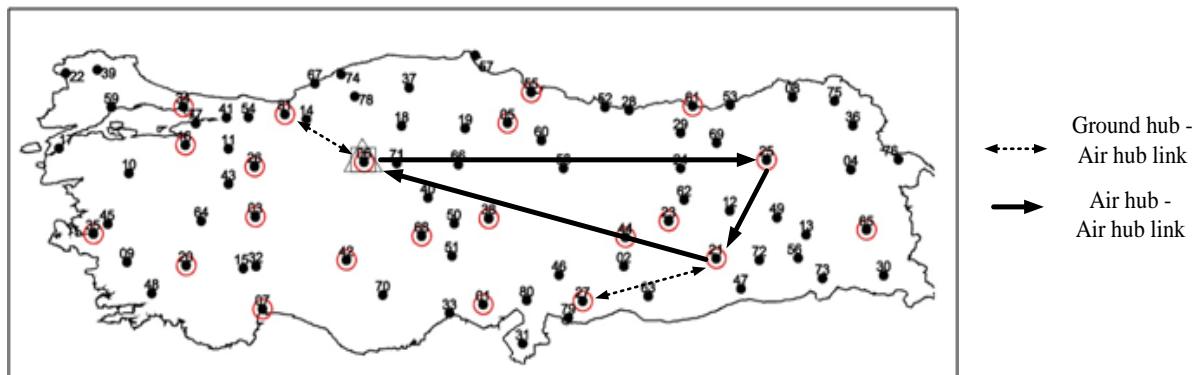
T (hr)	p	Hub Locations (Ground hubs→Air hubs)	Number of Airplanes	Total Number of Airplane Links	CPU Time(sec)
24	2	6,65	1	2	0,03
24	3	6,23,65→23	1	2	1,02
24	4	6,27,21→27,61→27	1	2	1,35
22	2	6,65	1	2	0,03
22	3	6,44,34→6	1	2	1,23
22	4	6,23,35→6,21→23	1	2	3,21
20	2	6,44	1	2	0,04
20	3	6,65,61→65	1	2	1,61
20	4	6,65,44→6,55→6	1	2	1,75
18	2	Infeasible	-	-	0,03
18	3	6,21,61→6	1	2	0,98
18	4	6,25,1→6,65→21	1	2	7,7
16	3	Infeasible	-	-	1,98
16	4	6,21,25,34→6	1	3	14,24
16	5	6,21,25,27→21,81→6	1	3	16,52
14	5	Infeasible	-	-	617,5
14	6	6,16,25,27,65,20→6	3	7	4364,2
13	6	Infeasible	-	-	4215,58

Table 1: Results with different time bounds and total number of hubs

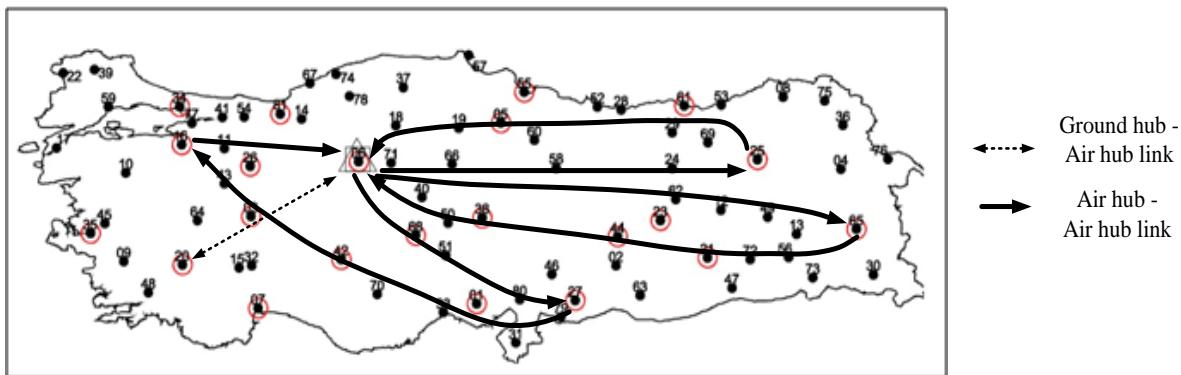
As can be seen from Table 1; 24 hour delivery is possible with 2 airline segments and 1 airplane. It is even possible to decrease time bound till 18 with only 2 airline segments. Depending on the time bound; the locations of hubs (ground and air) change. Interestingly, decreasing the time bound to 14 resulted in 7 airline segments with 3 airplanes. Figure 3 (a, b, c) depicts some of the results.



(a) $T = 20, p = 4$



(b) $T = 16, p = 5$



(c) $T = 14, p = 6$

Figure 3: Results on the Turkish Network

As a next step we will include routing and scheduling decisions in the second layer. Thus, we also will analyze the case (ring(s), ring(s), star) and discuss our findings.

References

- [1] O'Kelly, M.E., 1986. The location of interacting hub facilities. *Transportation Science*, 20(2), 92-105.
- [2] O'Kelly, M.E., 1986. Activity levels at hub facilities in interacting networks. *Geographical Analysis*, 18(4), 343-356.
- [3] O'Kelly, M.E., 1987. A quadratic integer program for the location of interacting hub facilities. *European Journal of Operational Research*, 32, 393-404.

- [4] O'Kelly, M.E., Lao, Y., 1991. Mode choice in a hub-and-spoke network: A zero-one linear programming approach. *Geographical Analysis*, 23, 283-397.
- [5] Yaman, H., 2009. The hierarchical hub median problem with single assignment. *Transportation Research Part B: Methodological*, 43(6), 643-658.

Multi-Band Robustness I: Model and Theory

Christina Büsing * Fabio D'Andreagiovanni †

The robustness concept introduced by Bertsimas and Sim in [3,4] has represented a breakthrough for robust linear optimization and robust network design [1,2]. In this setting the natural observation is taken into account that each uncertain parameter varies within a given interval but the number of values deviating from the lower bound is bounded by a parameter Γ . However, the central modeling assumption that the deviation band of each uncertain coefficient is single may be too restrictive in practice. In fact, the deviations in many real-world problems distribute asymmetrically within the single band. Breaking the band into multiple sub-bands to get a higher modeling resolution thus looks advisable.

This is the first talk of the three-talk session *Multi-Band Robustness*, where we introduce (i) the concept of Multi-Band Robustness, the supporting theory, solution algorithms, (ii) probabilistic analysis, construction of the uncertainty set, (iii) real-world network applications, and computational analysis.

In this talk, we will give an overview on our theoretical results for robust optimization problems under multi-band uncertainties (partially presented in [5]). Our main contributions are:

- a compact formulation for the robust counterpart of a MILP;
- an efficient method for the separation of robustness cuts (i.e., cuts that impose robustness), based on solving a min-cost flow instance;
- an analysis of 0-1 optimization problems with cost uncertainties given by a multi-band scenario set.

Acknowledgement

This work was supported by the German Federal Ministry of Education and Research (BMBF grant 03MS616A, project ROBUKOM, www.robukom.de).

References

- [1] P. Belotti, A. Capone, G. Carello, F. Malucelli, *Multi-layer MPLS network design: The impact of statistical multiplexing*, Computer Networks, 52 (6), 1291–1307, 2008.
- [2] W. Ben-Ameur and H. Kerivin, *Routing of uncertain demands*, Optimization and Engineering, 3, 283–313, 2005.
- [3] D. Bertsimas, M. Sim, *The Price of Robustness*, Operations Research, 52 (1), 35–53, 2004.
- [4] D. Bertsimas, M. Sim, *Robust discrete optimization and network flows*, Mathematical Programming B, 98, 49–71, 2003.
- [5] C. Büsing, F. D'Andreagiovanni, *New results about multi-band uncertainty in Robust Optimization*, Proc. SEA 2012, LNCS 7276, 63–74, 2012.

*RWTH Aachen University, Lehrst. f. Operations Research, Kackertstr. 7, 52072 Aachen, Germany, busing@or.rwth-aachen.de
†Zuse Institute Berlin, Takustr. 7, 14195 Berlin, Germany, d.andreagiovanni@zib.de

Multi-Band Robustness III: Application to Network Design Problems

Christina Büsing * Fabio D'Andreagiovanni † Arie M.C.A. Koster ‡
Manuel Kutschka ‡

In network design uncertain future demands have to be taken into account to achieve robust designs providing high quality and stable services. In practice, using the new multi-band concept allows a more refined and thus more cost-efficient model compared to the well-known Γ -robust approach [1].

This is the third talk of the three-talk session *Multi-Band Robustness*, where we introduce (i) the concept of Multi-Band Robustness, the supporting theory, solution algorithms, (ii) probabilistic analysis, construction of the uncertainty set, (iii) real-world network applications, and computational analysis.

In this talk we apply the multi-band robustness concept [2] to network design problems. To construct the multi-band robust model we use historical data from different real-world settings, e.g., realistic traffic measurements of the Abilene and GÉANT telecommunication networks. In an extensive computational study we address the computational tractability of the multi-band robust models. A comparison between the different solution algorithms is made. In addition, we compare the price of robustness of the multi-band robust solutions with that of the corresponding robust solutions obtained through the classical Γ -robustness model based on Bertsimas and Sim [1,4]. We show that the multi-band robustness is more cost-efficient in practice while keeping the level of protection according to the historical data. To evaluate the realized robustness of optimal multi-band and Γ -robust designs, we use the measures presented by Koster et al. [3].

Acknowledgement

This work was supported by the German Federal Ministry of Education and Research (BMBF grant 03MS616A, project ROBUKOM, www.robukom.de).

References

- [1] D. Bertsimas, M. Sim, *The Price of Robustness*, Operations Research, 52 (1), 35–53, 2004.
- [2] C. Büsing, F. D'Andreagiovanni, *New results about multi-band uncertainty in Robust Optimization*, Proc. SEA 2012, LNCS 7276, 63–74, 2012.
- [3] Arie M.C.A. Koster, M. Kutschka, C. Raack, *On the robustness of optimal designs*, Proc. ICC 2011, IEEE Xplore, 1–5, 2011.
- [4] Arie M.C.A. Koster, M. Kutschka, C. Raack, *Robust Network Design: Formulations, Valid Inequalities, and Computations*, to appear in Networks.

*RWTH Aachen University, Lehrst. f. Operations Research, Kackertstr. 7, 52072 Aachen, Germany, busing@or.rwth-aachen.de

†Zuse Institute Berlin, Takustr. 7, 14195 Berlin, Germany, d.andreagiovanni@zib.de

‡RWTH Aachen University, Lehrst. II f. Mathematik, Wüllnerstr. 5b, 52062 Aachen, Germany, {koster, kutschka}@math2.rwth-aachen.de

The Two-Level Diameter Constrained Spanning Tree Problem

Luis Gouveia, Markus Leitner, and Ivana Ljubić

Introduction and Problem Definition

Given a graph $G = (V, E)$ with edge costs $c_e \geq 0$, $e \in E$, the diameter constrained spanning tree problem (DMSTP) is to find a minimum spanning tree such that the distance (in number of edges) between every pair of nodes is at most some parameter $D_m \in \mathbb{N}$, $D_m > 1$. The DMSTP has a wide range of applications: in telecommunications, data compression, or parallel computing (see, e.g., [1, 3]). In telecommunication networks, for example, when multicasting is employed, the network latency between a pair of users is directly proportional to the length of the routing path. In a tree-multicasting, the maximum pairwise latency equals the diameter of that tree. The optimization goal consist of finding a minimum-cost spanning tree such that the maximum latency is restricted by some parameter D_m (see, e.g., [4]). In a more realistic scenario, media types transmitted over a network may range from text to video streaming, with different latency requirements being imposed for transmitting text, voice over IP, or video streaming in multi-player online games. In this work we will assume that there are two disjoint groups of nodes - those with more stringent latency requirements (e.g., video streaming users) and the remaining ones whose latency requirements are less demanding. At first glance, the assumption that there are two subsets with different

Luis Gouveia
Faculdade de Ciências, Universidade de Lisboa, DEIO, CIO Bloco C/2, Campo Grande,
1749-016 Lisbon, Portugal. e-mail: legouveia@fc.ul.pt

M. Leitner
Institute of Computer Graphics and Algorithms, Vienna University of Technology, Favoritenstr. 9-11, 1040 Vienna, Austria. e-mail: leitner@ads.tuwien.ac.at

I. Ljubić
Department of Statistics and Operations Research, University of Vienna, Brünnerstr. 72,
1210 Vienna, Austria. e-mail: ivana.ljubic@univie.ac.at

latency requirements might look too simplistic, but as we shall show, it is already rich enough to impose different and interesting research challenges.

More formally, we introduce a new problem which is a generalization of the DMSTP to which we will refer to as the *Two-Level Diameter Constrained Spanning Tree Problem* (2-DMSTP): Given a graph $G = (V, E)$ (where $|V| = n$) with edge costs $c_e \geq 0$, $e \in E$, and with the set of nodes V partitioned into two subsets: P , the subset of *primary* and more important nodes, and S , the subset of *secondary* and less important nodes. Two limits on the maximum length of communication paths are imposed: The maximum distance between nodes in P is allowed to be at most D' , and the maximum distance between nodes in S or between a node in P and a node in S is allowed to be at most D ($D > D' > 1$). The optimization goal consists of finding a minimum-cost spanning tree that satisfies these length restrictions.

Results

We first show that the 2-DMSTP can be solved in polynomial time if $D' = 2$ and $D = 3$ but is NP-hard for $D \geq 4$ and any $2 \leq D' < D$. Furthermore, we observe that the 2-DMSTP can be viewed as a diameter constrained spanning tree with diameter D that contains a diameter constrained *Steiner tree* with *terminal set* P and diameter D' . This permits us to use *center properties* for each of these two subproblems and to develop mixed integer programming (MIP) models that are more efficient than the traditional formulations based on the pairwise distance constraints.

The results presented are threefold: (a) Graph theoretical results: We first study feasible 2-DMSTP solutions from a graph theoretical perspective. We obtain upper bounds on the distance between the two centers and we provide necessary and sufficient conditions for those centers to be at minimum distance. (b) MIP models: We propose two new models, both relying on the concept of layered graphs. Layered graphs have been shown (see [2]) to provide the strongest MIP models for the DMSTP, both, from theoretical and computational perspective. Our first model can be viewed as an intersection of two layered graphs that independently model the Steiner tree and the spanning tree. On the other hand, the new graph theoretical results regarding the relative location of the two centers permit us to embed these properties in a layered graph construction. To do this, we propose a novel three-dimensional layered graph approach that also incorporates distance constraints w.r.t. primary nodes in its structure. To break symmetries, we use theoretical results regarding the minimum distance between the centers. (c) Computational results: Branch-and-cut algorithms are developed for the two proposed layered graph approaches. They are computationally tested on a set of benchmark instances for the DMSTP. They show that the novel three-dimensional layered graph model performs highly effective in practice.

References

1. N. Deo and A. Abdalla. Computing a diameter-constrained minimum spanning tree in parallel. In G. Bongiovanni et al., editors, *CIAC*, volume 1767 of *Lecture Notes in Computer Science*, pages 17–31. Springer, 2000.
2. L. Gouveia, L. Simonetti, and E. Uchoa. Modeling hop-constrained and diameter-constrained minimum spanning tree problems as Steiner tree problems over layered graphs. *Mathematical Programming*, 128:123–148, 2011.
3. T. Noronha, C. Ribeiro, and A. Santos. Solving diameter-constrained minimum spanning tree problems by constraint programming. *International Transactions in Operational Research*, 17(5):653–665, 2010.
4. K. Vik, P. Halvorsen, and C. Griwodz. Multicast tree diameter for dynamic distributed interactive applications. In *INFOCOM*, pages 1597–1605. IEEE, 2008.

Multi-Period Reverse Logistics Network Design

Sibel A. Alumur

*Department of Industrial Engineering
TOBB University of Economics and Technology
Ankara, Turkey
salumur@etu.edu.tr*

Stefan Nickel

*Institute of Operations Research
Karlsruhe Institute of Technology (KIT)
Karlsruhe, Germany
stefan.nickel@kit.edu*

Francisco Saldanha-da-Gama

*DEIO-CIO, Faculdade de Ciências
Universidade de Lisboa
Lisboa, Portugal
fsgama@fc.ul.pt*

Vedat Verter

*Desautels Faculty of Management
McGill University
Montreal, Canada
vedat.verter@mcgill.ca*

Keywords: Logistics, facility location, MILP, WEEE.

1 Introduction

Reverse logistics, as defined by the European Working Group on Reverse Logistics, is the process of planning, implementing and controlling backward flows of raw materials, in process inventory, packaging and finished goods, from a manufacturing, distribution or use point, to a point of recovery or point of proper disposal. In the broadest sense, this process involves collection, inspection, recycling, refurbishing, and remanufacturing of used or returned products, including leased equipment and machines.

An original equipment manufacturers (OEMs) direct involvement with the reverse logistics process is usually a function of the economic value that can be captured by processing the returned products. In this study, we focus on return streams with profit potential for the OEMs.

The configuration of the reverse logistics network is a complex problem comprising the determination of the optimal sites and capacities of collection centers, inspection centers, remanufacturing facilities, and/or recycling plants. In this study, we propose a new mathematical

formulation for the design of reverse logistics networks that is flexible to incorporate most of the reverse network structures plausible in practice.

In order to consider the possibility of making future adjustments in the network configuration to allow gradual changes in the network structure and in the capacities of the facilities, we consider a multi-period setting. We propose a multi-commodity formulation and use a reverse bill of materials in order to capture component commonality among different products and to have the flexibility to incorporate all plausible means in tackling product returns.

Our primary concern with this work is to propose a model which can be intrinsically simple but giving a strong basis for an easy extension to various settings. Moreover, this study intends to contribute to the reverse logistics literature by improving our understanding of how an OEM needs to react to the trends in the return streams and secondary markets by determining the extent of its involvement in reverse logistics so as to maximize its profits.

2 Mathematical Formulation

We introduce a mixed-integer linear programming (MILP) model for multi-period reverse logistics network design.

The proposed model addresses many features of practical relevance namely, a multi-period setting, modular capacities, capacity expansion of the facilities, reverse bill of materials, minimum throughput at the facilities, variable operational costs, finite demands in the secondary market, and a profit-oriented objective function.

Using our model, an organization responsible for the design and operation of the reverse network can decide on the network structure, while maximizing their profit. The decisions to be made include: When, where, and how many inspection centers to locate with which capacities, if it is profitable to establish remanufacturing facilities, when to invest for the capacity expansion for the facilities, the amounts of products or components to send to recycling facilities and external remanufacturing plants, and the amount of components to purchase for the remanufacturing plants.

In our multi-period setting, all decisions are taken over a planning horizon which is assumed to be finite and divided into several time periods. All the network design decisions are implemented in the beginning or end of the time periods. The decisions to be made in each time period comprise the location of inspection centers and remanufacturing facilities, capacity decisions, which include the initial capacities for the new facilities and possibly the expansion of the capacity in the existing ones, inventory to be held and components to purchase at the remanufacturing plants and the network flows.

A single organization is going to manage and operate this reverse logistics network. The revenues are assumed to be obtained from other companies. The organization gains revenues from recycling when the products or components are sold to recycling companies, from external remanufacturing when the products are sold to third party manufacturers, and from the

secondary market when the remanufactured products are sold to the market. Our proposed model determines the amount of products and components to be sent to recycling, external remanufacturing, and the secondary market. The recycling, external remanufacturing and the secondary market are strategic options within the proposed model rather than being geographically located facilities.

3 A Case Study

We applied our model on a case study inspired by a real life problem in Germany in the context of reverse logistics network design for washing machines and tumble dryers. We do not solve any specific company's problem. However, all parameters are chosen in a realistic order of magnitude. Our goal is to highlight the features of the proposed model and also to demonstrate that the model can be solved to optimality using a commercial solver for instances of a realistic size.

4 Sensitivity Analysis and Insights

The computational tests performed with the initial setting are extended so that we can get a better perception of the potential and value of our model, and also to see how the changes in the problem parameters can affect the optimal solution.

To this end, we initially analyze the sensitivity of the solutions to the changes in the amount of products supplied from the collection centers. Secondly, we look at the sensitivity of the location decisions to changes in the set-up costs and the capacities. To do so, we vary the set-up costs and the capacities of the modules and analyze the outcomes of the model. For each set-up cost, we test the model with three different capacity configurations, tight, medium and loose, each capacity configuration containing two capacity modules for each of the facilities.

One important aspect to be analyzed concerns the potential benefits for the company of using the proposed multi-period model, rather than utilizing a static model; that is, the value of the multi-period model. Our aim is to find whether the company will achieve additional profits by using the proposed multi-period model. For this analysis, we compare the profit performance of a single-period model with that of the multi-period model over the 5-year profit horizon on various instances.

Moreover, we vary all the unit revenues by 10% and 20% and study the sensitivity of the solutions to the changes in the unit revenues. Additionally, we analyze the changes in the optimal solution with alternative growth scenarios for the amounts of products generated.

5 Conclusions

Through our computational analysis, instances with realistic sizes could be solved to optimality by using a commercial solver. We were able to show that there can be gains in the profit by using

a multi-period model compared to using a static one. In our results, inspection centers and remanufacturing plants were often co-located due to potential savings in the transportation costs. We showed through sensitivity analysis that optimal solutions are robust to some changes in the problem data.

As a conclusion, in this study, we proposed a comprehensive MIP model for multi-period reverse logistics network design. The model accommodates several features of practical relevance. We presented case study for washing machines and tumble dryers within WEEE in Germany. We conducted extensive parametric and scenario analysis emphasizing managerial aspects and also proposed a measure for the value of the multi-period model.

Traffic Node Assignment in Wireless Networks: A Multi-Band Robust Knapsack Approach

Grit Claßen * † Arie M.C.A. Koster * Manuel Kutschka * Anke Schmeink †

The traffic node assignment problem as part of the wireless network planning problem [1] maximises the number of covered traffic nodes by the installed base stations guaranteeing a sufficient service for each assigned traffic node. One restriction is that a traffic node can only be assigned to at most one base station due to hard handover in future wireless networks. To ensure the fulfilment of the data rate requirements of every assigned traffic node, the base station capacity should not be exceeded. These capacity limitations can be formulated as knapsack constraints in an integer linear program formulation for the assignment problem. Introducing Lagrangian multipliers for the hard handover conditions, the Lagrange subproblem consists of a knapsack problem for every base station.

Demand fluctuations have to be incorporated in the model to guarantee sufficient available bandwidth during operation of the wireless network. Robust optimization is a widely accepted approach to handle such uncertainties. In this talk, we consider the Γ -robust knapsack problem where the weights of the items represent the uncertain data rate requirements of the traffic nodes. The weights of the items deviate in a predefined range, cf. Bertsimas and Sim [2]. The number of simultaneously deviating items is limited by a robustness parameter Γ . However, the restriction to one deviation interval for each item is often too rough for real-world applications leading to too conservative solutions. A refinement of the Γ -robustness is the so-called *multi-band robustness* where the deviation range is partitioned into several sub-bands [3]. The maximal number of deviations lying in each band is bounded by one parameter per band.

Dynamic programming is an appropriate technique to solve classical knapsack problems [4] and can also be applied to the Γ -robust knapsack due to a compact formulation of the deviations. In this talk, we extend the dynamic programming approach for the Γ -robust problem to the multi-band robust knapsack problem. We define domains for the additional parameters introduced by the multi-band approach such that the running time of the dynamic program remains linear in the knapsack capacity for the case of at most three bands.

An efficient solving of multi-band robust knapsack problems implies an efficient solving of the Lagrange relaxation of the traffic node assignment problem providing lower bounds on the maximum number of covered traffic nodes.

Acknowledgement

This work was supported by the UMIC Research Centre at RWTH Aachen University and the DFG research grant KO2311/3-1, SCHM2643/5-1.

References

- [1] G. Claßen, A. M. C. A. Koster, A. Schmeink, *A Robust Optimisation Model and Cutting Planes for the Planning of Energy-Efficient Wireless Networks*, Computers and Operations Research, 40(1), 80–90, 2013.
- [2] D. Bertsimas, M. Sim, *The Price of Robustness*, Operations Research, 52 (1), 35–53, 2004.
- [3] C. Büsing, F. D’Andreagiovanni, *New Results about Multi-band Uncertainty in Robust Optimization*, Proc. SEA 2012, LNCS 7276, 63–74, 2012.
- [4] P. Toth, *Dynamic Programming Algorithms for the Zero-One Knapsack Problem*, Computing, 25, 29–45, 1980.

*RWTH Aachen University, Lehrstuhl II für Mathematik, Wüllnerstr. 5b, 52062 Aachen, Germany, {klassen, koster, kutschka}@math2.rwth-aachen.de

†RWTH Aachen University, UMIC Research Centre, Mies-van-der-Rohe-Str. 15, 52074 Aachen, Germany, schmeink@umic.rwth-aachen.de

An incremental algorithm for the uncapacitated facility location problem

Ashwin Arulselvan^{1,2} Olaf Maurer^{1,3} Martin Skutella^{1,4}

*Institut für Mathematik
Technische Universität Berlin
Straße des 17. Juni 136, 10623 Berlin, Germany*

Abstract

We study the incremental facility location problem, wherein we are given an instance of the uncapacitated facility location problem. We seek an incremental sequence of opening facilities and an incremental sequence of serving customers along with their fixed assignments to facilities open in the partial sequence. Our aim is to have the solution obtained for serving the first ℓ customers in the sequence be competitive with the optimal solution to serve any ℓ customers. We provide an incremental framework that provides an overall competitive factor of 8 and a worst case instance that provides the lower bound of 3. The problem has applications in multi-stage network planning.

Keywords: incremental algorithm, facility location problem, network design

¹ This work was funded by DFG research center Matheon

² Email: arulsel@math.tu-berlin.de

³ Email: maurer@math.tu-berlin.de

⁴ Email: martin.skutella@tu-berlin.de

1 Introduction

The problem studied is motivated by the need to deploy telecommunication networks in phases due to budget, time or resource restrictions. In practical situations, problems arising in network planning require incremental solutions that help planners to build the network in stages in the most efficient way. The uncapacitated facility location problem (UFLP) with metric assignment cost has a long line of research [1,3,8]. The best approximation guarantee known for this problem is 1.48 [5]. The problem has also been studied in incremental settings [7,6] but prior work involve in different settings compared to one considered in our work. The authors in previous work were interested in a nested sequence of facilities and a threshold sequence for the scaling of the assignment costs. A solution in the sequence is a solution corresponding to a specific scaling factor, which could be inferred from the threshold sequence. Each solution in the sequence serves all customers and we differ from them in our problem definition.

The incremental facility location problem (IncFLP) is based on the robust facility location problem (RFLP) [2]. In that problem, we are given a set F of potential facility locations and a set R of customers. We are also given a service cost $c_S : R \times F \rightarrow \mathbb{R}_+$, a facility opening cost $c_F : F \rightarrow \mathbb{R}_+$ and a number ℓ of customers to be connected. We seek a set of facilities F^* and a set customers R^* incurring minimal cost with $|R^*| \geq \ell$. We denote the robust facility location problem with given number ℓ by ℓ -RFLP.

An instance of IncFLP comprises of the same input as the RFLP without the number ℓ . Let n the number of customers in the instance. We seek nested sets of facilities, customers and assignment edges between them such that for all $\ell = 1, \dots, n$, the induced solutions of the ℓ -RFLPs are k -approximative for some k .

We need more notation to make this precise: For a subset of facilities $L \subseteq F$, we denote the total facility cost of this set by $c_F(L)$. We denote the service cost obtained by serving a set of customers $M \subseteq R$ by a set of facilities $L \subseteq F$ by $c_S(M, L) = \sum_{j \in M} c_S(j, L)$, where $c_S(j, L)$ is the cost of the cheapest assignment of customer j to a facility in L . For an edge subset E , let $c_S(M, L, E) = \sum_{j \in M} c_S(j, L, E)$, where $c_S(j, L, E)$ is the cost of the cheapest assignment of customer j to a facility in L existing in E . Let $m = |F|$ be the number of facilities and $n = |R|$ the number of customers.

In the incremental version, the goal is to give a sequence to serve customers

$$R_1 \subsetneq R_2 \subsetneq \dots \subsetneq R_n = R$$

where $|R_\ell| = \ell$ and a sequence to open facilities

$$F_1 \subseteq F_2 \subseteq \cdots \subseteq F_n$$

such that the customers in $R_\ell \setminus R_{\ell-1}$ could be assigned only to facilities in F_ℓ , assuming $R_0 = \emptyset$. The objective is to minimize the competitive ratio of the sequence. The competitive ratio of a sequence is defined as

$$\max_{\ell=1 \dots n} \frac{c_F(F_\ell) + \sum_{\ell=1}^n c_S(F_\ell, R_\ell \setminus R_{\ell-1})}{OPT_\ell},$$

where OPT_ℓ is the optimal cost of the ℓ -RFLP.

Let A be an approximation algorithm for the RFLP. In our approximation algorithm, we will treat A as a black-box. By writing $(Z, M) = A(F, R, \ell)$ we mean that A takes as input the set of potential facilities locations F , the set of customers R and an integer, $\ell \leq n$, number of customers to be served. It produces the output (Z, M) , where $Z \subseteq F$ is the set of facilities opened and $M \subseteq R, |M| = \ell$ is the set of customers served by the facilities in Z . The algorithm does not need to return the actual assignment as it is easily computable. We are now ready to provide our incremental framework $FacInc(F, R)$.

2 Algorithm $FacInc(F, R)$

We start with a solution to the uncapacitated facility location problem obtained from any approximation algorithm. Let $F' \subseteq F$ be the set of facilities opened in this solution. The algorithm runs in two phases. The first phase constructs good partial solutions and in the second phase these partial solutions are glued together to construct an incremental solution. Let $D(F, R)$ denote the solution obtained from the solution (F, R) by removing exactly one customer with the highest service cost (ties broken arbitrarily) and removing any unused facilities (if any) from the solution.

Incremental Phase:

- 1: Initialize: $k = 1, F_n = F', R_n = R$
- 2: **for** $\ell = (n - 1)$ to 1 **do**
- 3: $(F_\ell, R_\ell) = D(F_{\ell+1}, R_{\ell+1})$ and $(L_O, M_O) = A(F, R, \ell)$
- 4: **if** $c_F(F_\ell) + c_S(F_\ell, R_\ell) \geq 2c_F(L_O) + c_S(M_O, L_O)$ **then**
- 5: $F_\ell = L_O$
- 6: $R_\ell = M_O$

```

7:       $k = k + 1$ 
8:  end if
9: end for
10: Let  $K = k$ 
```

We call the point when the “If” condition in step 4 passes as a refinement point. We have a total of K such refinement points. We shall reverse the indexing of the solutions from 1 to K , in the increasing fashion (smallest to largest), in order to help us present the analysis with clarity. Let us denote these solutions as $(F_1, R_1), (F_2, R_2), \dots, (F_K, R_K)$. Now (F_1, R_1) and (F_K, R_K) have the least and most number of customers respectively. Let r_k be the number of customers served in the k^{th} refinement point solution. Note that we can construct good solutions (within some constant factor of the optimal) from (F_k, R_k) to serve $\ell = (r_{k-1} + 1) \dots r_k$ customers. In the second phase, we glue these partial solutions together to construct an incremental solution. We denote by $J(F, R, k)$ the set of k customers from the set R with the cheapest service cost when assigned to the facilities in F and we denote such a service cost by $c_S(F, R, k)$.

Refinement Phase:

```

1: Initialize:  $\tilde{R}_0 = R_0$ ,  $\tilde{F}_0 = F_0$ 
2: for  $k = 0$  to  $K - 1$  do
3:    $\tilde{F}_{k+1} = \tilde{F}_k \cup F_{k+1}$ 
4:   Let  $J_{k+1} = J(\tilde{F}_{k+1}, R_{k+1} \setminus \tilde{R}_k, r_{k+1} - r_k)$ 
5:    $\tilde{R}_{k+1} = \tilde{R}_k \cup J_{k+1}$ 
6: end for
```

We now have our incremental sequence of customers and facilities:

$$\tilde{R}_0 \subseteq \tilde{R}_1 \dots \subseteq \tilde{R}_{K-1} \subseteq \tilde{R}_K = R \quad (1)$$

$$\tilde{F}_0 \subseteq \tilde{F}_1 \dots \subseteq \tilde{F}_{K-1} \subseteq \tilde{F}_K \quad (2)$$

We are yet to explain why this is an incremental sequence at the non-refinement or intermediate points. We will do this later.

3 Analysis

We would first like to show that for all $k = 1, \dots, K$, the following inequality holds:

$$c_F(\tilde{F}_k) + c_S(\tilde{F}_k, \tilde{R}_k) \leq 2c_F(A(F, R, r_k)) + c_S(A(F, R, r_k)) \quad (3)$$

where A is the approximation algorithm used as our black-box. Our refinement condition at step 4 of the refinement phase implies

$$2c_F(F_k) + c_S(F_k, R_k) \leq c_F(F_{k+1}) + c_S(F_{k+1}, R_{k+1}, r_k), \forall k \in \{1 \dots K-1\} \quad (4)$$

Lemma 3.1

$$\begin{aligned} 2c_F(F_k) + c_S(F_k, R_k) + c_F(F_{k+1}) + c_S(F_{k+1}, R_{k+1} \setminus \tilde{R}_k, r_{k+1} - r_k) \\ \leq 2c_F(F_{k+1}) + c_S(F_{k+1}, R_{k+1}), \text{ for all } k \in \{0, \dots, K-1\} \end{aligned}$$

Proof. Add $c_F(F_{k+1}) + c_S(F_{k+1}, R_{k+1} \setminus \tilde{R}_k, r_{k+1} - r_k)$ to both sides of the refinement condition 4. The right hand side is then equal to

$$\begin{aligned} 2c_F(F_{k+1}) + c_S(F_{k+1}, R_{k+1}, r_k) + c_S(F_{k+1}, R_{k+1} \setminus \tilde{R}_k, r_{k+1} - r_k) \\ \leq 2c_F(F_{k+1}) + c_S(F_{k+1}, R_{k+1}, r_{k+1}) = 2c_F(F_{k+1}) + c_S(F_{k+1}, R_{k+1}). \end{aligned}$$

The inequality is due to the observation that the cost of choosing the cheapest r_k customers and some $r_{k+1} - r_k$ customers from the set R_{k+1} must be less than the cost of choosing all r_{k+1} customers. \square

Lemma 3.2

$$\begin{aligned} 2c_F(F_1) + c_S(F_1, R_1) + \sum_{j=2}^k \left[c_F(F_j) + c_S(F_j, R_j \setminus \tilde{R}_{j-1}, r_j - r_{j-1}) \right] \\ \leq 2c_F(F_k) + c_S(F_k, R_k), \text{ for all } k \in \{0, \dots, K\} \quad (5) \end{aligned}$$

The left hand side is obviously an upper bound to the cost of the incremental solution.

Proof. For $k = 1$, the claim is true from Lemma 3.1. Let the claim be proven for k . We prove it for $k + 1$ and we know that (5) is true for k . We have to

show

$$\begin{aligned} 2c_F(F_1) + c_S(F_1, R_1) + \sum_{j=2}^{k+1} \left[c_F(F_j) + c_S(F_j, R_j \setminus \tilde{R}_{j-1}, r_j - r_{j-1}) \right] \\ \leq 2c_F(F_{k+1}) + c_S(F_{k+1}, R_{k+1}) \end{aligned}$$

By Lemma 3.1, we know that

$$\begin{aligned} 2c_F(F_k) + c_S(F_k, R_k) + c_F(F_{k+1}) + c_S(F_{k+1}, R_{k+1} \setminus \tilde{R}_k, r_{k+1} - r_k) \\ \leq 2c_F(F_{k+1}) + c_S(F_{k+1}, R_{k+1}) \end{aligned}$$

By applying the induction hypothesis, we get

$$\begin{aligned} 2c_F(F_1) + c_S(F_1, R_1) + \sum_{j=2}^{k+1} \left[c_F(F_j) + c_S(F_j, R_j \setminus \tilde{R}_{j-1}, r_j - r_{j-1}) \right] \\ \leq 2c_F(F_k) + c_S(F_k, R_k) + c_F(F_{k+1}) + c_S(F_{k+1}, R_{k+1} \setminus \tilde{R}_k, r_{k+1} - r_k) \\ \leq 2c_F(F_{k+1}) + c_S(F_{k+1}, R_{k+1}) \end{aligned}$$

□

We need to show that we can upper bound the incremental solution at intermediate points (non-refinement points) as well. In order to do this, we first need to explain how to retrieve an incremental solution at these points from the incremental solution we constructed. If we notice the incremental solution (1) and (2), we only specified the nested subsets of customers and facilities at the refinement points. Let us construct the customer set J_{k+1} in step 4 of the refinement phase in the following equivalent way: iteratively augment \tilde{R}_k by one customer until the size of the set reaches r_{k+1} . We denote this set after p augmentations by \tilde{R}_{k+1}^p . Let $R_{k+1}^p \subset R_{k+1}$, be the subset of customers of size $p = r_k + 1, \dots, r_{k+1}$, that we accepted in the refinement phase (If condition fails) and $F_{k+1}^p \subset F_{k+1}$ be the set of facilities that are serving these customers in that accepted solution. In each iteration, the new customer is picked from the set $R_{k+1}^p \setminus \tilde{R}_k^{p-1}$ and note that at least one such customer exist, since the cardinalities of these two sets differ exactly by 1. Now, we are ready to prove the following Lemma (which is similar to Lemma 3.1).

Lemma 3.3 *For all $p = |R_k| + 1, \dots, |R_{k+1}|$, we have the following true,*

$$\begin{aligned} 2c_F(F_k) + c_S(F_k, R_i) + c_F(F_{k+1}^p) + c_S(F_{k+1}^p, R_{k+1}^p \setminus \tilde{R}_k^{p-1}, 1) \\ \leq 2c_F(F_{k+1}) + c_S(F_{k+1}^p, R_{k+1}^p). \end{aligned}$$

Proof. The proof is identical to the proof of Lemma 3.1. The additional thing to observe is the fact that the sets $F_{k+1}^{r_k+1} \subseteq \dots \subseteq F_{k+1}^p \subseteq \dots \subseteq F_{k+1}^{r_{k+1}}$ are nested, since we only close down a facility if it is not serving any more customers. Hence, when we picked a customer from the set $R_{k+1}^p \setminus \tilde{R}_k^{p-1}$, we can pick the facility from the set F_{k+1}^p that was serving this customer into our incremental solution. The rest of the arguments are similar to proof of Lemma 3.1. \square

Now, with Lemma 3.3 and Lemma 3.2, we have established the following theorem.

Theorem 3.4 *For all $p = r_k + 1, \dots, r_{k+1}$ and for all $k \in \{0, \dots, K - 1\}$, we have the following true,*

$$\begin{aligned} & 2c_F(F_1) + c_S(F_1, R_1) + \sum_{j=2}^k \left[c_F(F_j) + c_S(F_j, R_j \setminus \tilde{R}_{j-1}, r_j - r_{j-1}) \right] \\ & + c_F(F_{k+1}^p) + c_S(F_{k+1}^p, R_{k+1}^p \setminus \tilde{R}_k^{p-1}, 1) \\ & \leq 2c_F(F_{k+1}) + c_S(F_{k+1}^p, R_{k+1}^p). \end{aligned}$$

4 Lower bound

The lower bound here imply the price we need to pay for seeking an incremental solution. The instances we present cannot have any sequence that can have a competitive factor less than 3. We give a family of instances, which yield a lower bound of at least 2.99 for $m \geq 200$, as can be evaluated computationally. The ratio for this construction does not seem to exceed a value of 3.

Let there be n facilities. Each facility has zero-cost-connections to a set R_i with 2^i customers. All sets R_i are mutually disjoint. The connection cost of all other connections is some constant $M \gg 0$. We refer to the facility together with its customers of zero service cost as a cluster.

Let x_0, \dots, x_{m-1} be rest facility opening costs. Consider the following system of linear inequalities:

$$x_{i+1} + \sum_{j=0}^{i-1} x_j \geq \alpha \cdot x_i \quad \forall i = 1, \dots, m-1.$$

α is the achieved minimal competitive ratio by these inequalities. As an example, for $m = 4$ this system is feasible for $\alpha = 2.246$, but infeasible for

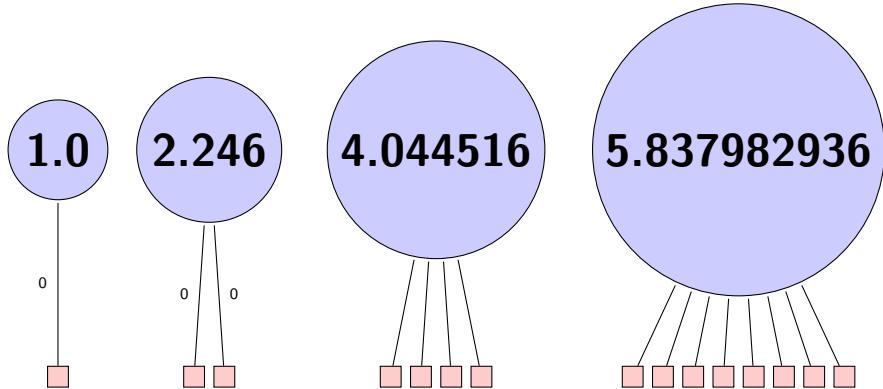


Fig. 1. Lower bound example for $m = 4$, $\alpha = 2.246$. All assignment costs are zero.

$\alpha = 2.247$. So by setting the facility costs to a solution of this system, we get a lower bound of 2.246.

The intuitive explanation behind these inequalities is the following. Suppose one looks for a solution of the RFLP with $\ell = 2^k$ customers. To keep the competitive ratio down below, one needs to open the first cluster, as the second cluster is α times as expensive. The following inequalities give an incentive to opening the clusters one by one in the order of their size, as the competitive ratio for the sequence is immediately as bad as α as soon as one skips a cluster.

An example for $m = 4, \alpha = 2.246$ can be seen in Figure 1.

5 Robust facility location problem

The best known approximation guarantee known for the robust facility location problem is 2 [4]. The LP has an integrality gap of 2 even after parametric pruning. This worst case example works on an instance with zero service cost, so we cannot hope to exploit the unequal guarantees that we achieved from our framework for the facility and service cost. This is the best one could hope to achieve with the known techniques to solve the facility location problem. If we plug in the algorithm of Jain et.al. [4] in our framework above, we can solve the incremental version that guarantees a competitive factor 8. For the worst case instance presented in section 4, the ℓ -RFLP could solved exactly as it is a knapsack problem with the budget being ℓ and our guarantee could be improved to 4 for these instances.

References

- [1] Jaroslaw Byrka and Karen Aardal. An optimal bifactor approximation algorithm for the metric uncapacitated facility location problem. *SIAM J. Comput.*, 39(6):2212–2231, 2010.
- [2] Moses Charikar, Samir Khuller, David M. Mount, and Giri Narasimhan. Algorithms for facility location problems with outliers. In *Proceedings of the twelfth annual ACM-SIAM symposium on Discrete algorithms*, SODA ’01, pages 642–651, Philadelphia, PA, USA, 2001.
- [3] Fabian A. Chudak and David B. Shmoys. Improved approximation algorithms for the uncapacitated facility location problem. *SIAM J. Comput.*, 33(1):1–25, 2003.
- [4] Kamal Jain, Mohammad Mahdian, Evangelos Markakis, Amin Saberi, and Vijay V. Vazirani. Greedy facility location algorithms analyzed using dual fitting with factor-revealing LP. *J. ACM*, 50(6):795–824, 2003.
- [5] Shi Li. A 1.488-approximation algorithm for the uncapacitated facility location problem. In *International Colloquium on Automata, Languages and Programming (ICALP)*, pages 77–88, 2011.
- [6] Guolong Lin, Chandrashekhar Nagarajan, Rajmohan Rajaraman, and David P. Williamson. A general approach for incremental approximation and hierarchical clustering. *SIAM J. Comput.*, pages 3633–3669, 2010.
- [7] Charles Greg Plaxton. Approximation algorithms for hierarchical location problems. In *Proceedings of the 35th Annual ACM Symposium On Theory of Computing*, pages 40–49, 2002.
- [8] David Shmoys, Eva Tardos, and Karen Aardal. Approximation algorithms for facility location problems. In *ACM Symposium on Theory of Computing*, page 265274, 1997.

Multi-Band Robustness II: Constructing the Uncertainty Set

Christina Büsing * Fabio D'Andreagiovanni † Arie M.C.A. Koster ‡
Manuel Kutschka ‡

The essence of robust optimization is to find optimal solutions that remain feasible under all deviations included in a specified uncertainty set, that should reflect the risk aversion of the decision maker. Though the definition of the uncertainty set constitutes a crucial issue [2], the question of how building it has been virtually neglected in literature.

This is the second talk of the three-talk session *Multi-Band Robustness*, where we introduce (i) the concept of Multi-Band Robustness, the supporting theory, solution algorithms, (ii) probabilistic analysis, construction of the uncertainty set, (iii) real-world network applications, and computational analysis.

In this talk, we focus on the uncertainty sets following the new multi-band robustness concept [3,4]. First, we derive probabilistic bounds on the violation of a constraint: a robust optimal solution is completely protected against deviations included in the uncertainty set, but it might still be infeasible in case of realizations not explicitly included in the considered set. In contrast to the classical bounds for Γ -robustness [1], our bound has the property of being strongly data driven, since it exploits past observations about coefficient deviations, that are commonly available in real applications. Then we present several rules to define the multiple bands and to set the number of realizations that may fall in each band. The rules are based on assumed distributions of the coefficients in the deviation intervals or on historical data. Finally, we present computational experiments on the knapsack problem to show how the different construction rules influence the robust solution.

Acknowledgement

This work was supported by the German Federal Ministry of Education and Research (BMBF grant 03MS616A, project ROBUKOM, www.robukom.de).

References

- [1] D. Bertsimas, M. Sim, *The Price of Robustness*, Operations Research, 52 (1), 35–53, 2004.
- [2] D. Bertsimas, D. B. Brown, *Constructing Uncertainty Sets for Robust Linear Optimization*, Operations Research, 75 (6), 1483–1495, 2009.
- [3] C. Büsing, F. D'Andreagiovanni, *New results about multi-band uncertainty in Robust Optimization*, Proc. of SEA 2012, LNCS 7276, 63–74, 2012.
- [4] C. Büsing, F. D'Andreagiovanni, *Robust Optimization under Multi-band Uncertainty*, Working paper, submitted for publication, 2012.

*RWTH Aachen University, Lehrst. f. Operations Research, Kackerstr. 7, 52072 Aachen, Germany, busing@or.rwth-aachen.de

†Zuse Institute Berlin, Takustr. 7, 14195 Berlin, Germany, d.andreagiovanni@zib.de

‡RWTH Aachen University, Lehrst. II f. Mathematik, Wüllnerstr. 5b, 52062 Aachen, Germany, {koster, kutschka}@math2.rwth-aachen.de

Earthwork Optimization Models for the Construction of a Large Highway System

Christian Bogenberger⁽¹⁾, Mauro Dell'Amico⁽²⁾, Gerhard Hoefinger⁽¹⁾,
Manuel Iori⁽²⁾, Stefano Novellani⁽²⁾, Barbara Panicucci⁽²⁾

(1) BPM, STRABAG, Ungargasse 64-66/4/2, 1030 Vienna, Austria
`{christian.bogenberger; gerhard.hoefinger}@strabag.com`

(2) DISMI, University of Modena and Reggio Emilia,
Via Amendola 2, 42122 Reggio Emilia, Italy

`{mauro.dellamico; manuel.iori; stefano.novellani; barbara.panicucci}@unimore.it`

Extended Abstract

We describe the activity that we performed to model the construction of *Autostrada Pedemontana Lombarda*, a large system of highways, access ramps and local streets that is now being built in the urban area of Milan, in Northern Italy. To minimize the impact over the residential areas involved, most of the highway is going to be built in trench, a few meters under the ground level. This implies that a huge amount of earth (tens of millions of cubic meters) has to be dug and moved to different locations by means of special excavators and trucks.

The goal of the activity has been to devise a support tool that aids managers to schedule the construction activities. These involve decisions on the use of several quarries and dump sites, the temporary rent of landfill areas, the installation of recycling plants, and the use of a large fleet of vehicles to dig and move earth. The final outcome of the tool we developed includes the amount of quantity of each material that must be moved in every time period, its origin and its destination, and allows to obtain a target profile of the overall construction process in the planning horizon at minimum cost.

The problem is far from being trivial. The construction of a highway includes indeed various activities to be done in different periods of the planned horizon. To model the process, the highway building site is first divided into several portions, called locations. Each location can require earth to be excavated (digging activity) and/or material to be brought there for the construction of the infrastructure (filling activity). The total amount of material to be dug is imposed by the construction plan and is divided into several types of materials, following the geological characteristics of the ground. The types and amounts

of materials to be brought to each location depend on the need of filling earth cuts, building bridges, artificial galleries, trenches, etc.

Dug material can be of low quality and then must be sent to dump sites, or can be of good quality and then be used directly for filling. It can also be sent to recycling plants to be refined and/or mixed with other materials, to obtain composite material that is then used for a different filling. In order to avoid throwing away good materials, some landfill areas are rented in proximity of the building site, to store some material when not needed and used in the following periods in which a lack of that material will occur.

The material used to respond to filling requests can not only come directly from digging or from temporary landfill areas, but can even be bought from private quarries by paying according to the required quantity. Quarries, dump sites and landfill areas have a restriction on the total quantity of material that can be bought, thrown away and stored, respectively.

Each activity is bounded in time by a time window, imposed by the master production schedule of the construction process. The overall horizon is discretized in weeks, and each activity time window is represented by a starting and an ending week. To move the materials, different capacitated vehicles can be used. The aim of the optimization is to decide when and how moving the materials, by responding to digging and filling requests and minimizing costs of transportation and costs of material bought from quarries and brought to dump sites.

The optimization approach we used is a network flow optimization. The highway building site has been modeled as a graph, in which vertices correspond to locations, quarries, etc.. The arcs connecting the vertices are assigned a maximum flow capacity, that models the total amount of material that can be carried over the arc and that can vary during the time. Some public roads external to the building site have been included into the formulation, to take into account different paths that can be used to reach dump sites and quarries.

The approach contemplates an aggregate formulation and a disaggregate one. The aggregate formulation is a linear programming model, that considers the cumulate flows on the arcs of the network with the main objective of providing feasible solutions. It can be run many times by making use of a rolling horizon perspective in order to take into consideration the evolution of the activity. The disaggregate model is also a linear programming model. It takes in input the solution of the aggregate formulation and finds the best way to divide the cumulate flows of each material in each period.

Despite the use of linear programming, the models may need large computing times, because they typically involve hundreds of thousands of variables and constraints. In the talk, we will describe the problem and the modeling activity in details. We will give some hints on the behavior of the models on the real-world case study, and we will then present detailed computational results on an ad-hoc example that we created.

Hub Location Problem under Competition: (r/X_p) hub-medianoid and (r/p) hub-centroid problems

A. İrfan Mahmutoğulları, Bahar Y. Kara

Bilkent University Department of Industrial Engineering, 06800 Ankara, Turkey

{a.mahmutogullari,bkara}@bilkent.edu.tr

Introduction & Related Literature

Hubs are consolidation and dissemination points in many-to-many flow network systems. Consolidation generates economies of scale and thus unit transportation cost is reduced between hubs. Hubbing also reduces the number of required links to ensure that each flow is routed to its desired destination.

Today, many industries are ruled by a few numbers of competing firms. Such a market is called as an oligopoly (from Greek words *oligoi*: few and *polein*: to sell). Hence, market share and profit of a firm is affected by the decision made by itself and other competitors in the market. Also, customer behavior is another concern in oligopolistic markets. Market share is affected by the criteria that customers prefer one firm among others. Competition in oligopolies has been studied by economists to observe optimal decisions (including location) of each competing firm. However, studies considering competition are rare in the hub location literature.

Hotelling (1929) presents first competitive model that includes location decisions. He considers the location and price decisions of two firms operating on a line segment. Each customer prefers the firm that offers lower cost. Cost function includes the price of the product and a linear transportation cost. The demand is assumed to be uniformly distributed on a line segment. In Hotelling's model decisions are made simultaneously, that is no decision-maker have chance to observe the decision of the other party. Hotelling's pioneering work attracted many researchers who later consider the same model with more than two competitors, different customer behaviors and investigate equilibrium points on different spaces.

Another streamline research in competitive models deals with not simultaneous but sequential decision making process. The preliminary work of sequential decision making of location is first proposed by Stackelberg (1943). Stackelberg considers a duopoly where the firm that makes the initial decision is called the *leader* and the other one as the *follower*. His model has three major assumptions: decisions are made once and for all, decisions are made sequentially and both decision-makers have full and complete knowledge about the system. These leader-follower situations can be modeled as bilevel optimization problems. Bilevel optimization models consider the follower's reaction function as an input to the leader's decisions.

Drezner (1982) and Hakimi (1983), first OR specialists considering competition, independently propose sequential location problems and attracted the community's attention. They both consider the same competitive environment in different solution spaces. They define sequential problems and provide theoretical results for the properties of optimum solutions without proposing any linear model.

Competitive hub location studies in the literature are rare. The first hub location problem with competition is proposed by Marianov et al. (1999). They propose mathematical models for the follower's problem where the leader has already been operating the market with existing hubs. Later, Eiselt and Marianov (2009) propose another problem for an airline transportation company who enters a market.

Sasaki and Fukushima (2001) propose a competitive hub location model where the decision space is a plane. In their study, the problem is examined by both the leader's and follower's perspectives. Sasaki (2005) applies the same idea to a discrete environment with some modifications where the leader and the follower locate p and q hubs on the network, respectively. In another study by Sasaki et al. (2009), the decision-makers do not locate hubs but they locate hub arcs. The leader and follower airline companies locate q^a and q^b hub arcs on the network to maximize the total revenue.

Combining retail location from marketing, spatial competition in economics and location theory in operations research, in this study, we propose a discrete Stackelberg hub location problem where two firms make decisions sequentially. Each decision-maker (or firm) decides the location of hubs and allocation of non-hub nodes to the hubs considering profit maximization. We introduce (r/Xp) *hub-medianoid* and (r/p) *hub-centroid* problems to the literature, provide linear models for these problems and conduct computational experiments to observe efficiencies of the models.

Problem Definitions

Given a network $G=(N,E)$ where N is the set of nodes and E is the set of edges, let w_{ij} be the flow between nodes i and j for all $i,j \in N$ and c_{ij} be the transportation cost of a unit flow from node i to node j for all $i,j \in N$. The interhub transportation cost is discounted by a factor α , $0 < \alpha < 1$. The leader and follower want to enter a market with prespecified number of hubs. Say p and r be the number hubs to be opened by the leader and follower, respectively. The customers prefer the leader or follower with respect to provided service levels. Service level is defined as the cost of routing the flow from a node to its destination via hubs. A customer prefers the follower if the service level provided by the follower is strictly less than the one provided by the leader, otherwise the demand is captured by the leader. Ties are broken in favor of the leader in case of equal service levels. Since the customer has already operating with the leader when the follower enters the market, the customer has no incentive to deviate from the current position in case of a tie.

Let $X_p = \{x_1, x_2, \dots, x_p\}$, $X_p \subseteq N$ be the set of leader's hubs. The flow originated from node i visits one or two hubs before arrival to its destination node j . Therefore, we can easily compute the service level, say β_{ij} , provided by the leader for the flow between nodes i and j .

$$\beta_{ij} = \min_{k,m \in X_p} \{c_{ik} + \alpha c_{km} + c_{mj}\} \quad (1)$$

On the other hand, the follower enters the market by opening hubs on set of nodes $Y_r = \{y_1, y_2, \dots, y_r\}$, $Y_r \subseteq N$. Similarly, follower's service levels, say γ_{ij} , for all node pairs i and j can be calculated as:

$$\gamma_{ij} = \min_{k,m \in Y_r} \{c_{ik} + \alpha c_{km} + c_{mj}\} \quad (2)$$

The flow w_{ij} is captured by the follower if $\gamma_{ij} < \beta_{ij}$. Given that the leader and follower's hubs are located on the subset of nodes X_p and Y_r , respectively, the total flow captured by the follower can be expressed by a function $f: P_p(N) \times P_r(N) \rightarrow [0, W]$ such that

$$f(X_p, Y_r) = \sum_{i,j \in N: \gamma_{ij} < \beta_{ij}} w_{ij} \quad (3)$$

where $P_p(N)$ is the collection of subsets of N whose cardinality is equal to p and W is the total flow over the network, that is $W = \sum_{i,j \in N} w_{ij}$.

Given X_p , the follower wants to find the set Y_r that maximizes $f(X_p, Y_r)$ assuming the follower will respond (or act) rationally.

We define set Y_r^* as (r/X_p) hub-medianoid if $f(X_p, Y_r^*) \geq f(X_p, Y_r)$, $\forall Y_r \in P_r(N)$. In plain words, (r/X_p) hub-medianoid is the subset of nodes with r elements to locate hubs that maximizes the demand captured by the follower given the hub set of the leader.

The leader wants to minimize the demand captured by the follower (or equivalently maximize demand captured by himself/herself) while deciding his/her hub set. The leader also has the information that the follower will respond rationally.

We define set X_p^* as (r/p) hub-centroid if $f(X_p^*, Y_r^*(X_p^*)) \leq f(X_p, Y_r^*(X_p))$, $\forall X_p \in P_p(N)$ where $Y_r^*(X_p)$ is the (r/X_p) hub-medianoid given X_p . To simplify, we can say that (r/p) hub-centroid is the best choice of the leader's hub locations so that in the remaining scenario the follower can capture the least possible demand.

Model Development & Computational Results

In (r/X_p) hub-medianoid problem, the follower captures flow w_{ij} if he/she can provide a strictly better service level than the leader for node pair i and j . Since the set of hubs located by the leader is observable by the follower, β_{ij} values are known parameters for (r/X_p) hub-medianoid problem. Hence, we

model (r/X_p) hub-medianoid problem as a multi-allocation maximal hub cover problem where the cover radius is defined as $\beta_{ij}\varepsilon$ for every pair of nodes i and j where ε is a very small positive value that is used to break ties in favor of the leader. Our (r/X_p) hub-medianoid model requires $3n^3+2n^2+1$ linear constraints and $2n^3+n$ binary decision variables.

The (r/p) hub-centroid problem is more challenging to model and solve optimally because the leader should observe the rational reaction of the follower in the remaining scenario after deciding hub set X_p . In other words, the leader should locate his/her hubs such that in the remaining scenario the follower's highest possible market capture is minimized. Then, we propose a *center- (minimax-) type* objective model for (r/p) hub-centroid problem. Our (r/p) hub-centroid model requires $2C(|N|, r) + n^3 + n^2 + 1$ linear constraints and $C(|N|, r) + n^4 + n$ binary decision variables where $C(|N|, r)$ is the number of all r -combinations of set N .

We conduct our preliminary computational studies on CAB data set proposed by O'Kelly (1986) to observe the optimal set of locations for both leader and follower. A sample of the computational studies for (r/X_p) hub-medianoid problem on CAB data set with $p, r \in \{2, 3, 4, 5\}$ and $\alpha = 0.6$ is presented in Table 1. In the table, percentages of captured flow by the follower are presented given that the leader has already located his/her hubs on optimal hub-median or hub-center solutions. For example, given that the leader located on the set of nodes $X_p = \{12, 20\}$ (2-hub median), the best response of the follower is to choose $Y_r = \{2, 6\}$ for $r = 2$. In this scenario, the follower captures 65.62% of total flow on the network. The 2-center solution $X_p = \{8, 21\}$ is even a worse choice for the leader since the follower can capture 75.86% by responding $Y_r = \{5, 19\}$. All the instances presented in Table 1 are solved optimally within minutes.

Table 1: The percentages of captured flow by the follower in response to given the hub sets of the leader

Follower	Leader								
	2-hub median	2-hub center	3-hub median	3-hub center	4-hub median	4-hub center	5-hub median	5-hub center	
r = 2	65.62%	75.86%	30.49%	51.82%	18.89%	37.63%	18.64%	36.69%	
	78.25%	85.20%	45.13%	70.26%	28.39%	47.39%	28.14%	47.72%	
	87.08%	90.98%	53.69%	79.08%	37.73%	57.38%	35.04%	58.56%	
	92.26%	94.74%	62.03%	85.23%	46.19%	66.94%	42.32%	68.99%	

The (r/p) hub-centroid problem is computationally more challenging due to large number of constraints and variables. The computation experiments for (r/p) hub-centroid problem are still in progress; however, some preliminary results reveal that the leader can make a superior decision considering the reaction of the follower. To illustrate, for $p = r = 2$, the 2-hub centroid is $X_p = \{4, 17\}$, the corresponding 2-hub medianoid is $Y_r = \{11, 25\}$ and the follower captures only 46.14% of the total flow.

In this study, we introduce two competitive hub location problems, (r/X_p) *hub-medianoid* and (r/p) *hub-centroid problem*, for the hub location decisions in a competitive environment. Due to competition, a decision-maker should consider the decisions of rivals while deciding the set of hubs. We propose linear mathematical models for both problems. According to our preliminary analysis, we observe that the locations so the market shares change substantially under the correct settings. The mathematical models and more detailed analysis which also includes TR data set (includes 81-nodes Turkish cities with 16 potential hub locations, proposed by Tan and Kara (2006)) will also be discussed.

References

- Drezner, Z. (1982). Competitive location strategies for two facilities. *Regional Science and Urban Economics*, 12(4), 485-493.
- Eiselt, H. A., & Marianov, V. (2009). A conditional p-hub location problem with attraction functions. *Computers & Operations Research*, 36(12), 3128-3135.
- Hakimi, S. L. (1983). On locating new facilities in a competitive environment. *European Journal of Operational Research*, 12(1), 29-35.
- Hotelling, H. (1929). Stability in competition. *The economic journal*, 39(153), 41-57.
- Marianov, V., Serra, D., & ReVelle, C. (1999). Location of hubs in a competitive environment. *European Journal of Operational Research*, 114(2), 363-371.
- O'Kelly, M. E. (1986). Activity levels at hub facilities in interacting networks. *Geographical Analysis*, 18(4), 343-356.
- Sasaki, M. (2005). Hub network design model in a competitive environment with flow threshold. *Journal of the Operations Research Society of Japan-Keiei Kagaku*, 48(2), 158-171.
- Sasaki, M., & Fukushima, M. (2001). Stackelberg hub location problem. *Journal of the Operations Research Society of Japan*, 44(4), 390-405.
- Sasaki, M., Campbell, J. F., Ernst, A. T., & Krishnamoorthy, M. (2009). *Hub arc location with competition*. Technical Report of the Nanzan Academic Society-Information Sciences and Engineering (NANZAN-TR-2009-02).
- von Stackelberg, H. (1951). *Grundlagen der theoretischen Volkswirtschaftslehre*. Mohr Siebeck.
- Tan, P. Z., & Kara, B. Y. (2006). A hub covering model for cargo delivery systems. *Networks*, 49(1), 28-39.

Lagrangian Decompositions of the Two-Level FTTx Network Design Problem (Extended Abstract)

Andreas Bley *† Ivana Ljubić ‡§ Olaf Maurer*†

November 30, 2012

In passive optical networks, there are no electronically active components beneath a level of the network called the central office (CO) level. All bandwidth requirements in the network are served by fibers which go all the way from the central offices to the optical network units (ONUs). To serve several ONUs from a single fiber, there are intermediate points called distribution points (DPs) where optical splitters with a fixed splitting ratio can be installed.

We introduce a new network design problem called the *Two-Level FTTx Network Design Problem* (2FTTx). It captures some important optimization aspects for this kind of network. We simplify the problem by not considering different cable types with various costs and capacities; instead, we assume that a fixed price has to be paid for each single fiber installed on an edge as well as for using the edge.

1 Problem Definition

We are given an undirected graph $G = (V, E)$ with the set of nodes V partitioned into: customers (V_C), potential distribution points (V_D), potential central offices (V_{CO}) and the remaining nodes (V_O). At least one central office needs to be opened and each customer $v \in V_C$ needs to be provided with at least $d_v \geq 0$ fibers. Thereby, fiber connections run from a CO, following a path of nodes in V_O or non-opened DPs or COs, through exactly one DP, until they reach an end customer. Splitters are installed at DPs. T denotes all available splitter types, s_t is the splitter ratio for each $t \in T$ and $J_{t,v}$ is the maximal number of the splitter-type t that can be installed simultaneously at the DP v . A subnetwork

*Department of Mathematics, Berlin Institute of Technology, Straße des 17. Juni 136, 10623 Berlin, Germany. Email: {bley,maurer}@math.tu-berlin.de.

†Supported by the DFG Research center MATHEON *Mathematics for key technologies* in Berlin.

‡Supported by the APART Fellowship of the Austrian Academy of Sciences.

§Department of Statistics and Operations Research, University of Vienna, Brünnerstr. 72, 1210 Vienna, Austria. Email: ivana.ljubic@univie.ac.at

containing the routing paths between COs and DPs is called the *feeder network* (FN), and a subnetwork containing the routing paths between DPs and end customers is called the *distribution network* (DN). Along each edge $e \in E$ at most u_e fibers can be installed in the fiber and the distribution network, independently. Finally, u_v is the capacity of a DP or CO v , i.e., the maximal number of the downstream-fibers or transceivers, respectively. In total, DPs and COs are allowed to be installed in at most N_D and N_{CO} locations, respectively. The following costs are associated to the input parameters of our network:

$$\begin{aligned}
c_v & \quad \forall v \in V_D \cup V_{CO} & : & \text{opening costs for } v \text{ being a DP or CO} \\
c_{t,v} & \quad \forall t \in T, v \in V_D & : & \text{the cost for the installation of a splitter of type } t \text{ at the DP } v \\
c_e & \quad \forall e \in E & : & \text{the installation cost for an edge } e \\
c_e^f & \quad \forall e \in E & : & \text{the cable cost along edge } e \text{ in the FN} \\
c_e^d & \quad \forall e \in E & : & \text{the cable cost along edge } e \text{ in the DN}
\end{aligned}$$

The optimization task consists of deciding which COs and which DPs to open, which splitters to install at the DPs, and how to route paths in the subnetwork so that demands of all customers are satisfied at minimum cost.

2 MIP Model

We use binary variables $x_v \in \{0, 1\}$ for each $v \in V_D$ to indicate whether v is chosen as a DP or not. Similarly, binary variables $z_v \in \{0, 1\}$ indicate whether a central office $v \in V_{CO}$ is opened or not. The number of splitters of type t installed at the DP location $v \in V_D$ is modeled by integer variables $y_{t,v}$. Finally, for each edge $e \in E$, binary variable w_e indicates whether the edge e is used or not, and the number of fibers in the DN and the FN installed along arc $a \in A$ is modeled using integer variables g_a and f_a , respectively. Using these variables, our objective function can be described as follows:

Objective function

$$\min \sum_{v \in V_{CO}} c_v z_v + \sum_{v \in V_D} c_v x_v + \sum_{e \in E} c_e w_e + \sum_{v \in V_D} \sum_{t \in T} c_{t,v} y_{t,v} + \sum_{a \in A} (c_a^f f_a + c_a^g g_a)$$

The first two terms are the installation costs for COs and DPs, followed by the installation costs for the edges, followed by the splitter installation costs. The last summation term corresponds to the total fiber costs installed in the DN and the FN.

Bounds on splitter installation, DPs and COs. The bounds regarding the total number of allowed DPs and COs are expressed using constraints (1) and (2), respectively, and the bounds regarding the maximal number of splitters

of type t at the DP node v are expressed using (3):

$$\sum_{v \in V_D} x_v \leq N_D \quad (1)$$

$$\sum_{v \in V_{CO}} z_v \leq N_{CO} \quad (2)$$

$$y_{t,v} \leq N_{t,v} x_v \quad v \in V_D, t \in T \quad (3)$$

Before we present the remaining constraints of our MIP model, we introduce the following set of auxiliary variables that do not contribute to the objective function value, but will simplify the further notation:

$$\begin{aligned} F_v &\in \mathbb{Q} & \forall v \in V_D & : \text{the number of upstream-fibers at the DP } v \\ G_v &\in \mathbb{Q} & \forall v \in V_D & : \text{the number of downstream-fibers at the DP } v \\ H_v &\in \mathbb{Q} & \forall v \in V_{CO} & : \text{the number of transceivers installed at the CO } v \\ w_a^f &\in \{0, 1\} & \forall a \in A & : 1 \text{ if the arc } a \text{ is used by the } f\text{-flow} \\ w_a^g &\in \{0, 1\} & \forall a \in A & : 1 \text{ if the arc } a \text{ is used by the } g\text{-flow} \end{aligned}$$

The integrality of the auxiliary variables F_v, G_v, H_v is implied as the f_a and g_a variables are integer.

Flow conservation in distribution and feeder network. In order to ensure a feasible routing in the DN, the following flow-preservation constraints (4) are used. They also state that the total customer demand has to be satisfied using the supply of the downstream-fibers that are available at distribution points. Constraints (5) and (6) make sure that the g -flow is routed along e only if the edge e is installed.

$$\sum_{a \in \delta^-(v)} g_a - \sum_{a \in \delta^+(v)} g_a = \begin{cases} d_v & v \in V_C \\ -G_v & v \in V_D \\ 0 & \text{else} \end{cases} \quad v \in V \quad (4)$$

$$g_{ij} \leq u_e w_{ij}^g \quad \{i, j\} \in E \quad (5)$$

$$w_{ij}^g + w_{ji}^g \leq w_e \quad \{i, j\} \in E \quad (6)$$

Similarly, to make sure that the routing between COs and DPs is feasible, we consider constraints (7). These constraints also make sure that the overall upstream-demand at distribution points (expressed using F_v variables) is to be satisfied by installing a sufficient number of transceivers at the corresponding COs (H_v variables). Constraints (8) and (9) make sure that the f -flow is routed along e only if the edge e is installed.

$$\sum_{a \in \delta^-(v)} f_a - \sum_{a \in \delta^+(v)} f_a = \begin{cases} F_v & v \in V_D \\ -H_v & v \in V_{CO} \\ 0 & \text{else} \end{cases} \quad v \in V \quad (7)$$

$$f_{ij} \leq u_e w_{ij}^f \quad \{i, j\} \in E \quad (8)$$

$$w_{ij}^f + w_{ji}^f \leq w_e \quad \{i, j\} \in E \quad (9)$$

Upstream-demand and downstream-capacity at DPs. For each installed splitter at a DP $v \in V_D$, a single fiber in the FN is required, and the total upstream-demand at v is calculated using constraints (10). The number of downstream-fibers available at the DP v is bounded by the total number of installed splitters and their capacity, see (11):

$$F_v = \sum_{t \in T} y_{t,v} \quad v \in V_D \quad (10)$$

$$G_v \leq \sum_{t \in T} s_t y_{t,v} \quad v \in V_D \quad (11)$$

Finally, the number of available downstream fibers at DPs and COs v is also bounded by u_v , i.e.:

$$G_v \leq u_v x_v \quad v \in V_D \quad (12)$$

$$H_v \leq u_v z_v \quad v \in V_{CO} \quad (13)$$

Finally, if FN and DN are required to have tree topologies, this can be guaranteed using the following in-degree constraints:

$$\sum_{a \in \delta^-(i)} w_a^f \leq 1 \text{ and } \sum_{a \in \delta^-(i)} w_a^g \leq 1, \quad i \in V \quad (14)$$

The set of feasible 2FTTx solutions is completely described using constraints (1) – (14). This MIP model will be called the *aggregated MIP model*. It contains a large number of variables and constraints and it is quite unrealistic that using this model one will be able to solve instances of practically relevant sizes. Also, the lower bounds obtained by this model are quite weak due to the “bigM”-constraints (5),(8),(12),(13).

3 Solution approaches

We propose two Lagrangian decomposition approaches we developed to deal with the size and complexity of the problem. To overcome the difficulties with the lower bounds, we present families of strengthening valid inequalities. A combination of these allows us to solve some of the real-world network planning instances to provable optimality or find very good solutions and lower bounds quickly.

Local Search for the Reload Cost Spanning Tree Problem

S. Raghavan⁽¹⁾ and Mustafa Sahin⁽¹⁾

⁽¹⁾ Smith School of Business & Institute for Systems Research

University of Maryland, College Park, MD 20742, USA

(email: raghavan@rhsmith.umd.edu, mustafa.sahin@rhsmith.umd.edu)

When the mode of transportation is changed in a network, it usually comes with a "reload cost". Even though reload costs have not been studied extensively in the literature, there are real life applications such as communication networks with diverse technologies, transshipment of goods from trains to ships or many other settings where this cost is incurred naturally. There are several network design problems involving reload costs, in this talk, we discuss the reload cost spanning tree problem (RCSTP). In the RCSTP, we try to find the spanning tree with the minimum total reload cost where each node has nonnegative demands that need to be satisfied by specific nodes, each edge is colored and a reload cost is incurred when a color change occurs. The RCSTP was introduced and shown to be NP-complete by Gamvros et al (2012), so unless P=NP, we have no hope of solving RCSTP optimally in polynomial time. Gamvros et al (2012) propose a mixed-integer programming formulation for the problem and show in their computational experiments that the problem becomes intractable (for their model) as the number of nodes, edges and colors in the network increases. We propose local search heuristics for the RCSTP. Our local search neighborhood consists of spanning trees that differ from the current solution by exactly one edge. We discuss how to efficiently search through this neighborhood structure. We also show how to efficiently update savings when the solution is updated. Finally, we present computational results comparing the heuristic solutions with optimal solutions. We also examine the performance of the heuristic on large-scale problems.

A three step decomposition approach for a transportation network design problem with non-linear costs

Lehuédé Fabien¹ Péton Olivier¹

*LUNAM Université, Ecole des Mines de Nantes, IRCCyN UMR CNRS 6597
(Institut de Recherche en Communications et Cybernétique de Nantes),
4 rue Alfred Kastler, 44300 Nantes, France*

Keywords: Distribution network design, decomposition, horizontal cooperation

1 Motivation of the study

We propose a methodology and simple solution approach to evaluate a collaborative distribution system. We consider a cluster of small and medium sized companies who rely on several logistic service providers to regularly ship small product quantities to a set of common customers spread over a large territory.

The main contribution of this work is to model and solve one practical transportation planning problem that arises in this horizontal collaboration. We integrate realistic costs and constraints. A practical and efficient decomposition scheme is proposed to solve this case. A similar problem has been studied in [1] with slightly different cost functions and hypothesis.

2 Benefits and costs of the collaborative distribution network

A first reason why horizontal collaboration is expected to generate great benefits is shipments consolidation. As shown in Figure 1, the carrier cost for

¹ Email:{fabien.lehuede;olivier.peton}@mines-nantes.fr

one Transportation Unit (TU) decreases when the number of TUs delivered increases. Hence, consolidating the orders of one customer to several shippers decreases the unitary distribution cost.

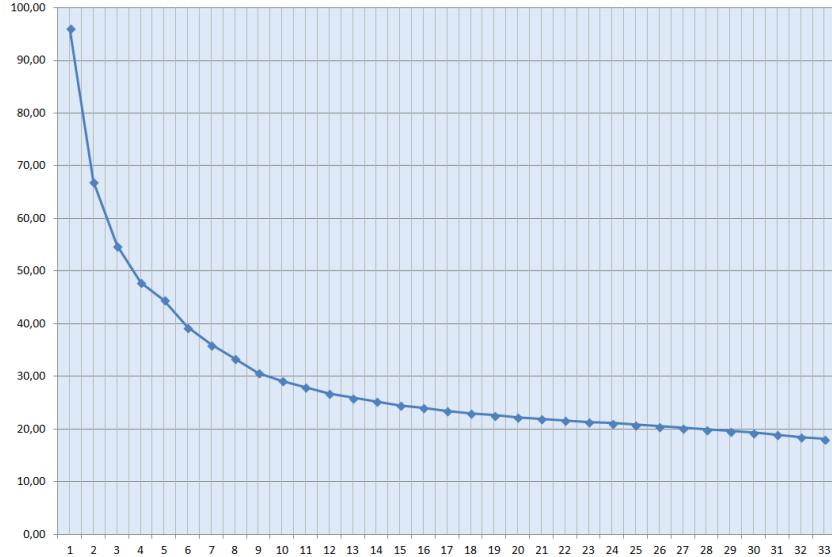


Fig. 1. Distribution price of one TU for a given pair (origin, destination) with respect to the number of TU to be distributed

A second reason is that the volume increase obtained through collaboration allows to perform Full TruckLoad (FTL) transportation from the companies to a set set of regional Distribution Centers (DC). As FTL rates are considerably lower on long distances, this can reduce the distribution costs even for orders to a single company.

Nevertheless, this collaborative system implies physical consolidation of products being delivered to a common customer, either to consolidate a shipment or to constitute full truckloads. This requires locating a shared new facility called Consolidation and Distribution Center (CDC) and defining routes from the companies to the CDC or to regional CDs. This part of the collaboration process generates two types of extra activities: pickup routes from the companies to the CDC and cross-docking operations at the CDC and at the DCs. Pickup routes are performed by the companies fleet of vehicles.

3 Modeling and Computational experiments

In order to evaluate the benefits of this collaborative model, we assessed the collaborative distribution system by simulating one complete year of distri-

bution orders. We compared three scenarios: separate deliveries (current system), collaborative deliveries based on a local CDC, and collaborative deliveries through a CDC and a network of several regional DCs. In the last scenario, simulating one day of distribution implies solving a combinatorial optimization problem that determines the number of FTL routes, designs the routes between the companies and the CDC, and assigns customer orders to vehicles.

This optimization problem is decomposed into three Mixed Integer Programs (MIPs) that are solved sequentially to obtain an approximate solution. The first MIP determines for each order if it is distributed from a provider, the CDC or a DC, as well as the number of FTL routes that serve the DCs. For orders distributed through DCs, the second MIP determines the pickup part and the load of each FTL route. Finally the last MIP determines the routes of the proprietary vehicles that gather the orders for consolidation at the CDC.

Our experiments show that in our case study, a significant benefit can be achieved through order consolidation and regional DCs.

References

- [1] Kathleen a. Lindsey, Alan L. Erera, and Martin W. P. Savelsbergh. A pickup and delivery problem using crossdocks and truckload lane rates. *EURO Journal on Transportation and Logistics*, September 2012.

Author:

Andreas Bärmann, FAU Erlangen-Nürnberg, Andreas.Baermann@math.uni-erlangen.de

Coauthors:

Frauke Liers, FAU Erlangen-Nürnberg, Frauke.Liers@math.uni-erlangen.de

Alexander Martin, FAU Erlangen-Nürnberg, Alexander.Martin@math.uni-erlangen.de

Maximilian Merkert, FAU Erlangen-Nürnberg, Maximilian.Merkert@math.uni-erlangen.de

Christoph Thurner, FAU Erlangen-Nürnberg, Christoph.Thurner@math.uni-erlangen.de

Dieter Weninger, FAU Erlangen-Nürnberg, Dieter.Weninger@math.uni-erlangen.de

Solving Network Design Problems via Iterative Graph Aggregation

In this talk, we present a new exact approach for solving a special class of network design problems (NDPs) that is based on iterative graph aggregation. In its classical version, the network design problem consists of determining a cost-optimal dimensioning of the arc capacities such that a specified demand pattern can be routed through the network. Motivated by some real applications, the instances considered here already contain preexisting capacities, which allow for a relatively high percentage of the demand to be routed. Additional capacities can be installed in integral multiples of a specified capacity value and for a price that can vary for each arc. The objective is a least-cost installation of new capacities such that the complete demand pattern can be routed, which is still far from trivial to achieve in practice. The above problem is motivated by infrastructure expansion problems where some network is already in place, as it is the case for European railway transport networks.

We propose a new method for solving the indicated expansion problem that exploits the information obtained by a successive solution of aggregated versions of the original problem instances. Starting with an initial aggregation of the network, we solve a sequence of network design master problems over increasingly fine-grained representations of the original network. In each step, a subproblem acts as a stopping criterion signalling optimality of the solution or giving a directive where to refine the representation of the network in the next iteration. The algorithm terminates with a globally optimal solution to the problem on the original instance. The capacitated network design problem is well-known to be NP-hard. It is very often solved via Lagrangian relaxation in order to cope with the difficulty of its LP-relaxation in the flow formulation. Our approach tries to overcome this difficulty by solving a sequence of much smaller aggregations of the originally large instance, constantly improving the accuracy by refining the representation. The method stops when a global optimal solution is found. The approach is motivated by the success of shortest path algorithms based on graph contractions. They allow for the solution of the problem on an equivalent but considerably smaller graph. Especially when employing the flow formulation for the solution of the network design problem, our method aims at exploiting the significantly smaller LP-relaxations of the aggregated instances. We present a statement of our algorithm for the single-commodity version of the network design problem and outline its extension to the multi-commodity case.

We show computational results for our method for a variety of test instances. It turns out that in many cases the original problem instance can be solved to optimality without the need to consider a full representation of the original network in the network design master problem.

On Debris Removal During the Response Phase

Halenur Şahin, Oya Ekin Karaşan, Bahar Yetiş Kara

Bilkent University, Industrial Engineering Department, Ankara/Turkey

halenur.sahin@bilkent.edu.tr, karasan@bilkent.edu.tr, bkara@bilkent.edu.tr

Turkey is an earthquake-prone country where there exist many small, medium and large scale earthquakes in the history. As a result of destruction of structures during an earthquake, debris, which obstructs to transmit emergency relief supplies to the disaster victims, occurs.

Debris management is part of disaster management cycle which consists of four phases: preparation, response, recovery and reconstruction.

In the preparation phase, it is required to take precautions beforehand in order to minimize the possible negative effects of the disaster whereas the response phase starts immediately after the disaster and involves transmitting all kinds of emergency services to the maximum possible number of disaster victims as soon as possible. During the recovery phase, it is aimed to recover the disaster affected region in terms of communication, transportation and infrastructure; finally, in the reconstruction phase, the main objective is to fully rehabilitate the disaster affected region and normalize disaster victims' daily lives.

Debris removal operations can be classified as follows:

- Debris removal for unblocking roads in order to transmit emergency relief supplies and aid teams to the disaster victims in the phase of response.
- Debris removal in order to rescue disaster victims from the wreckage.
- Complete debris removal in the phase of recovery.

Carbajal et al. [1] introduced the debris management problem to the operations research literature. Literature review studies show that the main studies about debris are performed by Federal Emergency Management Agency (FEMA) [2]. By considering the structure of United States, studies of FEMA focus on debris management, debris estimation and debris removal issues. Additionally, such studies define debris management processes in detail. However the analysis mainly focuses on debris management in the recovery phase. Moreover, the existing debris management studies developed are specific to hurricane, where the time and the location are known apriori. Following an earthquake, providing emergency supplies and search and rescue operations that occur in the response phase have also to be considered.

Our focus in this study is on debris management operations during the response phase.

To the best of our knowledge, debris management in the phase of response is never studied in the literature.

We developed a debris management model to transmit emergency services to the critical disaster affected districts as quickly as possible. In this context, characteristics of critical districts are also taken into consideration. In particular, highly populated districts, districts that have shelters, hospitals and schools are prioritized.

In the disaster affected region, some of the roads may be suitable for traversing, whereas some of them are blocked by debris. In order to provide disaster aid to the critical districts, it is necessary to travel on a path which may include blocked roads as well. In such a case, it is required to unblock these roads by debris removing operations.

To this end, we define "*Debris Removal Problem in the Response Phase*" as, visiting critical disaster affected districts as quickly as possible, taking into account priority levels, traversing (if necessary) along blocked arcs by carrying out unblocking operations.

Within the operations research literature our problem can be investigated as a variant of the general routing problems. The General Routing Problem (GRP) both includes arc and node routing aspects and it is the generalized version of the Arc Routing Problem (ARP) and the Node Routing Problem (NRP).

In GRP, there are a subset of edges and a subset of nodes which are required to be visited [3]. Some relevant problems under the umbrella of GRP are the Chinese Postman Problem (CPP), the Rural Postman Problem (RPP), the Capacitated Arc Routing Problem (CARP), which can be viewed as special cases of ARP, and the Vehicle Routing Problem (VRP), which is one of the most studied capacitated node routing problem [4], [5].

When the required node set is empty and it is aimed to visit all edges, the GRP reduces to CPP; if there is a subset of edges that need to be visited with an empty required node set, then GRP reduces to the RPP [3], [6]. On the other hand when the required edge set is empty and it is aimed to visit all nodes, the GRP reduces to the Vehicle Routing Problem (VRP) [5], [7].

Since our problem aims to visit critical nodes by considering some precedence relations, it shows similarities with VRP. However, there are blocked arcs on the network and debris removal on these arcs is also considered in our problem. In this respect, it also shows similarities with ARP. Accordingly, it can be said that, our problem both includes node routing and arc routing aspects, but it is a new variation, which has not received attention before in the operations research literature.

Disaster affected region is divided into districts. One of these districts is the supplier and some others are critical districts that require emergency supply. There are some districts which are neither critical nor supplier. Additionally, critical districts have different priority levels according to their characteristics, such as districts with hospitals, shelter areas etc. It is preferable to transmit emergency reliefs as early as possible to the high priority nodes.

Both debris removal and traversal of a road require some effort which is translated to time in our model. Time entails the capacity restriction of our model. There are predefined numbers of periods each consisting of predefined amount of time. This capacity restricts the number of traversals and debris removals within a period.

A vehicle departs from the supplier district and aims to transmit relief materials to the critical districts as soon as possible, by considering their priority levels and satisfying capacity restrictions.

We provide a linear mixed integer mathematical model for the "*Debris Removal Problem in the Response Phase*". The model has $O(n^3)$ binary variables where n is the number of districts.

We conduct our preliminary analysis on Kartal/İstanbul [8] data. There are 45 districts (nodes) with given pairwise distances. The time to traverse an edge is calculated by dividing the distance by the speed of the vehicle. The effort required to unblock edges is taken as proportional to the distances between districts. A set of edges is randomly determined as the initial set of unblocked edges. In the preliminary computation experimentation, the results of which are detailed below, it is assumed that there are 7 available periods and the critical nodes are identified as nodes 2 through 8. We test our model for different priorities and period capacities and list the CPUs, completion periods and visitation periods of critical nodes in the following table.

Table 1: Results of our preliminary computational analysis.

Instance No	Critical Nodes	Priority of Critical Nodes	Available #of periods ^[1]	Capacity Of Each Period(min) ^[2]	CPU(s)	Completion period ^[3]	Visitation Period of critical nodes ^[4]						
							Period 1	Period 2	Period 3	Period 4	Period5	Period 6	Period 7
1	2,3,4,5,6,7,8	1,20,1,50,90,120,300	7	100	361	3	#1: 5,6,8	#2: 3,7	#3: 2,4				
2	2,3,4,5,6,7,8	1,20,1,50,90,120,300	7	90	9	3	#1: 5,6,8	#2: 3,7	#3: 2,4				
3	2,3,4,5,6,7,8	1,20,1,50,90,120,300	7	80	80	4	#1: 5,6,8	#2: 2,7	#3: 3	#4: 4			
4	2,3,4,5,6,7,8	1,20,1,50,90,120,300	7	50	2336	6	#1: 8	#2: 6	#3: 7	#4: 3,5	#5: 2	#6: 4	
5	2,3,4,5,6,7,8	300,120,90,50,1,20,1	7	90	2831	3	#1: 2,4,5;	#2: 3,7	#3: 6,8				

^[1] indicates the available number of periods.

^[2] capacity of each period is considered to be identical and is defined in terms of minutes.

^[3] lists the completion period for all the critical nodes.

^[4] depicts the individual visitation periods of the critical nodes.

As indicated in the first row of the table, when the capacity of each period is 100 minutes, visitation of nodes completed in 3 periods. When the capacity is reduced to 90 minutes, 3 periods are still sufficient to complete the visits. However, as capacity reduces to 80 minutes, 3 periods are no longer sufficient and 4 periods are required. When the capacity is 50 minutes, 6 periods are needed. Additionally, when the priorities of the nodes are changed for a particular instance, the visitation periods of the nodes also change.

The following figures illustrate the travelling schema for instances 2, 4, and 5.

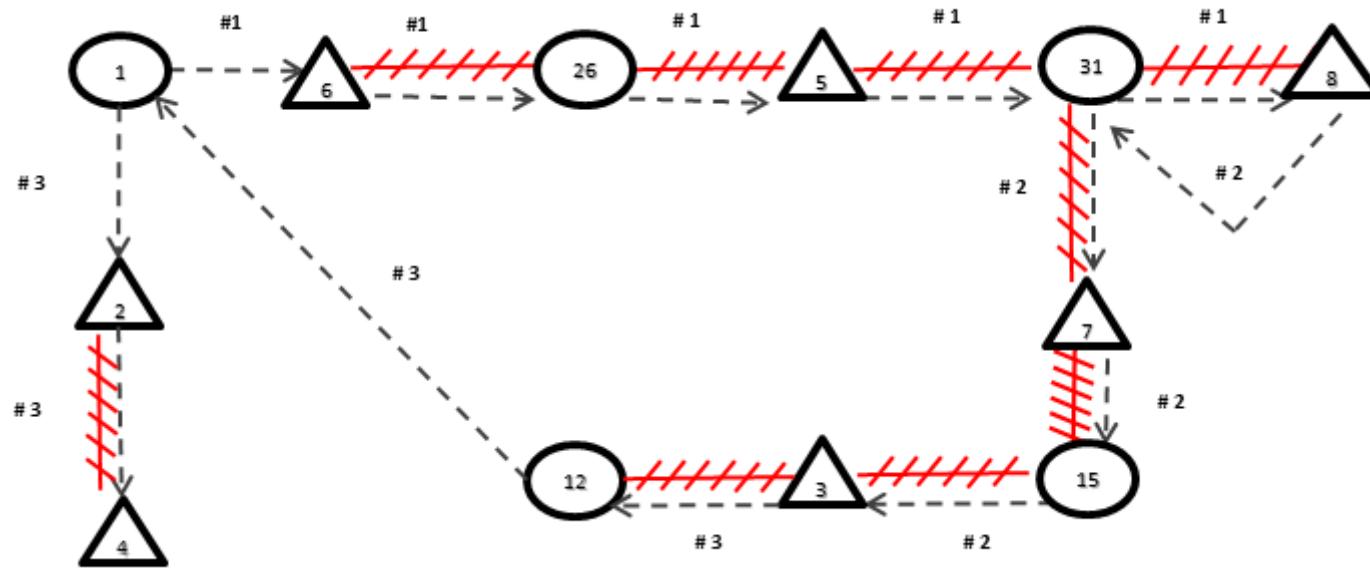
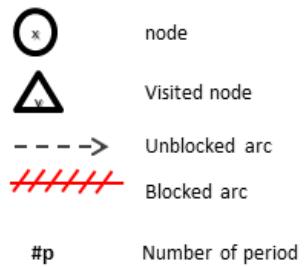


Figure1. (Instance 2)

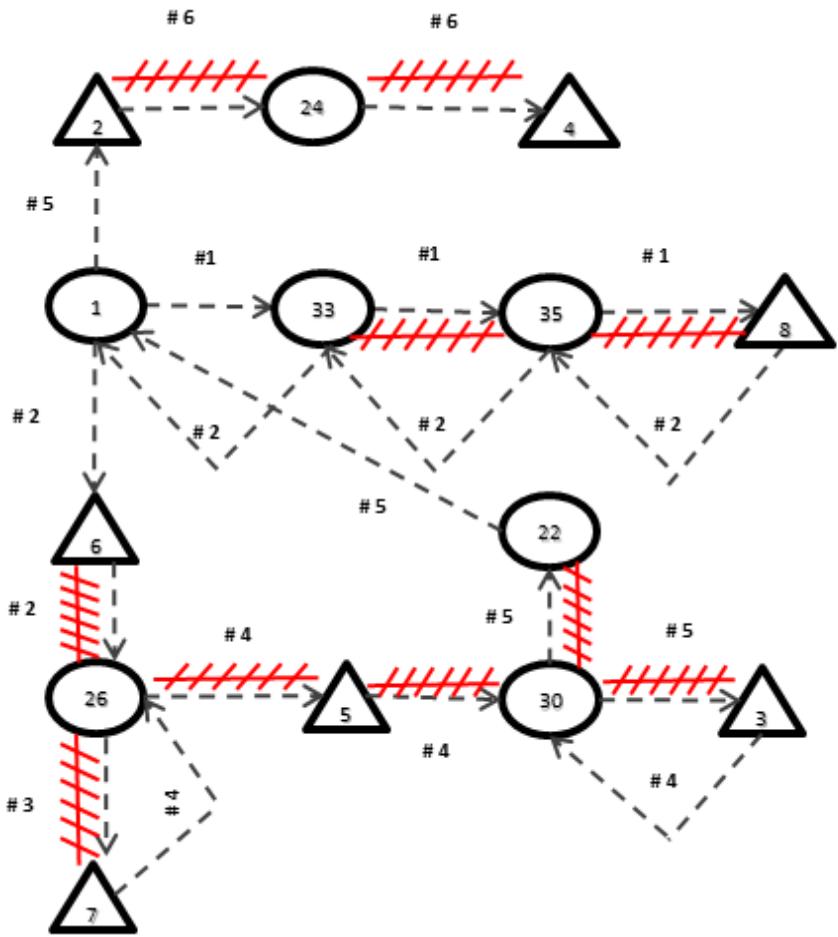


Figure2. (Instance 4)

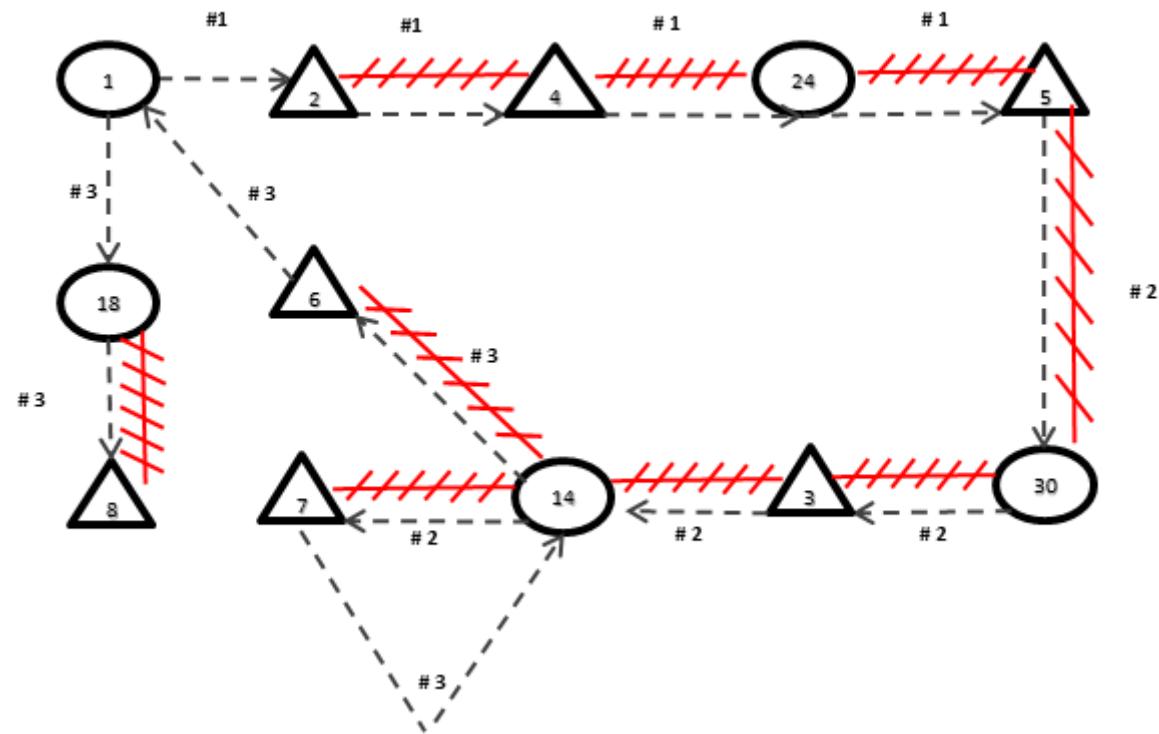


Figure3. (Instance5)

With this work, we introduce the debris removal problem in the response phase to the operations research literature. In the future, the model will be enhanced through the introduction of valid inequalities in order to reduce the CPU times and more computational experimentation will be conducted.

References

- [1] Carbajal J. A., Ergun O., Keskinocak P., Stilp K., Villareal M. "Debris Management". Managing Debris Operations. Georgia Tech. Supply Chain & Logistics Institute Center for Humanitarian Logistics2011 Health and Logistics Conference Poster Session. March 2-3, 2012.
<http://www.scl.gatech.edu/humlog2011/posters/#88>
- [2] FEMA: Debris Management Guide. U.S. Department of Homeland Security Federal Emergency Management Agency. <http://www.fema.gov/government/grant/pa/demagde.shtml>
- [3] Lenstra, J.K., A.H.G Rinnooy Kan. 1976. On General Routing Problems. Networks 6, 273-280.
- [4] Prins, C. 2004. [A simple and effective evolutionary algorithm for the vehicle routing problem.](#) Computers & Operations Research 31, 1985–2002.
- [5] Golden, B.L., Wong, R.T. 1981. Capacitated Arc Routing Problems. Networks 11, 305-315.
- [6] Corberán A., Sanchis J.M. 1994. A polyhedral approach to the rural postman problem. European Journal of Operational Research, 79(1): 95-114.
- [7] Laporte, G., 1992. The vehicle routing problem: an overview of exact and approximate algorithms. European Journal of Operational Research 59, 345-358.
- [8] Kilci, F., 2012. A decision support system for shelter site location with GIS integration: Case for Turkey. Technical Report, Turkey

Design of Fixed Charge Multicommodity Networks using k-Partition Facets

Yogesh K. Agarwal

Indian Institute of Management, Lucknow, INDIA; email: yka@iiml.ac.in

and

Yash P. Aneja

University of Windsor, Canada; email: aneja@uwindsor.ca

We are given an undirected graph $G=(V,E)$, where each edge in E has a fixed cost and a capacity associated with it. We need to select a subset of edges that will satisfy a given set of multi-commodity traffic demands between various node-pairs at minimum cost. This problem has many applications in infrastructure design, e.g. telecommunications, power distribution, highways and railroads etc. While the directed version of the problem has received some attention, there is little work on the undirected version, except for some heuristic approaches. Let $\{V_1, V_2, \dots, V_k\}$ be a k -partition of nodes in V . If the nodes in each subset V_i are shrunk, we obtain a k -node multi-graph, and a corresponding smaller version of the problem (called the k -node sub-problem) by aggregating the demands between each pair of subsets. We present a key theorem, accordingly to which, the facet inequalities derived from such a k -node sub-problem are also facet-defining for the original problem under certain conditions. For small values of k , e.g. $k = 2, 3$ or 4 , we propose an approach based on polarity theory to compute a violated facet. Given a current fractional solution \bar{x} , a facet violated by this solution can be found as follows. All integer solutions of the k -node sub-problem are generated. Each solution corresponds to a constraint of the polar polyhedron, whose extreme points correspond to the facets of the k -node sub-problem. By solving a linear programming problem over the polar polyhedron, the most violated facets of the subproblem is generated. By repeatedly adding such constraints to the master problem, we obtain a lower bound which is substantially higher than the LP bound. When the problem is solved by CPLEX branch and bound after adding the k -partition constraints, it leads to a substantial decrease in the solution time. Using this approach we report optimal solutions of problems with 25 nodes, 60 edges, and fully dense traffic matrices obtained within a few minutes of cpu time.

Fiber Optical Network Design Using Passive Splitters

Başak Yazar

Bahar Yetiş Kara

Oya Ekin Karaşan

Department of Industrial Engineering, Bilkent University, 06800 Bilkent, Ankara, Turkey

{basakyazar@gmail.com, bkara@bilkent.edu.tr, karasan@bilkent.edu.tr}

Technological developments enhance the ways of reaching information, expanding knowledge, and the accessing Internet from everywhere with a fast, secure and survival connection. Fiber optic usage is very common in telecommunication networks. Although types, band width, and telecommunication equipment show variety, a typical goal is generally to serve more customers with moderate budgets. By using fiber optical networks, customer can be served with ultra fast internet access and benefit from high broad band with (multi task) in download or upload operations.

We base our problem definition on an application arising from one of the largest internet service providers operating in Turkey. In this particular setting, there is a central station at a fixed location. This central station serves for both voice communication and interest access. Each customer in the field needs to be connected to this central station with fiber wires. The quality of the service is measured in terms of dBs. The company wants to offer services to all its customers guaranteeing a fixed level of dB service quality. Depending on the length of the fiber wire, dB decreases a fixed amount per km during traversal to the customer. Direct connections to each customer from the central station will be too costly and thus, certain splitters are used. The company prefers to use passive splitters. The splitters have many types mainly depending on the available number of ports. The utilization of passive splitters also results in extra loss of dB. For example in a passive splitter with 32 division when insertion loss is taken as 3 dB/km, then there are ($32 = 2^5$) five ports which results in (3x5) 15 dB decline at the output.

In the network design part of the fiber optic network design problem for this company, we decide the location of passive splitters while obeying the dB requirements. The passive splitter type selection is another challenging decision to be made involving the tradeoff between cost of splitter, splitting number and the insertion loss. More splitting increases reachability, but results in high insertion losses. In addition to these insertion losses, we aim to serve all customers in a cost effective way. Hence, the objective is to minimize both fiber optical wiring cost and passive splitter costs.

The telecommunication network design literature is motivated from different application areas, such as electrical and electronics engineering, computer science, and operations research. Wide area, metropolitan area, and local area are emphasized on different application areas. Our focus in this study

is on the local area network design. In the sense of operations research, a telecommunication network consists of a set of nodes and links (Klincewicz, 1998). These nodes are demand points that send/receive messages or information such as voice, data, and video. The design might involve a network design from scratch or expansion/improvement of the existing network such as capacity expansion.

There are different elements of a network such as cost, capacity, reliability, performance, and demand pattern (Klincewicz, 1998). The cost of a network includes establishing a link which may be linear or not, using a link which may depend on the usage volume, establishing a hub (apriori or not), using a hub (linear or nonlinear) as described in Gavish (1992). Usage costs are typically flow dependent. Capacities refer to both link and hub capacities. The performance depends on the amount of traffic arising in the network over time and whether or not the network can handle the increased amount of traffic with its existing hardware and software elements. Demand pattern may be many-to-many suggesting that any node can have a traffic with any other node in the network or many-to-one which is used when only one central location exists in the networks (Klincewicz, 1998).

In terms of the hub location perspective, a telecommunication network considers a “tributary” and a “backbone” network. Hubs are switching points of the telecommunication traffic. Tributary networks connect nodes to hubs and backbone networks interconnect the hubs. Depending on the problem definition, tributary networks can be “local” or “access” networks and backbone networks are “hub-level” networks. Hubs are telecommunication elements such as switches, gates, concentrators, control points, or access points (Klincewicz, 1998). In our problem definition, hubs are passive splitters and the supplier is the central station.

Existing in the literature are different types of studies that consider solely tributary or solely backbone networks as well as joint problems evaluating backbone and tributary networks together. Since the telecommunication problems are in general very challenging, typical research divides the core problem into subproblems and solve subsequently (Gavish, 1991). Even the subproblems are generally NP-hard (Gavish et al., 1992).

According to Ergün (2011), there are four main problem types in local access network design. The first is the concentrator location problem which decides on the number and location of concentrators and resembles the facility location problem. The second one is the terminal assignment problem which is about concentrator-terminal matching. Then, the terminal layout problem arises and the last one is the Telpak problem which is about the link capacities. The difference of backbone networks from local networks is the high transmission rate. The backbone networks can be fully-interconnected, ring or mesh in terms of their topology. Another problem type is that which involves the backbone and the access networks jointly and it involves two levels. This can also be referred to as hierarchical design as in the cases in Thomadsen (2005), Thomadsen and Stidsen (2007), Rosenberg (2005). According to Thomadsen (2005), two-level hierarchical network design includes hub location, clustering of nodes, interconnection of nodes in the backbone and cluster networks, and finally routing

between them. Klincewicz (1998) studies the two-level hierarchical network in a survey. There might be same or different topologies in each level, examples are star-star (Petrek and Siedt, 2004), ring-star (Labbe, 2004), ring-ring (Proestaki and Sinclair, 2007), fully interconnected- fully interconnected (Thomadsen and Larsen, 2007) or mesh-mesh (Thomadsen, 2005).

We provide a linear mixed integer mathematical model for our problem. The proposed model has $O(n^2)$ binary variables where n is the number of customer locations. It takes the location of the central station and the dB decay parameter value as input and outputs the configuration with minimum total cost. For preliminary analysis, we experimented our model on real data from the Kartal district of İstanbul (Kılçıcı, 2012). There are 45 nodes in this data set. These preliminary results are summarized in the table below. For exemplary purposes, we choose two insertion loss values listed as decline. K is the number of passive splitter types with an associated cost value. We range the cost values from low to high as shown in Table 1.

Table 1. Preliminary Results for Kartal - İstanbul

Instance #	Parameters		Results	
	K	Cost (k)	# of Passive Splitter Installed	CPU (sec)
1 Decline = 1 Low Cost	1	6.25	-	5.7
	2	7.5	-	
	3	10	1	
	4	12.5	2	
	5	15	-	
2 Decline = 2 Low Cost	1	6.25	-	77365.35
	2	7.5	-	
	3	10	3	
	4	12.5	1	
	5	15	-	
3 Decline = 1 Medium Cost	1	12.5	-	3.64
	2	15	-	
	3	20	-	
	4	25	1	
	5	30	-	
4 Decline = 2 Medium Cost	1	12.5	-	953.07
	2	15	-	
	3	20	2	
	4	25	-	
	5	30	-	
5 Decline = 1 High Cost	1	20	-	2.9
	2	30	-	
	3	40	-	
	4	50	1	

	5	60	-	
6 Decline = 2 High Cost	1	20	-	2.92
	2	30	-	
	3	40	-	
	4	50	-	
	5	60	-	

The following figures depict the results on the Kartal map. We select node 32 as the central station. Figure 1 corresponds to moderately low passive splitter costs with a decline of 1 dB/km (Instance 1). As can be seen, there are three splitters with 2^3 and 2^4 outputs.

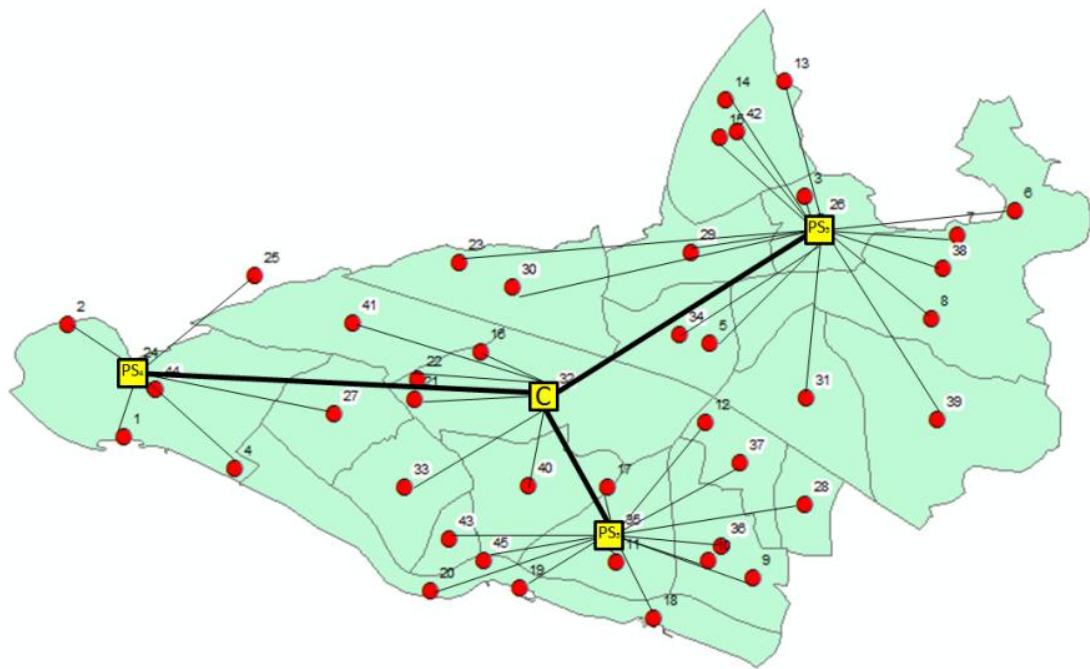


Figure 1- Illustration of Low Cost Structure for Passive Splitters (Instance 1)

Figure 2 shows the results for Instance 2 corresponding to low cost level for passive splitters and 2 dB/km decline. The sensitivity to the dB decline parameter can be seen when comparing Figures 1 and 2. In particular, in order to provide the same level of service, one more passive splitter has to be utilized in the high decline instance.

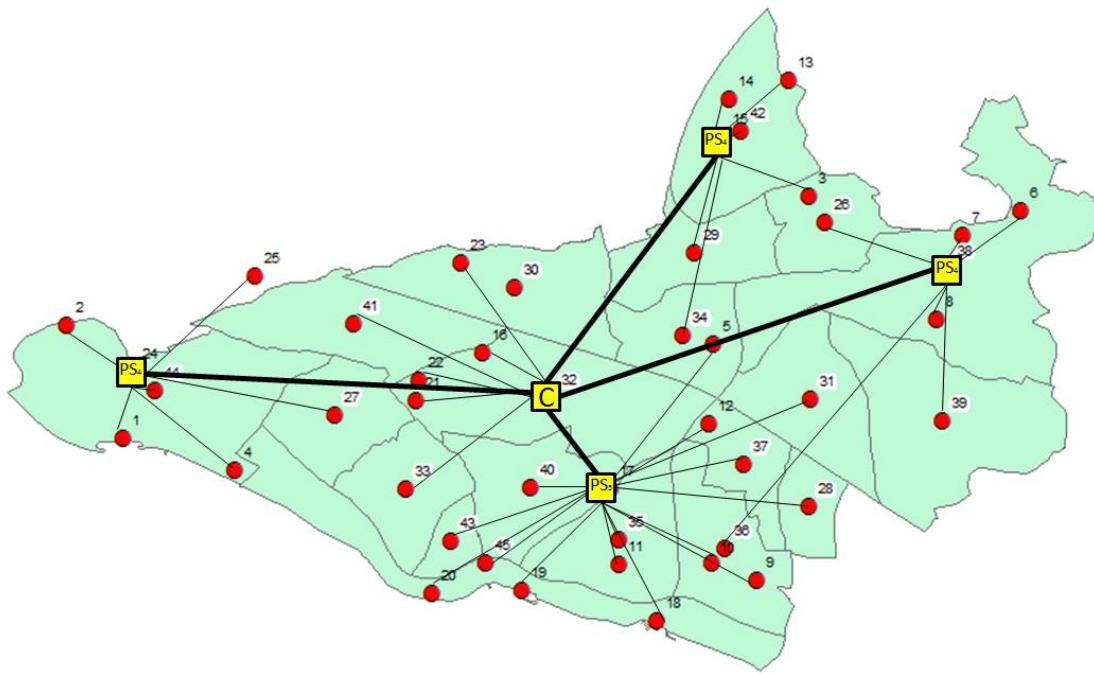


Figure 2- Illustration of Low Cost Structure for Passive Splitters (Instance 2)

Lastly, high cost structure is illustrated for İstanbul Kartal data in Figure 3. In this figure, the area that the central station serves is aggregated in a cloud shape to ease visualization. As can be seen, the number of passive splitters have decreased in accordance with the increase in cost.

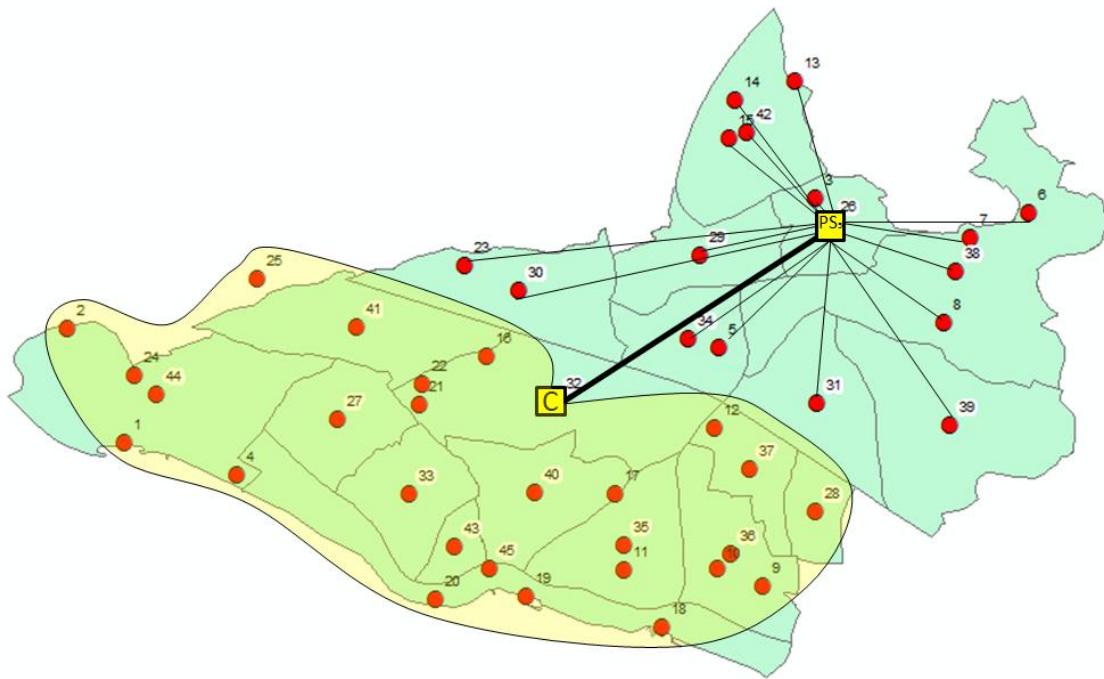


Figure 3 - Illustration of High Cost Structure for Passive Splitters (Instance 5)

In this preliminary analyses, we have observed that the resulting designs are sensitive to parameters such as passive splitter cost and dB decline values. As an immediate future research

direction the experimentation will be carried out for larger data sets and for a extensive range of parameter values.

References

1. Ergün, Y. İ., 2011. Network design problems in telecommunication: A review and future research issues. Technical Report, Middle East Technical University, Turkey.
2. Gavish, B., 1992. Topological design of computer communication networks-the overall design problem. European Journal of Operational Research 58, 149-172.
3. Gavish, B., 1991. Topological Design of Telecommunication Networks- Local Access Design Methods. Annals of Operations Research, 33: 17-71.
4. Gavish, B., Li, C., Simchi-levi, D., 1992. Analysis of heuristics for the design of tree networks. Annals of Operations Research, 36:77-86.
5. Kılçıl, F., 2012. A decision support system for shelter site location with GIS integration: Case for Turkey. Technical Report, Turkey.
6. Klincewicz, J. G., 1998. Hub location in backbone/tributary network design: a review. Location Science, 6: 307-335.
7. Labbe, G., Laporte, G., Martin, I.R., Salazar Gonzalez, J.J., 2004. The ring star problem: Polyhedral analysis and exact algorithm. Networks, 43(3):177-189.
8. Petrek, J., Siedt, V., 2004. A large hierarchical network star-star topology design algorithm. European Transactions on Telecommunications, 12(6):511-22.
9. Proestaki, A., Sinclair, M.C., 2007. Design and dimensioning of dual-homing hierarchical multi-ring networks. IEEE Proceedings-Communications, 147(2):96-104.
10. Rosenberg, E., 2005. Hierarchical Topological Network Design. IEEE/ACM Transactions On Networking, 13(6):1402-1409.
11. Thomadsen, T., 2005. Hierarchical Network Design. Technical Report, Technical University of Denmark, Denmark.
12. Thomadsen, T., Stidsen, T., 2007. The generalized fixed-charge network design problem. Computers & Operations Research, 34:997 – 1007.
13. Thomadsen, T., Larsen, J., 2007. A hub location problem with fully interconnected backbone and access networks. Computers & Operations Research, 34:2520-2531.

The Least Cost Influence Problem

Dilek Gunnec¹, S. Raghavan², Rui Zhang³

¹*Department of Industrial Engineering, Ozyegin University, Istanbul, Turkey*

^{2,3}*Smith School of Business and Institute for Systems Research, University of Maryland, MD*

¹dilek.gunnec@ozyegin.edu.tr, ²raghavan@umd.edu, ³ruizhang@umd.edu

Keywords: social networks, influence spread, dynamic programming, marketing.

Extended Abstract

Connectivity among people is amplified with recent advancements in internet technology as it increased the number of communication channels. Fast and easy spread of information over social networks facilitates the influence and consequently adoption among individuals. We consider the setting where peer influence plays a significant role in a consumer's product choice or there is a tangible benefit from using the same product as the rest of one's social network. We model cases where exterior incentives can be used to strengthen or "engineer" the product adoption process. Given a social network, the Least Cost Influence Problem (LCIP) finds the set of individuals to be given incentives in such a way that the product adoption reaches to a certain fraction of the population while the amount of incentives given is minimized.

More specifically, we are given an undirected graph $G=(V,E)$, where the node set V denotes the set of people in the network and edge set E represents the connections among people. Each node is associated with a hurdle and an "influence factor" which captures how much another node j influences node i if node j adopts. We follow a linear influence structure, i.e., node i adopts the product if and only if the sum of the influence from neighbors and the exterior incentive received is greater than or equal to his hurdle. Hurdle serves as a limit, which needs to be surpassed for the node to adopt. When a node adopts, the hurdles of the neighbors are updated (reduced). The goal of the problem is to minimize the amount of incentives given while guaranteeing that a certain fraction of the market will adopt at the end. We model the LCIP in a general setting using a mixed-integer program, and to capture the order of adoption, we introduce a time dimension.

The most prominent research in this area is the "influence maximization problem" [1]. Here the objective is, for a given parameter k , to find a set of k -nodes to maximize influence over the network. The k individuals are seeded (i.e., given the product for free) at the beginning and then the diffusion process takes place (with these k -nodes exerting an influence on their neighbors). Our LCIP deviates from this approach; the incentives in our

model are not seed products and they involve partial inducements tailored for a set of individuals. We introduce incentive-discrimination, which in a marketing setting could correspond to 10%, 50%, etc. discount coupons rather than free samples of a product. In our approach, an incentive is used to resolve a knot in the adoption spread that will lead to a larger diffusion with more influenced nodes at the end. Such catalysation addresses the trade-off of minimizing the amount of incentives given and reaching a greater number of individuals in the network.

The LCIP is an NP-Hard problem for general networks. However, we show that it is polynomially solvable on tree networks under the assumption that all neighbors of a node exert equal influence. We propose a dynamic programming (DP) algorithm to solve the problem over a tree network. Although the focus on trees as social networks may seem limited, they are prevalent. Recent visualization tools allow network data to be better read for data scientists. Popular visualizations include data from Facebook, Twitter and Wikipedia articles. Various aspects of these different environments may lead to visualizations of star subnetworks and tree networks. Further when a network can be decomposed into trees, our algorithm can be used to develop a fast dynamic programming (though not polynomial) algorithm for sparse networks. In our DP algorithm we fragment the tree network into subproblems. Each subproblem is solved optimally leading to an optimal solution for the problem. Subproblems are defined on star subnetworks which are collectively exhaustive. Before solving the problem over a star, an important step is to consider the link that connects the star with the rest of the network. We solve the problem over a star as if the influence over this link has been realized, meaning the influence from the rest of the network has traveled over this link and the hurdle of the root of the star has been updated. After the cost of this star is calculated, the star is compressed into a single node (with a new hurdle and a new influence factor) and it becomes a leaf node for the star in the next iteration. We repeat this process until we are left with a single star network. This promises that if we can iteratively reduce the tree network into a single star, we can solve the problem for the tree. We also describe a greedy algorithm for the problem. In the greedy algorithm, nodes are selected in the ascending order of the minimum of their current hurdles and influence factors, and the hurdles for the neighbors of the selected node are updated at each iteration. The optimality of the greedy algorithm is proved with a reinterpretation of the dynamic programming algorithm. Finally, we also provide a polyhedral description of the LCIP for tree networks. Specifically, a linear program whose constraint matrix is totally unimodular.

The LCIP is a combinatorial optimization problem that can be used in a variety of environments, from marketing to epidemiology. We enhanced our formulation for tree networks to address the LCIP on a general graph, and developed a branch-and-cut method. It is an interesting problem in diffusion of information as it offers a rich set of problems for future research. One of such problems is the LCIP under a stochastic influence structure. Understanding the structural relationship (depending on the location of the node over a network) of the optimal set of nodes is one direction especially for targeted marketing. Exploring both exact algorithms and heuristics to expand the solution approaches for these problems is a future research direction.

References

- [1] Kempe, D., J. Kleinberg, and E. Tardos. “Maximizing the spread of influence through a social network.” *Proceedings of the ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2003.

The One Commodity Pickup and Delivery Traveling Salesman Problem with Demand Intervals

Güneş Erdogan

School of Management University of Southampton, Highfield, Southampton, SO17 1BJ

Email: G.Erdogan@soton.ac.uk

Gilbert Laporte

Canada Research Chair in Distribution Management,
HEC Montréal, 3000 chemin de la Côte-Sainte-Catherine,
Montreal, Canada H3T 2A7

Email: gilbert.laporte@cirrelt.ca

Roberto Wolfler Calvo

LIPN, Université Paris 13, Paris, France

Email: Roberto.Wolfler@lipn.univ-paris13.fr

In the *One Commodity Pickup and Delivery Traveling Salesman Problem with Demand Intervals* (1-PDTSP-DI), we are given a complete undirected graph $G = (V, E)$, where $V = \{0, 1, \dots, n\}$ is the set of vertices and E is the set of edges. Vertex 0 is called the *depot*. Associated with each vertex $i \in V$, there are three parameters (l_i, b_i, u_i) that correspond to the lower bound, current supply, and the upper bound of the amount of commodity at the vertex, respectively. Associated with each edge $(i, j) \in E$, there is a travel time t_{ij} . The time required to pick up or deliver a commodity is denoted as h . A vehicle with capacity Q leaves the depot, performs a tour visiting each vertex at most once to pickup or deliver commodities, and returns to the depot. At the end of the tour, the resulting amount of commodities at every station i must lie within the interval $[l_i, u_i]$. The objective is to minimize the total time of traveling and handling the commodities.

1-PDTSP-DI is a generalization of the *One Commodity Pickup and Delivery Traveling Salesman Problem* (1-PDTSP), where a single capacitated vehicle visits all the customers to pick up and deliver the same commodity. Any instance of the 1-PDTSP can be converted to an instance of the 1-PDTSP-DI, where a pickup vertex i with supply $b_i > 0$ is assigned the parameters $(0, b_i, 0)$ and a delivery vertex j with demand $b_j < 0$ is assigned the parameters $(-b_j, 0, -b_j)$. This relationship proves that 1-PDTSP-DI is NP-Hard. The demand intervals introduce the flexibility of transshipment vertices, i.e. $\exists i \in V : l_i \leq b_i \leq u_i$. A transshipment vertex may or may not be visited, and may decrease the cost of routing by supplying or demanding commodities as required.

In this study, we provide an exact solution method for the 1-PDTSP-DI. We first study the subproblem of pickup and delivery decisions when the route of the vehicle is fixed and show that the subproblem is a Minimum Cost Network Flow Problem. Based on our findings, we present a model for the main problem that is capable of handling the routing decision as well as the pickup and delivery decisions. We provide our computational results based on two branch-and-cut algorithms, the first one directly using the model, and the second one utilizing a Benders decomposition.

Fleet sizing for offshore supply vessels

Yauhen Maisiuk¹, Irina Gribkovskaia²

*Faculty of Economics, Informatics and Social Sciences
Molde University Colelge - Specialized University in Logistics
Molde, Norway*

Abstract

Supply vessels provide offshore installations with necessary supplies on periodic basis from an onshore base according to weekly sailing plans. Each plan is built for a certain time horizon to guarantee required level of service to offshore installations at least cost. In a sailing plan several voyages are assigned to each vessel. The execution of sailing plans is affected by stochastic weather conditions. A vessel may not perform all visits within the planned voyage duration because of bad weather influencing vessel's sailing and service time. In such cases, additional vessels may be needed. Deciding on the number of supply vessels to hire on the long-term basis is an important part of the strategic fleet size planning. We present a discrete-event simulation model that evaluates alternative fleet size configurations of vessel fleet size for an annual time horizon.

Keywords: Fleet Sizing, Routing, Simulation

1 Introduction

Supply vessels provide offshore installations with necessary supplies from an onshore supply base. Installations require periodic visits that are performed

¹ E-mail: yauhen.maisiuk@himolde.no

² E-mail: irina.gribkovskaia@himolde.no

according to weekly sailing plans. A weekly sailing plan consists of a set of voyages with defined start days. A vessel may sail several voyages per week. Each voyage represents a route with duration of several days, starting and ending at onshore supply base and visiting a set of installations. The problem of optimal fleet composition and periodic routing of offshore vessels was studied in [2], [4].

Each sailing plan remains unchanged for a finite number of weeks before it is renewed because of changes in installations' demands and relocation of mobile offshore units. The sailing plan does not account for stochastic factors and therefore is used mainly for determining of fleet size. The major stochastic factor in supply vessel planning is uncertainty in weather conditions, which affects number of vessels needed to maintain the planned service. Because of delays associated with rough weather conditions some installations may not be serviced within the planned voyage duration, and not received supplies have to be delivered later with other vessels, which can significantly increase total cost of the fleet. Several approaches for creating robust weekly sailing plans are studied in [1].

The problem of determining the fleet size of vessels to hire on long-term basis to ensure supply service is an important part of the strategic fleet size planning. Similar problem for offshore anchor handling vessels was studied in [5]. We introduce a discrete-event simulation model for evaluation of alternative supply vessel fleet size configurations for an annual horizon consisting of several periods with different weekly sailing plans.

In the next section, we give an overview of the approach for construction of weekly sailing plans. Afterwards, the developed discrete-event simulation model is presented.

2 Algorithm for construction of weekly sailing plans

The two-stage method for generation of weekly sailing plans was introduced in [2]. We describe the method to introduce characteristics of the problem. At the first stage, a cheapest feasible voyage is generated for each vessel for every possible set of installations this vessel can serve within a possible voyage duration measured in days. The upper bound on a possible voyage duration is required to limit the lead time for delivery to installations included in voyage. Adding slack at the end of voyages is used to improve their robustness against weather uncertainty. Installations' locations, visit frequencies and amount of cargo per visit define the requirements, while vessel capacity, departure time from the base and installations' opening hours specify the constraints.

At the second stage, the selected voyages are input to a set covering model assigning voyages to start days. The model's notation is as follows. Set N_i contains all offshore installations, set V contains all available supply vessels, set T contains days of the planning horizon and set L is the set of all possible voyage durations measured in days. Let R_j be the set of pregenerated shortest feasible voyages a vessel $j \in V$ may sail, and where subset R_{jl} contains voyages with a duration of l days ($l \in L$). x_{jkt} is a binary variable equal to 1 if vessel j starts to sail voyage k ($l \in L$) on day t , and 0 otherwise. y_j is a binary variable equal to 1 if vessel j ($j \in V$) is used in the solution, and 0 otherwise. Parameter s_i is the required number of visits for installation i , and b_t is the maximum number of vessels that can be loaded at the supply base on day t . f_j defines the maximum number of days vessel j may be in service during the planning horizon. c_j is the daily time-charter cost of vessel j . The following three parameters are calculated in the voyage generator for each shortest feasible voyage. c_{jk} is the fuel cost of voyage k sailed by vessel j , d_{jk} is the duration of voyage k sailed by vessel j in days, and a_{ijk} is equal to 1 when vessel j services installation i on voyage k , and 0 otherwise. The model is formulated as follows:

$$(1) \quad \min \sum_{j \in V} c_j y_j + \sum_{j \in V} \sum_{k \in R_j} \sum_{t \in T} c_{jk} x_{jkt}$$

subject to

$$(2) \quad \sum_{j \in V} \sum_{k \in R_j} \sum_{t \in T} a_{ijk} x_{jkt} \geq s_i, i \in N$$

$$(3) \quad \sum_{k \in R_j} \sum_{t \in T} d_{jk} x_{jkt} - f_j y_j \leq 0, j \in V$$

$$(4) \quad \sum_{j \in V} \sum_{k \in R_j} x_{jkt} \leq b_t, t \in T$$

$$(5) \quad \sum_{k \in R_{jl}} x_{jkt} + \sum_{k \in R_j} \sum_{q=1}^{l-1} x_{jk, (t+q) \bmod |T|} \leq y_j, j \in V, t \in T, l \in L$$

The objective (1) of the model minimizes the sum of the total vessel time-charter costs and the fuel costs for all voyages. The frequency constraints (2) ensure that each offshore installation gets a required number of visits during the planning horizon. The linking constraints (3) guarantee that the total duration of all voyages sailed by a vessel does not exceed the total number of

days the vessel may be in service during the planning horizon T , while daily berth capacity constraints (4) are used to avoid that there are more vessels loaded at the supply base on a specific day than the onshore base can serve. The constraints (5) ensure that a vessel returns to the supply base before it starts a new voyage. To ensure a steady supply from the base, the departures of cargo from the supply base to each installation should be spread. The following constraint

$$(6) \quad \underline{p}_r \leq \sum_{j \in V} \sum_{k \in R_j} \sum_{h=0}^{h_r} a_{ijk} x_{jk, (t+h) \bmod |T|} \leq \bar{p}_r, \quad i \in N_r, t \in T, r \in F$$

guarantees that the departures to an installation are evenly spread throughout the planning horizon. Here F is the set containing all visit frequencies required by the installations, and N_r is the set of all installations with visit frequency r . To control the spread of departures throughout the planning horizon, the length of an auxiliary sub-horizon for the installations with visit frequency r is denoted by $0 \leq h_r \leq |T|$, where \underline{p}_r and \bar{p}_r are defined as the lower and the upper bound on the number of visits during this sub-horizon. This solution approach can generate optimal sailing plans for instances no more than 12 installations. For larger instances, a large neighborhood search heuristic of [4] can be applied.

3 Discrete-event simulation model

The planning of supply vessel operations is critical as operations at offshore installations depend on timely supplies from the shore. The planning period consists of periods with different sailing plans that define a simulation horizon. Voyages in a sailing plan are scheduled chronologically and repeated on weekly basis for the period of the sailing plan. To perform all voyages in the sailing plan several vessels are needed. Uncertainty in weather conditions affect execution of voyages resulting in a shortage of available vessels and a need for additional vessels. The option of allowing downtime at installations is not considered as waiting costs of drilling rigs and production installations are much higher than vessel costs. The company does not own offshore supply vessels. Instead, vessels are hired on time-charter and spot hire from shipping companies. Deciding on the number of supply vessels to hire on the long-term basis is an important part of the strategic fleet size planning. This decision has a strong economic effect on the total chartering cost as daily hire rates on the spot market can fluctuate between 20,000 and 45,000 euros.

The dependence of vessel supply operations on weather conditions makes

the problem highly stochastic. In compliance with safety regulations adopted by oil and gas producers working on Norwegian continental shelf, vessel is allowed to perform service at installations (loading and unloading) when the wave height does not exceed a certain threshold. The state of the weather at the time of planned visit to an installation may affect execution of vessel sailing plan, making some visits impossible during the planned voyage duration. Installations not visited on a voyage as planned must be served individually as soon as the weather allows. If there are no available vessels, a vessel from the spot market is hired for a single-visit voyage at a higher short-term rate. Spot rates represent another source of uncertainty as they are often significantly higher and volatile as opposed to time-charter rates. Time-charter vessels are cheaper than vessels from spot market, but require much longer commitment period. Probability distributions best fitting stochastic elements inherent to the problem are difficult to handle through analytical approaches. For these reasons, discrete-event simulation is chosen as a methodology.

The objective of the work was to develop a discrete-event simulation model for evaluation of alternative configurations of the fleet size. By experimenting with various values for the design factor (number of time-charter vessels) and examining corresponding efficiency measures (total vessel hire cost and number of vessel hire days), the number of vessels on time-charter hire minimizing the total cost is determined. The logic flowchart of our simulation model is depicted in the Figure 1 and key blocks are explained below.

Discrete-event simulation model imitates a sequence of voyages as it evolves over time. The state variables of the model describe model's state at only a countable number of points in time. The major state variable in the model is number of vessels in use. Another state variable is a number of non-performed planned visits to installations that arise because of the influence of the bad weather. The discrete points in time are the ones at which an event occurs, where an event is defined as an instantaneous occurrence that *may* change the state of the model [3].

The model accordingly simulates sequences of installations visit events that are triggered by corresponding voyage events. A voyage event is defined as a compound abstract object consisting of a number of visits to installations. A voyage occurs at a known start time and lasts for a planned period of time. Visit to an installation is defined as another event. These two types of events can be easily considered as processes as they are spread over time after the moment of its occurrence. The stochastic factor (weather conditions) influences on possibility and time of occurrence of installations' visit events.

The weather generation procedure is built upon the concept of alterna-

tion of durations of two distinct states of the weather window at each of the offshore installations for the duration of the simulation horizon. The simple distinction between only two weather states (operative and inoperative) is considered for the purpose of verification of visit occurrence at installation. The weather generator relies on the sampling of durations of operative and inoperative weather from a set of estimated probability distributions with respect to offshore location and month of the year. The applied weather modeling approach is similar to the one described in [5].

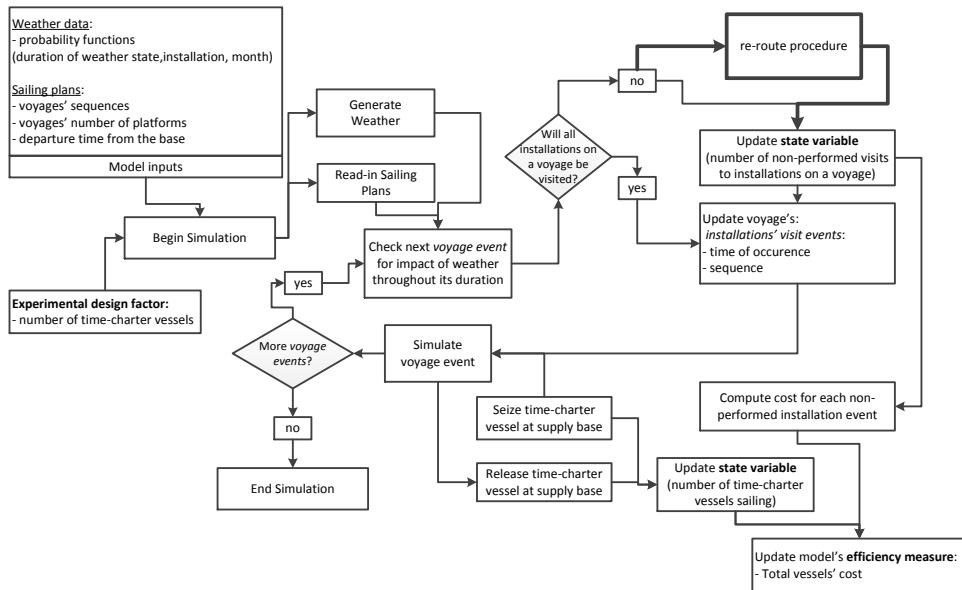


Fig.1 Flow diagram of discrete-event simulation model

The model has been implemented in two variants. In basic version, sequence of visits on a voyage is simulated as it is stated in sailing plan. However, in real life by the time when voyage should start, the weather forecast is known, and visits to installations may be re-scheduled. To account for that, we have included in the second variant of the model a re-route procedure. It imitates re-planning actions against weather uncertainty that are taken in real-life by marine coordinators in order to provide ordered supply service to offshore installations. The re-route procedure is used to find a feasible sequence of voyage visits where maximum number of installations affected by weather will

be visited within planned voyage duration. Comparison of two model variants shows that less non-performed visits to installations on planned voyages occur if the re-route procedure is included in the model. Finally, the model allows not only to define cost-efficient fleet size, but also evaluate robustness of sailing plans.

References

- [1] Halvorsen-Weare, E. and K. Fagerholt, *Robust supply vessel planning*, Lecture Notes in Computer Science **6701** (2011), 559-573.
- [2] Halvorsen-Weare, E., K. Fagerholt, L. M. Nonås and B. E. Asbjørnslett, *Optimal fleet composition and periodic routing of offshore supply vessels*, European Journal of Operational Research **223** (2012), 508-517.
- [3] Law, A. M., “Simulation Modeling and Analysis,” McGraw-Hill, Boston, 2007.
- [4] Shyshou, A., “Vessel planning in offshore oil and gas operations,” Ph.D. thesis, Molde University College, Molde, 2010.
- [5] Shyshou, A., I. Gribkovskaia and J. Barcelo, *A simulation study of the fleet sizing problem arising in offshore anchor handling operations*, European Journal of Operational Research **203** (2010), 230-240.

Pricing in telecommunication networks and bilevel optimization

Andrei V. Orlov*

*Institute for System Dynamics and Control Theory of SB RAS, Irkutsk, Russia, anor@icc.ru

1 Introduction

A hierarchy is one of the promising paradigm in mathematical programming in recent years [1]. The pioneering work on bilevel optimization [2] led to the monograph on the class of mathematical programs with equilibrium constraints (MPECs) [3] and, somewhat later, to the monographs on the bilevel programming problems (BPPs) [4, 5]. There are a lot of applications of the MPECs and BPPs in control, economy, traffic, telecommunication networks, etc. (see, e.g., [6]). An investigation of MPECs or BPPs in the view of elaboration of the efficiency numerical methods is the urgent challenge of contemporary theory and methods of Mathematical Optimization [1].

The bilevel problems in the classical statement [4, 5] represent optimization problems, which – side by side with ordinary constraints such as equalities and inequalities – include a constraint described as an optimization subproblem:

$$\left. \begin{array}{l} F(x, y) \downarrow \min_{x, y}, \quad x \in D, \quad y \in Y_*(x), \\ Y_*(x) \stackrel{\Delta}{=} \operatorname{Argmin}_y \{G(x, y) \mid (x, y) \in D_1\}. \end{array} \right\} \quad (\mathcal{BP})$$

Note, we consider an optimistic formulation of the bilevel problem (the goal of the upper level can be adjust with the actions of the lower level) [5]. Therefore the goal function F is to be minimized w.r.t. x and y simultaneously.

It seems that direct investigation of the problem (\mathcal{BP}) with the purpose of elaboration of solution methods is difficult to realize now. Therefore, we offer to investigate special bilevel classes [7, 8, 9] or bilevel applications.

In particular, tarification problems, which are pervasive in decision making, naturally lend themselves to a bilevel programming formulation with a specific structure (see, e.g., [6]). Several methods have been proposed in the literature, for a such kind of problems [10]–[15]. Nevertheless, development of new numerical methods, which are able to operate with high dimension problems, remains an urgent problem.

In this work we propose new methods of local and global search for the hierarchical problem of optimal pricing in telecommunication networks [16]. These methods based on a possibility of equivalent representation of a bilevel optimization problem as a nonconvex optimization problem [4, 5] (with the help of Karush-Kuhn-Tucker conditions), and on the applying the Global Search Theory (GST) (for solving of the obtained nonconvex problem) [17]–[20]. GST allows to construct efficient numerical methods for several classes of one-level and bilevel problems with nonconvex structures (with the dimension up to 1000) [7, 8, 9, 20, 21, 22, 23]. Therefore, this approach will be useful for bilevel problem of optimal pricing in telecommunication networks.

2 Problem formulation and its reduction

Consider a telecommunication traffic network $G = (V, U, A)$ consisting of a set of nodes V ($|V| = m$), and a set of arcs U ($|U| = n$) with incidence matrix A . The nodes represent origin, destination or transit stations for telecommunication traffics, and arcs represent transmission lines. Let the upper level player is the leader telecommunication operator, and the lower level player is the client that needs to communicate between nodes for transferring the information. The arc set of the network is partitioned into two subsets U_1 (toll arcs) and U_2 which are the set of links operated by the leader telecommunication operator and by the competitor operators respectively. Let $|U_1| = p$, $|U_2| = q$ ($p + q = n$).

Let c_i^1 is a fixed part of the cost of information unit transmission on the corresponding arc (for each arc of the graph G belonging to U_1), and x_i is an additional fee to the transfer, determined by the upper level player. So, the total cost of information unit transmission on the arbitrary arc belonging to U_1 is $c_i^1 + x_i$, $i = 1, 2, \dots, p$. Thus, $x = (x_1, x_2, \dots, x_p)$ is an upper level variable. The set $X = \{x \in \mathbb{R}^p \mid x_i \leq \bar{x}_i, i = 1, 2, \dots, p\}$ defines bounds of tariff changes. Also, let c_i^2 , $i = 1, 2, \dots, q$ is a fixed part of the cost of information unit transmission on the arcs belonging to U_2 .

Next, let the client (player of the lower level) wants to transfer K datasets from node s^k to node r^k . Each dataset has the volume δ^k , $k = 1, 2, \dots, K$. In that case a lower level variable is determined by the vector $y = (y^1, \dots, y^K)$, where $y^k \triangleq (y^{k1}, y^{k2}) \in \mathbb{R}^{p+q} = \mathbb{R}^n$, $k = 1, 2, \dots, K$. Each component of vector $y^k = (y^{k1}, y^{k2})$ is the volume of traffic passing through the corresponding arc: $y^{k1} = (y_1^{k1}, y_2^{k1}, \dots, y_p^{k1})$ is the traffic flow on the arcs belonging to U_1 , and $y^{k2} = (y_1^{k2}, y_2^{k2}, \dots, y_q^{k2})$ is the traffic flow on the arcs belonging to U_2 .

In addition, the throughput constraints on each arc are imposed:

$$0 \leq y_j^k \leq \bar{y}_j^k, \quad j = 1, 2, \dots, n, \quad k = 1, 2, \dots, K.$$

Also, we should introduce the following arc flows constraint:

$$Ay^k = \delta^k d^k, \quad k = 1, 2, \dots, K,$$

where $d^k \in \mathbb{R}^m$ is the zero vector with $d_{s^k}^k = +1$, $d_{r^k}^k = -1$. The latter constraint ensures that the total outgoing traffic from all nodes in the network is equal to the total amount of incoming traffic.

The goal of the upper level is to maximize the revenue, and the goal of the lower level is to minimize the cost of data transmission. So, we can formulate the following bilevel optimization problem:

$$\left. \begin{aligned} F(x, y) &\triangleq \sum_{k=1}^K \langle x, y^{k1} \rangle \uparrow \max_{x,y}, \quad x \in X, \\ y \in Y_*(x) &\triangleq \operatorname{Argmin}_y \left\{ \sum_{k=1}^K \langle c^1 + x, y^{k1} \rangle + \langle c^2, y^{k2} \rangle \mid \right. \\ &\quad \left. \mid Ay^k = \delta^k d^k, 0 \leq y_j^k \leq \bar{y}_j^k, \quad j = 1, 2, \dots, n, \quad k = 1, 2, \dots, K \right\}. \end{aligned} \right\} \quad (\mathcal{BP})$$

Note, we obtain a special bilinear-bilinear problem of bilevel optimization.

Next, let for the simplicity $K = 1$ without loss of generality. So, we get the following bilevel problem:

$$\left. \begin{aligned} \langle x, y^1 \rangle &\uparrow \max_{x,y}, \quad x \in X, \\ y \in Y_*(x) &\triangleq \operatorname{Argmin}_y \{ \langle c^1 + x, y^1 \rangle + \langle c^2, y^2 \rangle \mid Ay = \delta d, 0 \leq y \leq \bar{y} \}, \end{aligned} \right\} \quad (\mathcal{P})$$

where $x \in X = \{x \in \mathbb{R}^p \mid \underline{x}_i \leq x_i \leq \bar{x}_i, i = 1, 2, \dots, p\}$, $y = (y^1, y^2) \in \mathbb{R}^n$, $c^1 \in \mathbb{R}^p$, $c^2 \in \mathbb{R}^q$, A is the $(m \times n)$ -matrix, $d \in \mathbb{R}^m$, $\delta \in \mathbb{R}$, and $c^1 \geq 0$, $c^2 \geq 0$.

Bilevel problem (\mathcal{P}) may be reduced to several single-level nonconvex problems with the help of classical approach. First, we use the KKT-rule (or linear programming duality) for the lower level problem, and next we use the penalty approach (see [4], [5]):

$$\left. \begin{aligned} \Phi(x, y, u, v) &= \langle x, y^1 \rangle - \mu h(x, y, u, v) \uparrow \max_{x, y, u, v}, \\ (x, y, u, v) \in D &\triangleq \{(x, y, u, v) \mid \underline{x} \leq x \leq \bar{x}, Ay = \delta d, \\ &uA - v \leq (c^1 + x, c^2)^T, 0 \leq y \leq \bar{y}, v \geq 0\}, \end{aligned} \right\} \quad (\mathcal{DC}(\mu))$$

where $\mu > 0$ is a penalty parameter, $h(x, y, u, v) \triangleq \langle c^1 + x, y^1 \rangle + \langle c^2, y^2 \rangle - \langle \delta d, u \rangle + \langle \bar{y}, v \rangle$.

The latter problem is the problem of d.c. optimization with the goal function, represented as difference of two convex functions.

It is known, that nonconvex problems may have a large number of local solutions, which are far – even from the viewpoint of the goal function's value – from a global one [17, 24].

Direct application of standard convex optimization methods turns out to be inefficient from the view point of global search. So, there appears the need to construct new global search methods, allowing to escape from a stationary (critical) point.

3 Local and global search

For the purpose of solving the problem formulated above, we intend to construct the algorithms based on the Global Search Theory (GST) in d.c. programming problems developed in [17]–[23]. Global Search Algorithms based on GST consist of two principal stages: 1) a special local search methods, which takes into account the structure of the problem under scrutiny [17, 20, 22, 23, 25, 26]; 2) the procedures, based on Global Optimality Conditions, which allow to improve the point provided by the Local Search Method [17]–[23].

To the end of a local search for problem $(\mathcal{DC}(\mu))$ we apply the idea of consecutive solving partial problems with respect to two groups of variables [9, 20, 21, 22, 23, 26]. In order to do it, we separate the pair (x, v) and the pair (y, u) . For a fixed value of the pairs (x, v) and (y, u) problem $(\mathcal{DC}(\mu))$ becomes a problem of linear programming. So, we can consider the following auxiliary linear programs $((x^s, y^s, u^s, v^s) \in D)$:

$$\left. \begin{aligned} (1 - \mu)\langle(y^1)^s, x \rangle - \mu\langle\bar{y}, v \rangle &\uparrow \max_{(x, v)}, \quad (x, v) \in D(y^s, u^s), \\ \langle x^s - \mu(c^1 + x^s), y^1 \rangle - \mu\langle c^2, y^2 \rangle + \mu\langle \delta d, u \rangle &\uparrow \max_{(y, u)}, \\ (y, u) \in D(x^s, v^s), \end{aligned} \right\} \quad \begin{aligned} &(\mathcal{LP}_{xv}(y^s, u^s)) \\ &(\mathcal{LP}_{yu}(x^s, v^s)) \end{aligned}$$

where $D(x, v) \triangleq \{(y, u) \mid (x, y, u, v) \in D\}$, $D(y, u) \triangleq \{(x, v) \mid (x, y, u, v) \in D\}$.

These auxiliary problems can be solved with the help of standard software packages.

The procedures of Global Search for problem $(\mathcal{DC}(\mu))$ based on the corresponding strategy of global search for problems of d.c. minimization [17]–[23] because the goal function in problems of such kind may be represented as a difference of two convex functions.

In combination with directed selection, in the process of increasing the the value of parameters $\mu > 0$, the procedures of global search forms a method for solving problem (\mathcal{P}) .

The crucial moment of Global Search procedures consists in constructing an approximation of the level surface of the convex function, which generates the basic nonconvexity in the problem under consideration. For the purpose of constructing such an approximation we have to take account of the information related to the problems statements [17]–[23].

Computational testing of the elaborated methods on the test problems of optimal pricing in telecommunication networks has shown the workability and efficiency of the proposed approach.

4 Conclusions

In this work the hierarchical problem of optimal pricing in telecommunication networks is considered.

New algorithms of local and global search for this problem in optimistic formulation are elaborated. These algorithms are based on the reduction the problem to nonconvex d.c. optimization problems and on the Global Search Theory.

The workability and efficiency of the elaborated algorithms is demonstrated by computational simulation.

This work is carried out under financial support of Russian Foundation for Basic Research (projects no. 11-01-00270 and 12-07-33045).

References

- [1] J.-S. Pang, *Three modelling paradigms in mathematical programming*, Mathematical programming, Ser.B., Vol.125, pp. 297–323, 2010.
- [2] J. Bracken, J. T. McGill, *Mathematical programs with optimization problems in the constraints*, Operations research, Vol.21, pp. 37–44, 1973.
- [3] Z.-Q. Luo, J.-S. Pang, D. Ralph *Mathematical programs with equilibrium constraints*, Cambridge: Cambridge University Press, 1996.
- [4] J. F. Bard, *Practical Bilevel Optimization*, Dordrecht, The Netherlands: Kluwer Academic Publishers, 1998.
- [5] S. Dempe, *Foundations of Bilevel Programming*, Dordrecht, The Netherlands: Kluwer Academic Publishers, 2002.
- [6] B. Colson, P. Marcotte, G. Savard, *An overview of bilevel optimization*, Annals of operations research, Vol.153, pp. 235–256, 2007.
- [7] T. V. Gruzdeva, E. G. Petrova, *Numerical solution of a linear bilevel problem*, Computational Mathematics and Mathematical Physics, Vol.50, No.10, pp.1631–1641, 2010.
- [8] A. S. Strekalovsky, A. V. Orlov, A. V. Malyshev, *Numerical solution of a class of bilevel programming problems*, Numerical Analysis and Applications, Vol.3, No.2, pp.165–173, 2010.
- [9] A. S. Strekalovsky, A. V. Orlov, A. V. Malyshev, *On computational search for optimistic solutions in bilevel problems*, Journal of Global Optimization, Vol. 48, No. 1, pp.159–172, 2010.
- [10] M. Labbe, P. Marcotte, G. Savard, *A Bilevel Model of Taxation and Its Application to Optimal Highway Pricing*, Management Science, Vol.44, No.12, Part 1 of 2, pp. 345–358, 1998.

- [11] L. Brotcorne, M. Labbe, P. Marcotte, G. Savard, *A Bilevel Model and Solution Algorithm for a Freight Tariff-Setting Problem*, Transportation Science, Vol.34, No.3, pp.289–302, 2000.
- [12] L. Brotcorne, M. Labbe, P. Marcotte, G. Savard, *A Bilevel Model for Toll Optimization on a Multicommodity Transportation Network*, Transportation Science, Vol.35, No.4, pp.345–358, 2001.
- [13] M. Didi-Biha, P. Marcotte, G. Savard, *Path-based formulations of a bilevel toll setting problem*, In Optimization with Multivalued Mappings, eds. by S. Dempe and V. Kalashnikov, pp.29–50, Springer Science + Business Media, LLC, 2006.
- [14] V. Kalashnikov, F. Camacho, R. Askin, N. Kalashnykova, *Comparison of algorithms for solving a bi-level toll setting problem*, International Journal of Innovative Computing, Information and Control, Vol.6, No.8, pp.3529–3549, 2010.
- [15] J. F. Vallejo, R. M. Sanchez, *A path based algorithm for solve the hazardous materials transportation bilevel problem* Applied Mechanics and Materials, Vols.253–255, pp. 1082–1088, 2013.
- [16] Tsevendorj I. Optimality conditions in global optimization: contributions to combinatorial optimization / I. Tsevendorj // Habilitation to Supervise Research, University of Versailles Saint-Quentin, 2007. — 97 p.
- [17] A. S. Strekalovsky, *Elements of nonconvex optimization*, Novosibirsk: Nauka, 2003 (in russian).
- [18] A. S. Strekalovsky, *On the Minimization of the Difference of Convex Functions on a Feasible Set*, Computational Mathematics and Mathematical Physics, Vol.43, No.3, pp.380–390, 2003.
- [19] A. S. Strekalovsky, *Minimizing Sequences in Problems with d.c. Constraints*, Computational Mathematics and Mathematical Physics, Vol.45, No.3, pp.418–429, 2005.
- [20] A. S. Strekalovsky, A. V. Orlov, *A new approach to nonconvex optimization*, Numerical Methods and Programming (internet-journal: <http://num-meth.srcc.msu.su/english/index.html>), Vol.8, pp.160–176, 2007.
- [21] A. S. Strekalovsky, A. V. Orlov. *Bimatrix games and bilinear programming*, Moscow: FizMatLit, 2007 (in russian).
- [22] A. V. Orlov, A. S. Strekalovsky, *Numerical search for equilibria in bimatrix games*, Computational Mathematics and Mathematical Physics, Vol.45, No.6, pp.947–960, 2005.
- [23] A. V. Orlov, *Numerical solution of bilinear programming problems*, Comp. Math. and Math. Physics, Vol.48, No.2, pp.225–241, 2008.
- [24] R. Horst and H. Tuy, *Global Optimization. Deterministic Approaches*, Berlin, Springer-Verlag, 1993.
- [25] T. V. Gruzdeva, A. S. Strekalovsky, *Local search in problems with nonconvex constraints*, Computational Mathematics and Mathematical Physics, Vol.47, No.3, pp.381–396, 2007.
- [26] A. S. Strekalovsky, A. V. Orlov, A. V. Malyshev, *Local search in a quadratic-linear bilevel programming problem*, Numerical Analysis and Applications, Vol.3, No.1, pp.59–70, 2010.

Solving the Single Vehicle Routing Problem with Variable Capacity

F.V. Louveaux

Department of Business Administration, University of Namur, Belgium

J.J. Salazar-Gonzalez

DEIOC, Universidad de La Laguna, Tenerife, Spain

Email: flouveau@fundp.ac.be, jjsalaza@ull.es

In the capacitated vehicle routing problem (VRP), the objective is to find a set of delivery routes, starting and ending at the depot, of minimal total length. Customers have a known demand and the total demand on each route may not exceed a given capacity Q . As a consequence, the vehicle may have to return a few times to the depot in order to unload and resume its trip. Alternatively, several vehicles are needed. In the VRP literature, no difference is made between the two cases as the objective is to minimize total distance (unless vehicles of different types are used, which is known as the heterogeneous fleet case).

All these versions assume a vehicle with given size. However, at the moment to acquire a new vehicle or to rent one, there is some freedom of choice. Selecting a vehicle with a larger size may reduce the number of returns to the depot (or the number of vehicles). It is clear however that larger vehicles are more costly, both to acquire as to operate. Thus, a larger vehicle capacity implies a lower total distance travelled but larger operating costs. The reverse is true for a smaller vehicle. Thinking in terms of cost minimization (and thus not only in terms of distance minimization), a trade-off is clearly desirable there. To reach this trade-off, it is assumed that the decision maker can obtain a cost minimization function, which includes acquisition and routing/operating costs. It must be expressed as a function of the vehicle capacity and the travelled legs. As is typical in industry, most decision makers will distinguish fixed and variable (acquisition and operating) costs in a linear fashion. In the most simplified version, operating costs are linear in the distances and acquisition costs are linear in the vehicle capacity. This is in line with the literature, in particular on the heterogeneous fleet case (e.g., Fukasawa et al. (2006), Yaman (2006)).

Even in this simplest case where the objective function is linear in the routing costs and in the vehicle capacity, the question remains on how to adapt the formulations and methods of the VRP to the variable capacity case. As an example, in the standard two-index formulation of the VRP, the so-called capacity constraints define the feasibility set in terms of the demand of a set divided by the vehicle capacity. Plugging in a variable capacity into the existing constraints creates a non-linear model.

If only a very limited number of alternative capacities are available, the problem can trivially be solved by direct calculation of the few alternatives. The purpose of this paper is to examine the other cases. We will do this assuming some continuous range for the vehicle capacity. Of course, a given brand is only offering a discrete set of vehicle capacities. However, different brands typically offer different capacities. Thus, considering the many brands offering different vehicle capacity sizes, the offer is large enough to be considered continuous. It is the case for the possible truck volume, or truck load, or the number of seats in busses, e.g. Besides, there are cases where the vehicle is designed on demand which implies almost full choice of capacity.

In this paper, we consider the VRP version where a single vehicle may return several times to the depot. It is called the Single VRP (SVRP) and is a NP-hard problem that has been extensively studied (see Toth and Vigo (2002)).

Even if the potential vehicle capacity is included in a continuous range, it is easy to see that only a finite number of possible vehicle capacities have to be considered. We first consider an enumerative approach, which consists of solving a sequence of VRP's, starting from the one having the largest capacity. The number of VRP's to solve in this approach is unknown in advance. Based on computational experiments, this number is mostly very large in our benchmark instances.

We then proceed to a direct approach, based on two-index formulations of the VRP. We introduce several valid inequalities that allow us to have an Integer Linear Programming formulation of the SVRP with variable capacity. We describe separation procedures for these inequalities. We also show computational results that confirm the utility of these results for solving benchmark instances from VRPLIB.

References cited in this abstract

- R. Fukasawa, H. Longo, J. Lysgaard, M. Poggi de Aragao, M. Reis, E. Uchoa, and R.F. Werneck. Robust branch-and-cut-and-price for the capacitated vehicle routing problem. *Mathematical Programming*, 106:491-511, 2006.
P. Toth and D. Vigo. *The vehicle routing problem*. Society for Industrial and Applied Mathematics, 2002.
H. Yaman. Formulations and valid inequalities for the heterogeneous vehicle routing problem. *Mathematical Programming*, 106:365-390, 2006.

Computing the probabilities of small component sizes in large networks

Paolo Serafini

*Department of Mathematics and Computer Science, University of Udine, Italy
paolo.serafini@uniud.it*

We consider the problem of characterizing the structure of a very large random network by considering its asymptotic behavior. In particular, the distribution of the vertex degrees is known and we want to find the size distribution of the small components. It is a well known result [1,2,3] that either the network consists of a unique giant component (by giant it is meant that the component size goes to infinity as the network size goes to infinity) plus a number of small components (whose size remains finite) or it consists of an infinite number of small components. It turns out that these small components are trees with high probability.

A nice theory [2,3] has been developed based on generating functions of degree distributions from which one can derive the generating functions of the small component sizes. In a few cases it is possible to derive analytical results. In this note we want to show how to carry out a numerical computation of the coefficients of the generating functions for general arbitrary degree distributions.

Let p_k be the probability that a randomly selected vertex has degree k and let q_k be the excess degree probability, i.e., the probability that by selecting a random vertex of degree at least one and by selecting randomly one of its adjacent vertices this vertex has degree $k+1$. Asymptotically $q_k = (k+1)p_{k+1}/\bar{d}$, with \bar{d} the average degree. The associated generating functions are

$$G_0(x) = \sum_{k \geq 0} p_k x^k, \quad G_1(x) = \sum_{k \geq 0} q_k x^k = \frac{\sum_{k \geq 1} k p_k x^{k-1}}{\bar{d}} = \frac{G'_0(x)}{G'_0(1)}$$

Let s_k be the probability that a randomly selected vertex belongs to a component of size k , necessarily small. Let r_k the probability that after selecting a random vertex of degree at least one, this vertex belongs to a small component and by selecting randomly one of its adjacent vertices and by removing the corresponding arc this vertex belongs to a component of size k . Let the corresponding generating functions be (necessarily they don't have the constant term)

$$H_1(x) = \sum_{k \geq 1} r_k x^k, \quad H_0(x) = \sum_{k \geq 1} s_k x^k$$

The functional equations linking together $G_0(x)$, $G_1(x)$, $H_0(x)$ and $H_1(x)$ are

$$H_1(x) = x G_1(H_1(x)), \quad H_0(x) = x G_0(H_1(x)) \quad (1)$$

The usual problem consists in finding $H_0(x)$ and $H_1(x)$ from $G_0(x)$ and $G_1(x)$. We may also consider the inverse problem of finding $G_0(x)$ and $G_1(x)$ from $H_0(x)$ and $H_1(x)$, i.e., computing the degree distributions which gives raise to a particular small component size distribution. This problem presents interesting features. Arbitrary degree distributions of r_k and s_k may not be feasible, i.e., there may be no degree distribution which can lead to those values.

Here we consider the power series representation of the generating functions, so that the problem consists in finding the coefficients of $H_1(x)$ and $H_0(x)$ from the coefficients of $G_0(x)$ and $G_1(x)$.

We derive from (1)

$$\begin{aligned} \sum_{k \geq 1} r_k x^k &= x q_0 + x \sum_{h=1}^{n-1} q_h \left(\sum_{j \geq 1} r_j x^j \right)^h \\ \sum_{k \geq 1} r_k x^k &= x q_0 + \sum_{h=1}^{n-1} q_h x^{h+1} \left(\sum_{j \geq 0} r_{j+1} x^j \right)^h = x q_0 + \sum_{h=2}^n q_{h-1} x^h \left(\sum_{j \geq 0} r_{j+1} x^j \right)^{h-1} \end{aligned} \quad (2)$$

Let a_k^h be the coefficient of x^k in $(\sum_{j \geq 0} r_{j+1} x^j)^h$. Note that $a_k^1 = r_{k+1}$. From (2)

$$r_k = \sum_{h=2}^k q_{h-1} a_{k-h}^{h-1}$$

Hence the computation of r_k requires the coefficients $a_{k-2}^1, a_{k-3}^2, \dots, a_0^{k-1}$. In turn the computation of a_k^h requires the terms r_1, \dots, r_{k+1} and so to compute r_k we only need knowledge of r_1, \dots, r_{k-1} .

The recursion works as follows: initially $r_1 = q_0$, and then

$$\left. \begin{aligned} r_k &= \sum_{h=2}^k q_{h-1} a_{k-h}^{h-1}, \\ a_{k-1}^1 &= r_k \\ a_{k-1}^h &= \sum_{j=0}^{k-1} a_j^{h-1} a_{k-1-j}^1 = \sum_{j=0}^{k-1} a_j^{h-1} r_{k-j}, & h = 2, \dots, k-1 \\ a_h^k &= \sum_{j=0}^h a_j^{k-1} r_{h+1-j} & h = 0, \dots, k \end{aligned} \right\} \quad k = 2, \dots$$

We also derive from (1)

$$\sum_{k \geq 1} s_k x^k = x p_0 + \sum_{h=2}^n p_{h-1} x^h (\sum_{j \geq 0} r_{j+1} x^j)^{h-1} \quad (3)$$

so that

$$s_1 = p_0, \quad s_k = \sum_{h=2}^k p_{h-1} a_{k-h}^{h-1}, \quad k \geq 2$$

In this case the computation is straightforward since it involves all quantities previously computed. The computation can be also carried out symbolically. For instance in the case of a Poisson graph ($p_k = q_k = e^{-d} d^k / k!$, with d average degree) we may find out the exact result

$$r_k = s_k = \frac{k^{k-1} d^{k-1} e^{-k d}}{k!}$$

and for an exponential graph ($p_0 = 0$, $p_k = (1-a) a^{k-1}$, $k \geq 1$ with $0 < a < 1$ a parameter) we get

$$r_k = \frac{1}{2k-1} \binom{3(k-1)}{k-1} (1-a)^{2k-1} a^{k-2}, \quad s_k = \frac{1}{k} \binom{3(k-1)+1}{k-1} (1-a)^{2k} a^{k-1}$$

As for the inverse problem the recursion can be inverted, i.e., knowing the r_k values, we can infer the probabilities p_k and q_k . Indeed we have

$$r_k = q_{k-1} a_0^{k-1} + \sum_{h=2}^{k-1} q_{h-1} a_{k-h}^{h-1} \implies q_{k-1} = \frac{r_k - \sum_{h=2}^{k-1} q_{h-1} a_{k-h}^{h-1}}{a_0^{k-1}}$$

Computing the a_h^k values is straightforward knowing the r_k values. From the probabilities q_k we easily deduce the probabilities p_k , apart from the fact that p_0 cannot be derived from the q_k values. However $p_0 = s_1$ and so it is known a priori. Then we first set $p_k = q_{k-1}/k$ for $k > 0$ and then normalize these values so that $\sum_{k>0} p_k = 1 - p_0$.

However, it would be perhaps more interesting computing p_k and q_k from the s_k values without knowing the r_k values. This is more difficult and it will be matter of future investigation.

References

- [1] U. Brandes and T. Erlebach eds., *Network Analysis*, Springer Berlin, 2005.
- [2] M.E.J. Newman, “The structure and function of complex networks”. *SIAM Review*, **45**, 167–256, (2003).
- [3] M.E.J. Newman, “Component sizes in networks with arbitrary degree distributions”, *Physical Review E*, **76**, 2007.

A Branch-and-cut Algorithm for the Hub-Cycle Location Problem

Inmaculada Rodríguez Martín

DEIOC, Facultad de Matemáticas,

Universidad de La Laguna, Tenerife, Spain

Email: irguez@ull.es

Juan José Salazar González

DEIOC, Facultad de Matemáticas,

Universidad de La Laguna, Tenerife, Spain

Email: jjosalaza@ull.es

Hande Yaman

Department of Industrial Engineering,

Bilkent University, Ankara, Turkey

Email: hyaman@bilkent.edu.tr

Hubs facilities serve as switching points in many-to-many distribution networks that arise in telecommunications and transportation. Flows from different origins to different destinations are consolidated at hubs and routed together to benefit from economies of scale.

In this study, we propose a branch-and-cut algorithm for the hub-cycle location problem (HCLP). In HCLP, we are given a set of demand points. These points have pairwise traffic demands. We are also given the routing costs between all pairs of points. We open p hubs and assign each demand point to exactly one of these hubs. The demand points assigned to the same hub are connected with a cycle. The hub nodes are connected by direct links. The traffic between demand points assigned to the same hub are routed on the cycle incident at this hub whereas the traffic between demand points assigned to different hubs are routed through the hub network and through the cycles. The cost of routing on the cycles is independent of the quantity transported and is a function of the distance traversed. On the other hand, the routing cost in the hub network is a function of the distance and the quantity transported. The aim of the problem is to minimize the total cost of routing the traffic in the network.

HCLP arises in transportation and logistics applications where hubbing is used and demand nodes do not have sufficient demand to justify direct connection with the hubs.

We have not encountered any study on HCLP. If the costs associated with the cycles are zero and if we are given costs of assigning demand points to hubs, then we obtain the single allocation p -hub median problem. See Campbell et al. (2002) and Alumur and Kara (2008) for surveys on hub location. If the cost of routing traffic on the hub network is zero, then HCLP reduces to a plant-cycle location problem for which Labb   et al. (2004) propose a branch-and-cut algorithm. Another closely related problem is the multi-depot vehicle routing problem. Laporte et al. (1984, 1988) propose branch-and-bound algorithms and Baldacci and Mingozzi (2009) model it as a heterogeneous VRP where the vehicles at each depot are seen as a different type of vehicle and propose an exact algorithm. Nagy and Salhi (1998) consider a hub location routing problem with capacity and distance constraints. The authors present a model and propose a nested solution methodology.  etiner et al. (2010) study a multiple allocation hub location and routing problem for the Turkish postal services. They consider two objectives: minimization of the variable transportation cost and the number of vehicles needed to achieve a given service level. They propose an iterative hubbing and routing heuristic and present computational results using Turkish data where they allow tours of at most 450km's (one day travel time).

Our study contributes to the literature by introducing a new problem and a solution methodology that handles decisions on different levels of the network simultaneously. We derive strong formulations and devise a branch-and-cut algorithm for HCLP.

References

- S. Alumur and B.Y. Kara. Network hub location problems: The state of the art. *European Journal of Operational Research*, 190:1–21, 2008.
- R. Baldacci and A. Mingozzi. A unified exact method for solving different classes of vehicle routing problems. *Mathematical Programming*, 120:347–380, 2009.
- J.F. Campbell, A.T. Ernst, and M. Krishnamoorthy. Hub location problems. In Z. Drezner and H. Hamacher, editors, *Facility Location: Applications and Theory*, pages 373–407. Springer-Verlag, Berlin, 2002.
- S.  etiner, C. Sepil, and H. S ural. Hubbing and routing in postal delivery systems. *Annals of Operations Research*, 181(1):109–124, 2010.
- M. Labb  , I. Rodr  guez-Mart  n, and J.J. Salazar-Gonz  lez. A branch-and-cut algorithm for the plant-cycle location problem. *Journal of the Operational Research Society*, 55(5):513–520, 2004.

- G. Laporte, Y. Nobert, and D. Arpin. Optimal solutions to capacitated multidepot vehicle routing problems. *Congressus Numerantium*, 44:283–292, 1984.
- G. Laporte, Y. Nobert, and S. Taillefer. Solving a family of multi-depot vehicle routing and location-routing problems. *Transportation Science* 22:161–172, 1988.
- G. Nagy and S. Salhi. The many-to-many location-routing problem. *TOP*, 6:261–275, 1998.

Integer Programming Formulations for the k -Edge-Connected 3-Hop-Constrained Network Design Problem

I. Diarrassouba¹, V. Gabrel², L. Gouveia³, A. R. Mahjoub², P. Pesneau⁴

1. LMAH, University of Le Havre, 25 Rue Philippe Lebon, 76600, Le Havre, France
ibrahima.diarrassouba@univ-lehavre.fr

2. LAMSADE, Paris Dauphine University, Place du Maréchal De Lattre de Tassigny, 75775, Paris Cedex 16, France
{gabrel,mahjoub}@lamsade.dauphine.fr

3. Departamento de Estatística e Investigação Operacional - Centro de Investigação Operacional, Faculdade de Ciências, Universidade de Lisboa, Portugal
legouveia@fc.ul.pt

4. University of Bordeaux, INRIA Bordeaux - Sud-Ouest, IMB UMR 5251, France,
pierre.pesneau@math.u-bordeaux1.fr

Let $G = (V, E)$ be an undirected graph and $D \subseteq V \times V$, a set of pairs of nodes, called *demands*. If a pair $\{s, t\}$ is a demand in D , we call s and t *demand nodes* or *terminal nodes*. Let $L \geq 2$ be a fixed integer. If s and t are two nodes of V , an L -st-path in G is a path between s and t of length at most L , where the length is the number of edges (also called *hops*).

Given a weight function $c : E \rightarrow \mathbb{R}$, which associates the weight $c(e)$ to each edge $e \in E$ and an integer $k \geq 1$, the k -edge-connected L -hop-constrained network design problem (k HNDP for short) consists in finding a minimum cost subgraph of G having at least k edge-disjoint L -st-paths between each demand $\{s, t\} \in D$.

The k HNDP has applications in the design of survivable telecommunication networks where bounded-length paths are required. Survivable networks must satisfy some connectivity requirements that is, networks that are still functional after the failure of some links. As pointed out in [11] (see also [10]), the topology that seems to be very efficient (and needed in practice) is the uniform topology, that is to say that corresponding to networks that survive after the failure of $k - 1$ or fewer edges, for some $k \geq 2$. However, this requirement is often insufficient regarding the reliability of a telecommunications network. In fact, the alternative paths could be too long to guarantee an effective routing. In data networks, such as Internet, the elongation of the route of the information could cause a strong loss in the transfer speed. For other networks, the signal itself could be degraded by a longer routing. In such cases, the L -path requirement guarantees exactly the needed quality of the alternative routes. Moreover, in a telecommunication networks, usually several commodities have to be routed in the network between pairs of terminals. In order to guarantee an effective routing, there must exist a sufficient number of hop constrained paths between each pair of terminals.

The k HNDP has been studied in several special cases. In particular [1, 3, 4, 9] investigated the polytope of the problem when a single demand is considered ($|D| = 1$) and $L \geq 2$. They give a complete description of this polytope when $L = 2, 3$ and show that the problem can be solved in polynomial time in this case.

In [8], the authors study the problem when $|D| = 1$ and $L = 4$ and give an integer programming formulation for the problem based on the design variables. Papers [2, 5, 6, 7] also studied the problem but when several demands are considered ($|D| \geq 2$ and $L \geq 2$). They present integer programming formulations and lead some computational studies for the problem.

In this work, we investigate new integer programming formulations for the k HNDP when $|D| \geq 2$, $L = 2, 3$ and $k \geq 1$. We first present integer programming formulations introduced in the literature and which are defined on the original graph. Then, we propose two new approaches for the problem that are based on directed layered graphs, when $L = 2, 3$ and $k \geq 1$. One approach (called "separated" approach) uses a layered graph for each hop-constrained subproblem and the other (called "aggregated" approach) uses a single layered graph for the whole problem. These new approaches yield new integer programming formulations for the problem when $L = 2, 3$. We compare the different formulations in terms of linear programming relaxation. Finally, in the last section we present a computational study when $k = 3$.

Références

- [1] F. Bendali, I. Diarrassouba, A. R. Mahjoub and J. Mailfert, "The k edge-disjoint 3-hop-constrained paths polytope", *Discrete Optimization* 7 (4), 2010, pp. 222-233.
- [2] Q. Botton, B. Fortz, L. Gouveia and M. Poss, "Benders decomposition for the hop-constrained survivable network design problem", *To appear in INFORMS Journal of Computing*, 2011.
- [3] G. Dahl and L. Gouveia, "On the directed hop-constrained shortest path problem", *Operations Research Letters* 32, 2004, pp. 15-22.
- [4] G. Dahl, D. Huygens, A. R. Mahjoub and P. Pesneau, "On the k edge-disjoint 2-hop-constrained paths polytope", *Operation Research Letters* 34 (5), 2006, pp. 577-582.
- [5] B. Fortz, A. R. Mahjoub, S. T. McCormick and P. Pesneau, "Two-edge connected subgraphs with bounded rings : Polyhedral results and Branch-and-Cut", *Mathematical Programming* 105 (1), 2006, pp. 85-111.
- [6] L. Gouveia, "Using variable redefinition for computing lower bounds for minimum spanning", *INFORMS Journal on Computing* 10 (2), 1998, pp. 180-188.
- [7] D. Huygens, M. Labbe, A. R. Mahjoub and P. Pesneau, "The two-edge connected hop-constrained network design problem : Valid inequalities and Branch-and-Cut", *Networks* 49 (1), 2007, pp. 116-133.
- [8] D. Huygens and A. R. Mahjoub, "Integer programming formulation for the two 4-hop-constrained paths problem", *Networks* 49 (2), 2007, pp. 135-144.
- [9] D. Huygens, A. R. Mahjoub and P. Pesneau, "Two edge-disjoint hop-constrained paths and polyhedra", *SIAM Journal on Discrete Mathematics* 18 (2), 2004, pp. 287-312.
- [10] H. Kerivin and A. R. Mahjoub, "Design of Survivable Networks : A Survey", *Networks* 46, 2005, pp. 1-21.
- [11] C.-W. Ko and C. L. Monma, "Heuristics for designing highly survivable communication networks", *Technical Report, Bellcore, Morristown, New Jersey*, 1989.

Energy-aware operations management in telecommunication network with shared protection mechanism

Bernardetta Addis^{*} Giuliana Carello^{*} Sara Mattia[◊]

^{*}Dipartimento di Informatica, Università degli Studi di Torino, addis@di.unito.it

^{*}Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano, carello@elet.polimi.it

[◊]Istituto di Analisi dei Sistemi ed Informatica, CNR, sara.mattia@iasi.cnr.it

In the past years the concern on environmental impact and energy issues has been constantly increasing. Information and communication technology (ICT) plays a fundamental role in reducing the energy consumption of several human activities. Nevertheless, ICT is itself responsible for a significant portion of the global energy consumption, in fact it is considered to be responsible of a percentage which varies between 2% and 10% of the world power consumption.

Telecommunication networks are usually over-dimensioned with respect to the real amount of traffic. A careful operating management of telecommunication networks can reduce the carbon footprint of ICT. In fact, a significant amount of energy can be saved by routing the traffic demands in order to switch off unused network devices.

Another important feature of today networks is resiliency to fault, due to the relevance of the provided services. Several mechanisms are applied in real life networks in order to guarantee fault resiliency. Among them, protection strategies provide each demand with two paths, one used in the nominal condition, the primary path, the other in the fault one, the backup path. According to the share protection mechanism, the backup capacity can be shared by backup paths of demands which are not affected by the same failures, thus reducing significantly the needed capacity.

In this work we focus on the problem of optimally managing an energy aware resilient network: given a network and a set of available links of the network, a set of demands have to be routed with the aim of minimizing the overall link device energy consumption. Single arc failure and share protection mechanism are considered: for each demand, a primary and a backup path are provided. An amount of capacity is given on each arc, which cannot be exceeded by both primary and backup paths using that arc. When an arc is not used, it can be switched off and the corresponding energy saved. As fault conditions are usually quickly resolved, we assume that the energy consumption is due only to the link devices which provide the primary paths capacity. The aim of the problem is to minimize the overall energy consumption of the network. We investigate formulations of the problem, with the aim of developing upper and lower bounds, to be combined in an exact approach.

Enhanced dynamic programming algorithms for the constrained subtree of a tree problem

Agostinho Agra^a, Cristina Requejo^a, and Luidi Simonetti^b

^aDepartment of Mathematics, University of Aveiro, 3810-193 Aveiro, Portugal.

{aagra, crequejo}@ua.pt

^bInstituto de Computação, Fluminense Federal University, Brazil.

luidi@ic.uff.br

Extended Abstract

We consider an undirected tree $T = (V, E)$ rooted at node 0 with node set $V = \{0, 1, \dots, n\}$ and edge set E , a nonnegative integer profit c_i and a nonnegative integer weight w_i associated with each node $i \in V$, and a positive integer H representing the capacity of the knapsack. We define the profit of a subtree as the sum of the profits of the nodes included in it. The Constrained Subtree of a Tree Problem (CSTP) is to find a subtree $T' = (V', E')$ of T rooted at node 0 and such that $\sum_{i \in V'} w_i \leq H$ and $\sum_{i \in V'} c_i$ is maximized.

For each $i \in V$, let p_i denote the predecessor of node i in T and let x_i be the binary variable indicating whether node i is in V' . The CSTP can be formulated as the following integer linear programming problem:

$$\max \left\{ \sum_{i \in V} c_i x_i : x_{p_i} \geq x_i, i = 1, \dots, n, \sum_{i \in V} w_i x_i \leq H, x_i \in \{0, 1\}, i \in V \right\}.$$

This problem is related with the Precedence-Constrained Knapsack Problem (PCKP). In this more general problem the constraints $x_{p_i} \geq x_i$, $i = 1, \dots, n$, are replaced with the more general set of precedence relations: $x_i \leq x_j$, $\forall (i, j) \in R$ where R is the set of the precedence related pairs. In the PCKP the set R must define a partial ordered set. Obviously, the CSTP is a particular case of the PCKP. The CSTP appears in several practical problems. Some applications can be found in [2, 4, 5].

The CSTP is clearly NP-hard since the 0-1 knapsack problem is a particular case of the CSTP when the depth of the tree is one. However, the CSTP can be solved in pseudo-polynomial time using dynamic programming algorithms. Several different such algorithms have been developed, see [1, 2, 3, 4]. Specialized exact algorithms have also been considered. A specialized Branch and Bound was developed by Cho and Shaw [6]. More recently, an exact algorithm based on partitions of the feasible set was presented by Merwe and Hattingh [7] where the authors provide a

computational comparison between their algorithm, the Branch and Bound algorithm developed in [6] and the standard CPLEX optimizer.

We present algorithms corresponding to enhancements of the well-known dynamic programs for the CSTP. There are two main reasons to reuse and improve the dynamic programming algorithms for the CSTP. Firstly, the computational tests conducted in [7] ignored the dynamic programming algorithms since, following Cho and Shaw's arguments [6], their specialized Branch and Bound algorithm outperforms the dynamic programming algorithm presented in [1]. However this fact hardly can be accepted for all values of n since for $n \approx H$ dynamic programming algorithms become polynomial and, therefore, it would be expected that the dynamic programming algorithms to perform well when compared with algorithms whose worst case is exponential in n such as the Branch and Bound algorithm. Secondly, recent approaches have been successfully applied to improve dynamic programming algorithms for other (related) problems such as the weight constrained shortest path problem. Those approaches include the use of label-setting algorithms in order to avoid the record of dominated states; the use of a bidirectional search to avoid the increase of the non-dominated states; and the use of preprocessing to reduce the size of the problem.

We present two enhanced dynamic programming approaches. The first one is an enhancement of the algorithms given in [1, 2] with the inclusion of a preprocessing step and tests to eliminate dominated states. The second enhanced algorithm splits the tree into two subtrees and applies the first algorithm to each one. Then the optimal solution is found by merging the two lists of non-dominated states.

We report computational tests conducted to that compare, for a large set of instances, the performance of the Xpress Optimizer solver, the algorithm presented by Merwe and Hattingh [7] and the two dynamic programming approaches. The results show that the enhanced dynamic programming approaches outperform the two other algorithms.

Acknowledgements

The research of the first two authors was partially supported by *FEDER* funds through *COMPETE*– Operational Programme Factors of Competitiveness (“Programa Operacional Fatores de Competitividade”) and by Portuguese funds through the *Center for Research and Development in Mathematics and Applications (CIDMA)* and the Portuguese Foundation for Science and Technology (“FCT–Fundação para a Ciência e a Tecnologia”), within project PEst-C/MAT/UI4106/2011 with COMPETE number FCOMP-01-0124-FEDER- 022690.

The research of the third author was partially supported by CNPq 483.243/2010-8 and project CAPES/FCT 311/11.

References

- [1] G. Cho and D.X. Shaw. A depth-first dynamic programming algorithm for the tree knapsack problem. *INFORMS Journal on Computing*, 9:431–438, 1997.
- [2] D.S. Johnson and K.A. Niemi. On knapsacks, partitions, and new dynamic programming technique for trees. *Mathematics of Operations Research*, 8:1–14, 1983.

- [3] J.A. Lukes. Efficient algorithm for the partitioning of trees. *IBM Journal Research Development*, 18:217–224, 1974.
- [4] T. Magnanti and L. Wolsey. Optimal trees. In M.O. Ball, T.L. Magnanti, C.L. Monma, and G.L. Nemhauser, editors, *Network Models*, Handbooks in Operations Research and Management Science, Vol. 7, pages 503–615. Elsevier Science Publishers, North-Holland, 1995.
- [5] D.X. Shaw. Reformulation, column generation and lagragian relaxation for local access network design problems. Working paper, School of Industrial Engineering, Purdue University, 1994.
- [6] D.X. Shaw and G. Cho. The critical-item, upper bounds, and branch-and-bound algorithm for the tree knapsack problem. *Networks*, 31:205–216, 1998.
- [7] D.J. van der Merwe and J.M. Hattingh. Tree knapsack approaches for local access network design. *European Journal of Operational Research*, 174:1968–1978, 2006.

Checking Feasibility in Stationary Models of Gas Transportation Networks - Methodological Foundation

Ralf Gollmer, Rüdiger Schultz, **Claudia Stangl**

claudia.stangl@uni-due.de

University Duisburg-Essen, Germany

Checking the feasibility of transportation requests belongs to the key tasks in gas pipeline operation. In its most basic form, the problem is to decide whether a certain quantity of gas can be sent through the network from prescribed entries to prescribed exit points. In the stationary case, the physics of gas flow together with technological and commercial side conditions lead to a pretty big (nonlinear, mixed-integer, finite dimensional) inequality system.

The approach presented in this talk relies on transforming nonlinearities into a more accessible form, reducing the problem dimension of the underlying NLP. This transformation approach is embedded into a heuristic procedure for finding promising switching decisions. The system of (linear) flow conservation and (nonlinear) pipe equations which results from Kirchoff's laws is transformed into an equivalent nonlinear system, where most flow and pressure variables get eliminated, because they are explicit functions of a relatively small group of variables, consisting of one flow variable per fundamental network cycle and two pressure variables per active arc. Apart from the explicit formulae for flow and pressure variables, the transformed system contains implicit equations whose number equals the number of fundamental cycles of the network and whose unknowns are just the variables from the mentioned group. The approach aims at checking feasibility for a set of switching states of active elements, either predefined or resulting from a transshipment heuristic. The principal idea behind the transformation, i.e. elimination of variables from relations using Kirchhoff's laws, dates back to at least the work of Hamam and Brameller [HB71] and has been picked up repeatedly later on by Mallinson and Fincham [MFB⁺93] and Rios-Mercado et al. [RMWSB02]. Compared to this work, the approach taken here incorporates not only compressors, but further active elements, such as resistors and control valves. Moreover, it can handle substantially meshed gas distribution networks. It aims at checking feasibility for a set of switching states of active elements, either predefined or resulting from a transshipment heuristic. Any NLP solver can be used.

Transformation of Nonlinearities For a given setting of all the active elements (*switching state*) of the network, the directed graph $\bar{G} = (V, \bar{A}) \subset G$ models the relevant network, where \bar{A} denotes the set of all arcs not being in closed state.

Let \mathcal{A}^+ denote the node-arc-incidence-matrix and \mathcal{A} be the submatrix of full row rank arising from the deletion of one row, corresponding to a preselected, pipe-incident root node $\bar{u} \in V$ with pressure variable $p_{\bar{u}}$.

Let π and $|Q_0| Q_0$ denote the vectors with components $\pi_u, u \in V \setminus \bar{u}$, and $|Q_{0,a}| Q_{0,a}, a \in \bar{A}$. For $a = (u, v) \in A_{cg} \cup A_{cv}$, i.e., for compressor groups and control valves, π_u and π_v denote squared outgoing and ingoing pressures and $\Delta_a := \pi_v - \pi_u$. For $a \notin A_{cg} \cup A_{cv}$, we define $\Delta_a = 0$. The diagonal matrix $\alpha = \text{diag}(\Lambda_a)$ has entries which either are the pipe specific values Λ_a , $a \in A_{pi}$, defined by the lenght, roughness and diameter, or 0, otherwise. The vector of all ones is denoted by $\mathbb{1}$.

The equations for flow conservation at the nodes and pressure drop on the arcs read:

$$\mathcal{A}Q_0 = d, \quad (1)$$

$$\Delta - \mathcal{A}^T \pi - \alpha |Q_0| Q_0 = -\pi_{\bar{u}} \mathcal{A}^T \mathbb{1}, \quad (2)$$

After splitting $\mathcal{A} = (\mathcal{A}_B, \mathcal{A}_N)$ into basis and non-basis parts according to a spanning tree and some reformulations, (1)–(2) are equivalent to

$$Q_{0,B} = \mathcal{A}_B^{-1} d - \mathcal{A}_B^{-1} \mathcal{A}_N Q_{0,N}, \quad (3)$$

$$\pi = \pi_{\bar{u}} \mathbb{1} - (\mathcal{A}_B^T)^{-1} (\alpha_B |Q_{0,B}| Q_{0,B} - \Delta_B), \quad (4)$$

$$\alpha_N |Q_{0,N}| Q_{0,N} - \Delta_N = \mathcal{A}_N^T (\mathcal{A}_B^T)^{-1} (\alpha_B |Q_{0,B}| Q_{0,B} - \Delta_B). \quad (5)$$

Compared to (1)–(2), one ends up with a much smaller implicit part in the transformed system, namely (5). It has just $\text{card}(Q_{0,N})$ equations, i.e. as many as fundamental cycles in \bar{G} . For strongly meshed gas distribution networks, such as those met in many parts of Europe, this number still can be substantial. For weakly meshed gas transportation networks, however, values of 1

or 2 already become practically relevant. If the cycles of \bar{G} are arc disjoint, then (5) is separable with respect to the components of $Q_{0,N}$.

Altogether, the feasibility system we use comprises relation (5), pressure bounds applied to the right-hand sides of (4), and some specific conditions for resistors, control valves, and compressor groups.

Control valves allow for a pressure reduction, when traversed in the nominal direction. Compressor groups are modeled at an aggregated level, without resolution to individual machines. Inequalities specify that compressor groups allow for a pressure increase in their active mode when traversed in the nominal direction. In order to approximate the characteristic diagram, lower and upper bounds are imposed on the flow for the active mode. We impose lower and upper bounds on the pressure ratio and on the pressure increase. Resistors either produce a constant pressure loss, or a loss nonlinearly depending on ingoing pressure and flow.

Search for Promising Switching Decisions To enable the approach described above, we have to fix switching states for control valves, valves, and compressor groups. We use two different techniques to fix these binary decisions. First we use a transshipment heuristic, and then we try some sets of given switching states for all active elements. These sets are constructed by using expert knowledge about the network and by collecting sets of switching states from the transshipment heuristic in cases where they previously led to a feasible solution for some nominations.

Building the Transshipment Model In gas networks often there are subnetworks of elements representing bigger entities typically comprising compressor groups and other active arcs, enabling a limited number of internal flow paths only. We collapse internal nodes and arcs of such an entity into a single node, which is connected to all nodes on the boundary of the entity in both directions. The arcs are assigned small cost coefficients. The node representing the entity is assigned the balance of all in- and outflows related to nodes within the entity.

All switchable arcs, i.e., valves, control valves, and compressor groups, outside of entities are substituted by one or two directed arcs depending on the signs of their flow bounds. Cost coefficients are assigned heuristically, for instance, in increasing order starting with forward arcs representing compressor groups, followed by forward arcs modeling control valves and arcs representing valves pointing into both directions. Finally, backward arcs for control valves and compressor groups are assigned the largest costs.

Each connected component of the network remaining after the removal of all entities and switchable arcs is collapsed into a single node, which is assigned the balance of the nominations for the entries and exits within the component. Arcs in both directions connect the node representing such a connected component with all boundary nodes of entities and endpoints of other switchable elements being incident to the component. To all these arcs, the sum of the friction coefficients α_a (c.f. (2)) in the component is assigned as cost.

Using the Transshipment Model The moderate size of the transshipment problem allows for rapid solution by any state-of-the-art linear programming solver. From the optimal solution, switching states for active arcs in the original network are derived. Namely, if in the optimal solution there is no flow passing through an element outside an entity, we choose the off-state for it. Reflecting expert knowledge, for all the elements inside a specified entity, a suitable decision from the corresponding subnetwork operation mode is chosen by a set of rules based on the amount of flow and the flow direction through the entity.

References

- [HB71] Y.M. Hamam and A. Brameller. Hybrid method for the solution of piping networks. *Proc. IEE*, 118(11):1607–1612, 1971.
- [MFB⁺93] J. Mallinson, A.E. Fincham, S.P. Bull, J.S. Rollet, and M.L. Wong. Methods for optimizing gas transmission networks. *Annals of Operations Research*, 43:443–454, 1993.
- [RMWSB02] R. Z. Ríos-Mercado, S. Wu, L. R. Scott, and E. A. Boyd. A reduction technique for natural gas transmission network optimization problems. *Annals of Operations Research*, 117(1):217–234, 2002.

Checking Feasibility in Stationary Models of Gas Transportation Networks – Case Study

Ralf Gollmer, Rüdiger Schultz, Claudia Stangl
ralf.gollmer@uni-due.de
University Duisburg-Essen, Germany

Checking the feasibility of transportation requests belongs to the key tasks in gas pipeline operation. In its most basic form, the problem is to decide whether a certain quantity of gas can be sent through the network from prescribed entries to prescribed exit points. In the stationary case, the physics of gas flow together with technological and commercial side conditions lead to a pretty big (nonlinear, mixed-integer, finite dimensional) inequality system. Using the elimination and approximation techniques presented in the talk "‘Checking Feasibility in Stationary Models of Gas Transportation Networks - Methodological Foundation’" the remaining system can be solved using standard NLP-solvers.

Within an industry project these ideas are applied to the planning tasks of the biggest german transportation system operators and it's real-world networks, comprising in total about 11.000 km of pipes.

As done for the traditionally used simulation tool the gas network is split in three parts – one for low-calorific gas (L-gas) and the northern and southern parts of the network for high-calorific gas (H-gas).

These three networks have different characteristics. While the L-gas network in its main part serves as a distribution network, both H-gas networks combine high-pressure transport lines with low-pressure distribution parts.

Besides the different numbers of network elements the complexity of the resulting transformed nonlinear problem differs due to the topologies of the networks, since each fundamental cycle corresponds to one flow variable.

The L-gas network has a peculiarity: there is a couple of distribution regions being fed via control valves without remote access, i.e. with fixed output pressure. This allows for a decomposition of the problem.

Although the combinatorial character of the planning problem is tackled by a heuristic, the results for these real-world problems are at least very encouraging and of substantial use for network operators.

The talk presents the characteristics of these networks, the complexity of the resulting problems, and computational results.

The Double Vehicle Routing Problem with Multiple Stacks: Exact Approaches

María Batista-Galván *Transportes Interurbanos de Tenerife, S.A.U., C/Punta de Anaga 1,38111 Santa Cruz de Tenerife, Spain, mdbatista@titisa.com*

Manuel Iori *DISMI, Università degli Studi di Modena e Reggio Emilia, Via Amendola 2, Pad. Buccola, 42122, Reggio Emilia, Italy, manuel.iori@unimore.it*

Jorge Riera-Ledesma*

DEIOC, Universidad de La Laguna, Av. Astrofísico Francisco Sánchez s/n, 3271 La Laguna, Spain, jriera@ull.edu.es

The *Double Traveling Salesman Problem with Multiple Stacks* [1] (DTSPMS) is a pickup-and-delivery single-vehicle routing problem which performs the pickup operations before the deliveries, and loads the collected products into a capacitated vehicle as they are picked up. This problem arises when the pickup and delivery regions are widely separated, and the transportation cost between both regions is fixed and therefore not considered as part of the optimization problem.

The pickup and the delivery points, in addition to a depot in each region, are known in advance. There is a routing cost related to each pair of points for each region. Each product is associated exactly with a pickup point in the pickup region and with a delivery point in the delivery region. All products have identical shape and size, and the vehicle has a loading space divided into stacks of a fixed height. This loading space is big enough to store all products, and the loading operations follow a Last-In-First-Out (LIFO) policy. That means that each loaded product is placed at the top of one of those stacks, and only the products located at the top of a stack can be unloaded from the vehicle.

The DTSPMS collects each product following a Hamiltonian tour in the pickup region starting at the pickup depot, and delivers the products also following a Hamiltonian tour in the delivery region starting at the delivery depot. Note that, the LIFO policy determines the order of collection and delivery, therefore any description of both Hamiltonian tours should specify unambiguously the pickup and delivery sequence. The aim is to minimize the total routing cost satisfying the stacks height and the LIFO policy. Therefore, the DTSPMS combines two instances of the directed Traveling Salesman Problem, one for the pickup region and another for the delivery region, with a combinatorial loading problem. This loading problem establishes whether both tours are compatible with respect to the features of the vehicle container.

A generalization of this problem considers the variation where a single container is not enough to collect all products and therefore more than one vehicle have to perform the collection and the delivery. This implies that the problem has to design multiple routes for the vehicles collecting and delivering the products.

We introduce and formulate this generalization, called the Double Vehicle Routing Problem with Multiple Stacks. We propose a three index formulation and a set-covering formulation that have motivated a Branch-and-Cut algorithm and a Branch-and-Cut-and-Price algorithm, respectively. The performance of both algorithms has been studied on a wide family of benchmark test instances.

References

- [1] H. L. Petersen, O. B. Madsen, The double travelling salesman problem with multiple stacks – Formulation and heuristic solution approaches, *Eur. J. Oper. Res.* 198 (2009) 139–147.

A Multi-Start Approach for Optimizing Routing Networks with Vehicle Loading Constraints

Angel A. Juan

Department of Computer Science, Multimedia, and Telecommunication. IN3-Open University of Catalonia, 08018 Barcelona, SPAIN. e-mail: ajuanp@uoc.edu

Javier Faulin

Department of Statistics and Operations Research. Public University of Navarre, 31006 Pamplona, SPAIN. e-mail: javier.faulin@unavarra.es

Oscar Domínguez

Institute of Intelligent Systems and Numerical Applications in Engineering (SIANI). University of Las Palmas de Gran Canaria, 35017 Las Palmas de Gran Canaria, SPAIN. e-mail: oscar@opein.com

Keywords *Vehicle Routing Problem; Vehicle Packing; Multi-start algorithms; Biased randomization; Heuristics.*

Extended Abstract

1. Introduction

This contribution proposes an efficient algorithm, with a reduced number of parameters, for solving the Two-Dimensional Loading Capacitated Vehicle Routing Problem (2L-CVRP) in routing networks. This problem combines two of the most important issues in logistics, i.e. vehicle routing and packing problems. The algorithm uses a multi-start approach, which is designed to avoid local minima and also to make the algorithm an easily parallelizable one. At each restart, a biased randomization of a savings-based routing algorithm is combined with an enhanced version of a classical packing heuristic to produce feasible good solutions for the 2L-CVRP. Experimental results show that our approach outperforms several best-known solutions from previous work, both in quality as in terms of the computational time needed to obtain them.

2. Our multi-start method

First of all, notice that our method integrates the packing issue as part of the routing-construction process. That is, the proposed algorithm will generate routing solutions implicitly taking into account the loading constraints too. The algorithm starts by computing a dummy but feasible solution. For each customer, this initial solution assigns one round-trip route from the depot to it. The algorithm also computes the savings associated with each edge as proposed in

Clarke and Wright (1964). These edges are then listed from highest to lowest savings. At this point, a multi-start process is initiated. At each iteration of this process, the savings list of edges is randomized using a biased probability distribution. In our case, a geometric distribution is employed to induce this biased randomization behavior. This distribution uses one single parameter, α ($0 < \alpha < 1$). As discussed in **Juan et al (2010)**, by performing a biased randomization of the savings list, edges are selected in a different order at each iteration of the multi-start process while, at the same time, the logic behind the savings-based heuristic is maintained, i.e.: edges with higher savings are more likely to be selected than those with lower savings. Then, until the savings list gets empty, an iterative process begins in which the edge at the top of the biased-randomized list is extracted. This edge connects two different routes. If – and only if– these two routes can be merged without violating any constraint –both in terms of capacity and in terms of loading–, then the route merging is carried out.

In order to check feasibility of items packing in the truck, the algorithm employs a multi-start biased-randomized version of the Best-Fit heuristic with items rotation (**Burke et al, 2004**). This biased-randomization process is similar to the previous one. In this case, however, the biased randomization applies over the list of items to be loaded. Again, we propose to use a geometric distribution with one single parameter, β ($0 < \beta < 1$). A new parameter, *maxPackIter*, controls the maximum number of times that the randomized Best-Fit heuristic will be run before assuming that the items cannot be fitted into a single vehicle –i.e. before assuming that the merge is not possible since the loading constraint would be transgressed.

At this point, a new feasible solution is ready. The routing component of this solution can be improved further by using the cache and splitting techniques described in **Juan et al (2011)**. The cache technique is a fast local search process that adds ‘memory’ to the algorithm, allowing it to achieve a faster convergence to a pseudo-optimal solution. The splitting technique constitutes an even more interesting local-search process. This divide-and-conquer strategy performs the following basic sequential steps: (a) it splits the new feasible solution into different disjoint (and feasible) parts according to some geometric properties; (b) for each of these parts, it dissolves the associated routes and starts an iterative process in which the aforementioned methodology is repeated during *maxSplitIter* iterations to obtain an improved ‘local’ solution to the current part –which, by construction, represents a smaller-size 2L-CVRP–; and (c) reunifies all ‘local’ solutions into an enhanced ‘global’ solution. Notice that the use of a multi-start process with randomized savings lists contributes to escape from local minimum and also facilitates parallelization of the algorithm.

3. Some computational results

In order to compare the efficiency of our approach, some classical benchmark instances for the 2L-CVRP were selected from the web site www.or.deis.unibo.it/research.html. Moreover, for each of these instances, five different classes are considered. For each of the chosen

benchmark instances, our algorithm was tested in all five classes. For each instance-class combination, ten independent iterations (replicas) were run using a different seed for a pseudo-random number generator. Each replica was run for a maximum time of 500 seconds, i.e. each instance-class combination was run for a total time of 83 minutes. Both the best solution found (*BEST10*) as well as the average value of the generated solutions (*AVG10*), were registered. These *BEST10* and *AVG10* values were compared with the results obtained by **Fuellerer et al (2009)** and are shown in **Table 1**.

Table 1. Gaps between Fuellerer's (2009) and our Multi-start approach for class 1.

Class	Instance	BEST10			AVG10		
		Fuellerer	MS	Gap	Fuellerer	MS	Gap
1	E016-03m	278.73	278.73	0.00%	278.73	278.73	0.00%
	E016-05m	334.96	334.96	0.00%	334.96	334.96	0.00%
	E023-05s	568.56	568.56	0.00%	568.56	568.56	0.00%
	E031-09h	610.00	610.00	0.00%	611.22	610.00	-0.20%
	E033-04g	837.67	837.67	0.00%	837.67	837.67	0.00%
	E041-14h	861.79	861.79	0.00%	862.37	861.96	-0.05%
	E076-07s	690.20	687.60	-0.38%	691.94	687.60	-0.63%
	E101-10c	819.56	819.56	0.00%	819.56	819.56	0.00%
	E256-14k	616.69	613.00	-0.60%	632.68	619.39	-2.10%
<i>Averages</i>				-0.11%			-0.33%

References

- Burke E.K., Kendall G, and Whitwell G. 2004. A new placement heuristic for the orthogonal stock-cutting problem. *Operations Research*, 52:655–671.
- Clarke G. and Wright J.W. 1964. Scheduling of vehicles from a central depot to a number of delivery points. *Operations Research* 12:568-81.
- Fuellerer G, Doerner K, Hartl R and Iori M 2009. Ant colony optimization for the two-dimensional loading vehicle routing problem. *Computers and Operations Research* 36:655–673.
- Juan A, Faulin J, Jorba J, Riera D, Masip D and Barrios B 2011. On the use of Monte Carlo simulation, cache and splitting techniques to improve the Clarke and Wright saving heuristics. *Journal of the Operational Research Society* 62(6): 1085-1097.
- Juan, A., Faulin, J., Ruiz, R., Barrios B., and Caballé S 2010. The SR-GCWS hybrid algorithm for solving the capacitated vehicle routing. *Applied Soft Computing* 10: 215-224.

Integrated Airline Robust Scheduling and Fleet Assignment under Demand Uncertainty *

Luis Cadarso, Ángel Marín

Matemática Aplicada y Estadística, Universidad Politécnica de Madrid

Madrid, Spain

luis.cadarso@upm.es, angel.marín@upm.es

Abstract

This paper looks at the airline-scheduling problem and develops an integrated approach that optimizes schedule design, fleet assignment and passenger use so as to reduce costs and create fewer incompatibilities between decisions. As passenger demand is characterized by uncertainty, we introduce stochastic variations caused by daily passenger demands in actual operations. To consider such stochastic disturbances we develop a stochastic-demand scheduling model where robust itineraries are introduced to ameliorate misconnected passengers. This integration leads to a huge model difficult to solve: an improved and accelerated Benders decomposition is proposed. The analytical work is supported with a case study involving the Spanish airline, IBERIA. Our approach shows that the number of misconnected passengers can be reduced when robust planning is applied.

Key words: Schedule design, Fleet assignment, Stochastic demand, Robustness.

1 Introduction

The airline schedule planning problem is defined as the sequence of decisions that need to be made to make a flight schedule operational. Given the high level of competition in the airline industry, effective decision making is crucial to the profitability of an airline. However, this decision making should not be only based on the available airline's resources. Passenger demand fluctuations arising from stochastic market demands could affect the actual performance of the planned schedules. In practice, the performance of an optimal plan could be reduced when applied to actual operations where passenger demand fluctuations occur. In other words stochastic disturbances arising from variations in daily passenger demand could affect the optimality of the fleet assignments and timetables. Therefore, to set a good flight schedule, not only does the fleet and related supply have to be considered, but passenger demand fluctuations arising from stochastic market demands in actual operations also have to be taken into account. This is the motivation for this study in which we focus on the integration of the decision making process. Our goal is to achieve simultaneous rather than sequential solution, because a simultaneous solution will generate more economical solutions and create fewer incompatibilities between the decisions. Moreover, with the integration of the different planning

*This research was supported by project grant TRA2011-27791-C03-01 by the 'Ministerio de Economía y Competitividad, Spain'.

process phases a greater robustness degree may be achieved, obtaining smoother solutions, which in case of incidents may be recovered in an easier way.

1.1 Contributions

In this paper, we present a mixed integer linear programming model for the schedule design and fleet assignment problem that accounts for demand uncertainty.

Our major contributions include:

1. We develop a robust, stochastic and integrated approach to solve the airline-scheduling problem, where schedule design, fleet assignment and passenger use are jointly solved.
2. Passengers' flows are obtained through different itineraries in the network accounting for demand uncertainty.
3. We introduce robustness into the model accounting for expected misconnected passengers.
4. We use Benders decomposition. In order to speed up convergence we use a new set of cuts.
5. The model is tested using a simplification of IBERIA's network.

2 Literature Review

The fleet assignment problem has been deeply studied. Hane et al. [5] present a multi-commodity flow model. They show various ways to reduce the problem size: variable aggregation, cost perturbations, dual simplex with steepest-edge pricing, and intelligent branch and bound strategies. Sherali et al. [11] present a tutorial on the basic and enhanced models and approaches that have been developed for the fleet assignment problem, including integration with other airline decision processes.

Lohatepanont and Barnhart [9], in their incremental optimization approach, select flight legs to include in the flight schedule and simultaneously optimize aircraft assignments to these flight legs. Kim and Barnhart [7] consider the problem of designing the flight schedule for a charter airline. Exploiting the network structure of the problem, they develop exact and approximate models and solutions, and compare their results using data provided by an airline. Lan et al. [8] consider passengers who miss their flight legs due to insufficient connection time. They develop a new approach to minimize passenger misconnections by re-timing the departure times of flight legs within a small time window. Jiang and Barnhart [6] maximize the number of potentially connecting itineraries weighted by their respective revenues allowing for limited changes in the schedule. Dumas et al. [4], using a passenger flow model devised in Dumas and Soumis [3], improve the fleet assignment model by taking stochastic demand prediction as inputs, and aim at computing expected numbers of passengers on each itinerary. Cadarso and Marín [2] look at the deterministic airline-scheduling problem and develop an integrated approach that optimizes schedule design, fleet assignment and passenger use so as to reduce costs and create fewer incompatibilities between decisions. The analytical work is supported with a case study involving the Spanish airline, IBERIA.

3 Problem Description

Frequencies and departure times must be determined for every itinerary for each market. Moreover, fleet types must be assigned to every flight leg. Two agents interact; the aircraft flow in the physical network

(supply), and passengers using the flight legs (passenger demand).

Supply. The network consists of airports and the feasible airways linking them. The airports are defined by the operations that can be performed within them and are characterized by available slots for landing and taking off. A flight leg is defined by an origin, destination and a departure time.

Each flight is assigned to a fleet type, with each fleet mainly characterized by its seating capacity and cruise speed and flight time depending on the assigned fleet type. In a tactical problem, used airplanes may vary due to uncertainty. It makes no sense, however, to plan to use a large number of aircraft if they have low utilization rates because of crew costs and issues of amortizing of the fleet. Consequently, minimum average block hour utilization is imposed for every fleet type.

A completely new schedule is not usually welcome in an airline because the carrier may have obligations in some markets especially if governments funding for 'essential services' is involved or there is a need to retain a market presence as a competitive force against rivals or to ensure retention of slots.

Passenger Demand. Unconstrained demand is characterized by the origin airport, destination airport, and the desired departure time. Demands for markets vary from day to day and we accordingly consider them as random variables. Their distributions are generally modeled as normal truncated at zero, or gamma, for small demands (Swan [12]).

Although the passengers have a desired departure time, it is not a fixed value, passengers will accept without any additional cost a departure time from a set of compatible times in each market.

For each market, passengers are considered in all possible itineraries. Each itinerary is defined by a set of flight legs that connect airports and a departure time and can be composed of one or more flight legs including intermediate stops at airports. If connecting time is not enough, passengers may misconnect and to minimize this, robust itineraries are introduced. A robust itinerary is one that minimizes misconnections resulting from a lack of time to change planes. Connecting time is thus a trade-off between an airline's available resources and passengers' perception regarding the inconvenience of waiting time.

Many markets face the problem that passengers will not have available ideal flight. Then, passengers will choose either a compatible, but different market or travel with another airline. We define compatible markets as those ones with identical origin and destination but alternative departure times. Therefore, some passengers will choose a compatible market, according to the relative recapture rate, and some will choose to travel with a different carrier; the recapture rate being a measure of the probability of accepting an alternative itinerary based on the time of day of departure, length of trip, and connections. It may also be that passengers willing to fly in a flight cannot find a seat, and thus the airline will try to offer them alternatives to fly. Depending on the available itineraries of other airlines the recapture rate can be calculated, and some of the disrupted passengers will take alternative itineraries offered.

Because competition effects are not considered, every flight leg will probably be crowded because demand in the market is unconstrained. This situation is not, however, realistic because demand will be shared between flight legs in real life. To represent this, the capacity offered for each flight leg is not 100% but that of the average load factor found in the airlines' records. Additionally, competition within the same airline is avoided by imposing a separation time between flight legs operating from the same origin and to the same destination. Consequently, the model adjusts the unconstrained demand to the available capacity on the scheduled flight legs.

4 Integrated Robust Airline Scheduling and Fleet Assignment Model

We use an integrated, stochastic and robust model for timetable, fleet assignment and passenger use optimization to obtain a global optimum solution. The model is based on Cadarso and Marín's [2] multi-commodity flow problem approach. The model in that paper was deterministic. However, due to the uncertainty in the passenger demand a two-stage optimization model is proposed: all supply related decisions are made in the first stage, while passenger demand realizations are in the second stage.

Solution Approach: Benders Decomposition. The problem has two decision levels: first, the system operator chooses flight legs and fleet assignment and second, passengers choose itineraries. To apply Benders decomposition, we divide the problem into a master problem defining a feasible network and a subproblem assigning the demand to this network. Benders decomposition iterates between the Master Model (MM) and the Sub-Model (SM) to find an optimal solution. In each iteration, the dual variables of the SM define Optimality Benders Cuts (OBC), which are added to the constraints of the master problem. The process continues until it converges under convex assumptions verified by the model.

Accelerating Benders Decomposition. The proposed Benders SM has a set partitioning structure, making it degenerate, that is, it has a non-unique dual solution. Thus, the chosen dual solution may correspond to a weak cut, increasing the number of iterations needed to reach convergence. Magnanti and Wong [10] solved this problem by choosing the strongest cut in the sense of the pareto-optimality. A cut is called pareto-optimal if no cut dominates it.

Benders [1] showed that after a finite number of steps his algorithm finds an optimal solution or proves that not exists. Finding a solution in a finite number of steps is in practice not good enough, and hence performance issues are improved using the previous pareto-optimal cuts.

Improved Accelerated Benders Algorithm. The necessity of solving repeatedly the Benders Master Problem is a bottleneck in the previous proposed algorithm. This is due to the fact that the problem is integer and the cuts added in each iteration complicate the problem.

Consequently, a new approach is proposed. First, the Benders Master Problem is relaxed. Then, the Accelerated Benders Algorithm is used to solve the problem until the convergence is reached. Every obtained OBC will be valid for the integer Master Problem, so they are automatically added when integrality constraints are reactivated. Then, again the Accelerated Benders Algorithm is used to solve the problem until convergence is reached.

Although the number of integer problems to solve has been reduced, the problem still remains because once the convergence for the relaxed problem has been reached, the first iteration with integrality constraints faces with a lot of $OBCs$, making it hard to solve to optimality by exact methods. However, surrogate constraints theory may be used to solve this problem. We also limit the solution space in order to get solutions in reasonable computational times.

5 Summary of Computational Experiments

All of our computational experiments is for the major airline in Spain, IBERIA. We present some results related to different realistic networks. Computational results show how robustness could be achieved, although at a price. The robust approach is compared with a non-robust one: misconnected passengers are reduced.

References

- [1] Benders, J. F., *Partitioning procedures for solving mixed-variables programming problems*, Numerische Mathematik. **4**(1) (1962), 238–252.
- [2] Cadarso, L. and Marín, A., *Robust passenger oriented timetable and fleet assignment integration in airline planning*, Journal of Air Transport Management. **26** (2013), 44–49
- [3] Dumas, J. and Soumis, F., *Passenger Flow Model for Airline Networks*, Transportation Science. **42**(2) (2008), 197–207.
- [4] Dumas, J., Aithnard, F. and Soumis, F., *Improving the objective function of the fleet assignment problem*, Transportation Research Part B: Methodological. **43**(4) (2009), 466–475.
- [5] Hane, C., Barnhart, C., Johnson, E., Martsen, R., Nemhauser, G. and Sigismondi, G., *The Fleet Assignment Problem: Solving a Large-Scale Integer Program*, Mathematical Programming. **70** (1995), 211–232.
- [6] Jiang, H., and Barnhart, C., *Robust Airline Schedule Design in a Dynamic Scheduling Environment*, Computers and Operations Research. (2011), 10.1016/j.bbr.2011.03.031.
- [7] Kim, D., and Barnhart, C., *Flight Schedule Design for a Charter Airline*, Computers & Operations Research. **34** (2007), 1516–1531.
- [8] Lan, S., Clarke, J.P., and Barnhart, C., *Planning for Robust Airline Operations: Optimizing Aircraft Routings and Flight Departure Times to Minimize Passenger Disruptions*, Transportation Science. **40** (2006), 15–28.
- [9] Lohatepanont, M., and Barnhart, C., *Airline Schedule Planning: Integrated Models and Algorithms for Schedule Design and Fleet Assignment*, Transportation Science. **38** (2004), 19–32.
- [10] Magnanti, T. L., and Wong, R. T., *Accelerating Benders Decomposition: Algorithmic Enhancement and Model Selection Criteria*, Operations Research. **29**(3) (1981), 464–484.
- [11] Sherali, H.D., Ebru, K., and Xiaomei, Z., *Airline Fleet Assignment Concepts, Models, and Algorithms*. European Journal of Operational Research, **172** (2006), 1–30.
- [12] Swan, W.M., *Airline demand distributions: passenger revenue management and spill*, Transportation Research Part E. **38**(3) (2002), 253–263.

Heuristics procedures to solve the multi-commodity Pickup-and-Delivery Traveling Salesman Problem

Hipólito Hernández-Pérez ^{1,2}

*DEIOC, Facultad de Matemáticas
Universidad de La Laguna
La Laguna, Tenerife, Spain*

Juan-José Salazar-González ³

*DEIOC, Facultad de Matemáticas
Universidad de La Laguna
La Laguna, Tenerife, Spain*

Abstract

The “multi-commodity Pickup-and-Delivery Traveling Salesman Problem” (m -PDTSP) is a generalization of the well-known “Traveling Salesman Problem” in which cities correspond to customers providing or requiring known amounts of m different products, and the vehicle has a known capacity. Each customer must be visited exactly once by the vehicle serving the demands of the different products while minimizing the total travel distance. It is assumed that a unit of a product collected from a customer can be supplied to any other customer that requires this product. We discuss heuristic algorithms for the m -PDTSP. First, we introduce a heuristic that finds a solution of the m -PDTSP. After, we present several procedures which improve a solution (even when this solution is infeasible). These improvement procedures include a branch-and-algorithm where some variables are fixed. Computational

experiments on randomly generated instances.

Keywords: Traveling Salesman; Pickup-and-Delivery; Branch-and-Cut; Multi-commodity Flow.

1 Introduction

Many practical applications in transportation involve routing and delivery optimization problems. This paper considers the following generalization of the *Traveling Salesman Problem* (TSP). A finite set of cities is given and the travel cost from one city to another city is assumed to be known, and not necessarily symmetric. One specific city is considered to be a *depot* while the other cities are identified as *customers*. Each customer requires some given quantities of different products and/or provides some given quantities of other different products. A unit of a product collected from a customer can be supplied to any customer that requires this product. It is assumed that the vehicle has a fixed *capacity* and must start and finish the route at the depot. The route must be a Hamiltonian tour through all the cities. Then, the *multi-commodity Pickup-and-Delivery Traveling Salesman Problem* (*m-PDTSP*) is the problem of finding a route for the vehicle such that it picks up and delivers all the quantities of the different products satisfying the vehicle-capacity limitation and minimizing the total travel cost. Since each city is visited once, each unit of a product loaded on the vehicle stays on the vehicle until it is delivered to its destination. For that reason we say that the *m-PDTSP* is a *non-preemptive* problem.

The initial load of any product in the vehicle when leaving the depot is unknown, and must be determined within the optimization problem. The variant of the *m-PDTSP* where the initial load of any product is fixed can also be solved through the approach described in this paper with a slight modification of the instance.

An application of the *m-PDTSP* occurs in the context of inventory repositioning. Assume that a set of retailers are geographically dispersed in a region. Often, due to the random nature of the demands, some retailers have an excess of inventory of some products while others have such strong sales that they

¹ This work has been supported by “Ministerio de Educación y Ciencia”, Spain (research project MTM2009-14039-C06-01).

² Email: hhperez@ull.edu.es

³ Email: jjsalaza@ull.es

need additional stock. In many cases the firm may decide to transfer inventory from retailers that have experienced below average sales to those that have experienced above average sales. Determining the cheapest way to execute a given stock transfer (with the requirement that each localization has to be visited exactly once) is the m -PDTSP. Anily and Bramel [1] give applications for a problem related to the m -PDTSP, and these applications are also valid for the m -PDTSP.

Another application arises in the context of a self-service bike hiring system, where every night a capacitated vehicle must visit the bike stops in a city to collect or deliver bikes to restore the initial configuration of the system. Chemla et al. [4] and Raviv et al. [12], among others, approached the case where the bikes are all identical as a problem related to the 1-PDTSP. When there are different types of bikes (for example, with and without baby chairs) the problem is related to the m -PDTSP.

The m -PDTSP was introduced by Hernández-Pérez and Salazar-González [11]. They introduce a Mixed Integer Linear Programming model for the m -PDTSP, discuss a classical decomposition technique and describe some strategies to solve the problem based on a branch-and-cut algorithm. Computational experiments on randomly generated instances with up to 30 customers, 3 products and small vehicle capacities are analyzed.

A problem closely related to the m -PDTSP which includes many sources and many destinations with various commodities is the *Non-Preemptive Capacitated Swapping Problem* (NCSP) proposed by Erdoğan et al. [5]. In the NCSP there is one depot and a set of customers. Each customer may supply one item of a commodity and/or demand one item of another commodity. Every item of each commodity has a weight. When a customer supplies and demands one item of the same commodity the customer is called a *transshipment customer*. The items cannot be split and they cannot be dropped off in an intermediate customer. However, a transshipment customer can be used as an intermediate customer for the commodity supplied and demanded by this customer. The problem consists of finding a minimum cost route to satisfy all customers. Differences in this problem with respect to the m -PDTSP are: (1) a feasible solution may not be a Hamiltonian circuit; (2) each customer can only supply and/or demand one unit of one product; (3) there exist transshipment customers; (4) each commodity has a weight. Erdoğan et al. [5] describe a branch-cut-algorithm to solve the NCSP. Bordenave et al. [2] describe a branch-and-cut algorithm to solve the particular case of the NCSP where the vehicle capacity is equal to one unit and all the item weights are also one unit.

The *one-to-one* m -PDTSP is a particular case of the m -PDTSP where each commodity has one origin and one destination. It can be considered as a DARP without time windows requirements. In that sense, in the one-to-one m -PDTSP one assumes that the initial load of the vehicle when leaving the depot is zero unless the depot is the source of a commodity. Hernández-Pérez and Salazar-González [10] describe a branch-and-cut algorithm for this problem solving instances involving up to 24 customers and 15 commodities. Rodríguez-Martín and Salazar-González [13] propose and compare several metaheuristic approaches to solve instances with up to 300 customers and 600 commodities.

The literature have also addressed many other related problems. We now mention some articles dealing with one-commodity variants. Chalasani and Motwani [3] study the special case of the 1-PDTSP where the delivery and pickup quantities are all equal to one unit. This problem is called *Q -delivery TSP* where Q is the capacity of the vehicle. Anily and Bramel [1] consider the same problem with the name *Capacitated TSP with Pickups and Deliveries*. For the 1-PDTSP, Hernández-Pérez and Salazar-González [9] present an exact algorithm solving instances with up to 200 customers. Hernández-Pérez et al. [7] describe an hybrid algorithm that combines Greedy Randomized Adaptive Search Procedure and Variable Neighborhood Descent paradigms. Zhao et al. [14] propose a Genetic Algorithm that on average gives better results. Finally, Mladenovic et al. [6] describe a General Variable Neighborhood Search improving the best-known solution for all benchmark instances and solving instances with up to 1000 customers.

2 The Algorithm for the m -PDTSP

Clearly, the m -PDTSP is an \mathcal{NP} -hard optimization problem in the strong sense since it coincides with TSP when the vehicle capacity is large enough. What is more, checking if there is a feasible solution of a m -PDTSP instance is a strongly \mathcal{NP} -complete problem even when $m = 1$. It is shown in [8].

First Solution

This section provides a heuristic procedure to find a first solution, not necessary a feasible solution. Before that, we introduce some notation. The m -PDTSP is defined on a complete directed graph $G = (V, A)$. The node set $V = \{1, \dots, n\}$ represents the customers, included the depot which is denoted by 1. For each pair of customers i and j we have the arc $a = (i, j) \in A$ and a

travel cost c_{ij} . Let $K = \{1, \dots, m\}$ be the set of products. For each customer $i \in V$ and each product $k \in K$ let q_i^k be the demand of product k associated with customer i . When $q_i^k > 0$ customer i offers q_i^k units of product k and when $q_i^k < 0$ customer i requires $-q_i^k$ units of product k . We assume that $\sum_{i \in V} q_i^k = 0$ for all $k \in K$, i.e., each product is conserved through the route. The capacity of the vehicle is denoted by Q .

As mentioned above, contrary to what happens in the TSP, finding a feasible solution (optimal or not) for the 1-PDTSP is a problem with a hard computational complexity.

Nevertheless, checking if a given TSP solution is feasible for m -PDTSP is a very trivial task as it can be done $O(mn)$. Indeed, let us consider a path \vec{P} defined by the vertex sequence v_1, \dots, v_s for $s \leq n$. Let $l_i^k(\vec{P}) := l_{i-1}^k(\vec{P}) + q_{v_i}^k$ be the load of the vehicle when coming out from v_i if the vehicle follows this path and enters customer v_1 with load $l_0^k(\vec{P})$. Notice that $l_i^k(\vec{P})$ could be negative if $l_0^k(\vec{P}) = 0$ and, therefore, the minimum quantity of load of commodity k for a feasible solution through the path \vec{P} is $-\min_{i=0}^s \{l_i^k(\vec{P})\}$.

With this notation, the *infeasibility* of the path \vec{P} as:

$$\text{infeas}(\vec{P}) := \max \left\{ 0, \max_{i=0}^s \left\{ \sum_{k \in K} l_i^k(\vec{P}) \right\} - \sum_{k \in K} \min_{i=0}^s \{l_i^k(\vec{P})\} - Q \right\}$$

Note that the infeasibility of any path \vec{P} does not depend on the initial loads $l_0^k(\vec{P})$ for $k = 1, \dots, m$. For example, if $l_0^1(\vec{P})$ increases in one, then $\max_{i=0}^s \left\{ \sum_{k \in K} l_i^k(\vec{P}) \right\}$ and $\sum_{k \in K} \min_{i=0}^s \{l_i^k(\vec{P})\}$ increase one also, but the difference remains the same.

To generate an initial solution (not necessary a feasible solution) we apply a randomized greedy heuristic based on the *nearest neighbor heuristic* for the TSP. It extends a partial solution starting from a random node, and iteratively adding a new location until they are all in the solution. At each iteration a customer is added at the end of the partial path, try to obtain a feasible solution of good quality (i.e., with small length). To this end we evaluate all possible candidates. We introduce a random node between the nearest to the last node added and keeping the partial tour feasible. If it is not possible we introduce one of them minimizing the “infeasibility”.

Improvement Phase

Inspired in the work of Mladenovic et al. [6] for the 1-PDTSP, the improvement phase consists of a Variable Neighborhood Descendent (VND) with dif-

ferent neighborhood structures. The improvement phase starts directly from the initial solution generated by the first heuristic. In a local search algorithm a neighborhood structure is designed by introducing moves from one solution to another.

The neighborhood structures for the m -PDTSP tried for the improvement phase are:

- *2-opt*. The 2-opt procedure is the simplest method. It removes two arcs from the tour and reconnects the two paths created. Except for the orientation of the tour, there is only one way to reconnect the two paths and still have a valid tour.
- *Insertion*. The insertion is a special case of 3-opt. where each customer is inserted between any two other customers.
- *A complete 3-opt*. The complete 3-opt procedure removes three arcs from the tour. There are several ways to reconstruct a tour after deleting 3 arcs. This procedures checks all possible ways.
- *Double bridge*. It is a special case of 4-opt. It is very important for some combinatorial problems because it keeps the orientation of the tour.
- *MIP-based heuristic*. It solves a restricted version of the MIP model for the m -PDTSP. The restriction consists of fixing variables to get easier-to-solved model. More precisely. a given percentage of binary variables are fixed to 1 and another percentage are fixed to 0. The resultantly restricted MIP model is the solved using the branch-and-cut approach described in [11].

To check the feasibility of the new tour is can be done in $O(mn)$ time as is shown before. However, in order to do this more efficiently, we propose a *binary indexed tree* data structure. This data structure will allow us to check the feasibility of the neighborhood tour in $O(m \log n)$. If a better solution is fond we need to update a new binary indexed tree in $O(mn)$ time.

This work discusses which is the best combination of neighborhood structures in terms of solution quality and time consuming.

3 Conclusions

This paper proposes an heuristic approach to solve the multi-commodity Pickup-and-Delivery Traveling Salesman Problem. This problem is a very interesting and complex routing problem as it generalizes or is related to many other variants that have been addressed in the vehicle routing literature during recent years. It concerns the problem of finding a minimum-cost Hamiltonian

route for a capacitated vehicle that must collect and deliver several products, each one with possibly several origins and several destinations.

References

- [1] S. Anily, J. Bramel, Approximation algorithms for the capacitated traveling salesman problem with pickups and deliveries, *Naval Research Logistics* 46 (1999) 654–670.
- [2] C. Bordenave, M. Gendreau, G. Laporte, A branch-and-cut algorithm for the nonpreemptive swapping problem, *Naval Research Logistics* 56 (5) (2009) 478–486.
- [3] P. Chalasani, R. Motwani, Approximating capacitated routing and delivery problems, *SIAM Journal on Computing* 28 (1999) 2133–2149.
- [4] D. Chemla, F. Meunier, R. Wolfler Calvo, Bike hiring system: solving the rebalancing problem in the static case.
- [5] G. Erdogan, J. F. Cordeau, G. Laporte, A branch-and-cut algorithm for solving the non-preemptive capacitated swapping problem, *Discrete Applied Mathematics* 158 (15) (2010) 1599–1614.
- [6] N. Mladenovic, D. Uroševic, S. Hanafi, A. Ilic, A general variable neighborhood search for the one-commodity pickup-and-delivery travelling salesman problem, *European Journal of Operational Research* volume 220, issue 1, year 2012, pp. 270 - 285
- [7] H. Hernández-Pérez, I. Rodríguez-Martín, J. J. Salazar-González, A hybrid grasp/vnd heuristic for the one-commodity pickup-and-delivery traveling salesman problem, *Computers and Operations Research* 36 (5) (2009) 1639–1645.
- [8] H. Hernández-Pérez, J. J. Salazar-González, Heuristics for the one-commodity pickup-and-delivery traveling salesman problem, *Transportation Science* 38 (2) (2004) 245–255.
- [9] H. Hernández-Pérez, J. J. Salazar-González, The one-commodity pickup-and-delivery traveling salesman problem: inequalities and algorithms, *Networks* 50 (4) (2007) 258–272.
- [10] H. Hernández-Pérez, J. J. Salazar-González, The multi-commodity one-to-one pickup-and-delivery traveling salesman problem, *European Journal of Operational Research* 196 (3) (2009) 987–995.

- [11] H. Hernández-Pérez, J. J. Salazar-González, The multi-commodity pickup-and-delivery traveling salesman problem, Accepted in Networks (2012).
- [12] T. Raviv, M. Tzur, I. A. Forma, Static repositioning in a bike-sharing system: Models and solution approaches.
- [13] I. Rodríguez-Martín, J. J. Salazar-González, Hybrid heuristic approaches for the multi-commodity one-to-one pickup-and-delivery traveling salesman problem, Submitted (2011).
- [14] F. Zhao, S. Li, J. Sun, D. Mei, Genetic algorithm for the one-commodity pickup-and-delivery traveling salesman problem, Computers and Industrial Engineering 56 (4) (2009) 1642–1648.

Introducing a new station on a high-speed railway corridor competing with another transportation mode

Federico Perea ¹

*Departamento de Estadística e Investigación Operativa Aplicadas y Calidad
Universitat Politècnica de València
Valencia, Spain*

Juan A. Mesa²

*Departamento de Matemática Aplicada II
Universidad de Sevilla
Sevilla, Spain*

Gilbert Laporte³

*CIRRELT and Canada Research Chair in Distribution Management
HEC Montréal
Montréal, Canada*

Abstract

This paper introduces a mixed integer non-linear programming model for the problem of locating a new station on an already existing high-speed train corridor competing with another transportation mode. The new station is to be linked with one point of the existing network. The application of this model is illustrated over a toy example.

Keywords: Station location, railway network design, non-linear programming.

1 Introduction

Railway systems have many advantages with respect to other transportation modes. Among others we emphasize their safety, speed, stable travel times, and non-dependence on petrol. For these and other reasons large investments are being devoted to build or update railway networks. When designing such networks one not only has to take connectivity issues into account, but also competition with other transportation modes.

In this paper we propose a mixed integer non-linear programming model to find the optimal location of a new station on a high-speed train corridor and a point in an existing road network that is to be connected to the new station. The objective is to maximize the proportion of travelers that will use the new station by comparing the new travel times (assuming they have to use the new station) with the old travel times. Additionally, budget constraints are also imposed.

Building high-speed train stations relatively far from the nearest city, and link such stations with the road network is becoming a trend in certain high-speed train networks (see for instance the Spanish case at the moment in which this paper was written, in particular the set of cities that want connections with the station to be built in municipal district of *Villena* and its connection with the highway A3)

Traditional location-station models (see for instance [1], [2], [3]) do not take into account the competition with other transportation models. Recently, [4] introduce and solve a location-station model with a competing transportation mode. We now extend the models in the latter paper by allowing connections with any point of the road network and by using the Euclidean distance for such connection.

The rest of the paper is structured as follows. In Section 2 we introduce the input data for our problem and define the objective. Section 3 details the necessary variables and constraints to build the mixed integer non-linear programming model that solves our problem.

¹ Email: perea@eio.upv.es

² Email: jmesa@us.es

³ Email: gilbert.laporte@cirrelt.ca

2 Problem specifications

- (i) A set N^1 of sites, $N^1 = \{n^1, \dots, n^k\}$ with $n^i = (n_1^i, n_2^i) \in \mathbb{R}^2$, $i = 1, \dots, k$. Without loss of generality, assume $n^1 = (0, 0)$ and $n^k = (b, 0)$. For the sake or notation, we sometimes identify nodes with their indexes.
- (ii) A set of directed arcs A^1 joining pairs of elements N^1 (excluding (n^1, n^k) and (n^k, n^1)) is considered. In a practical setting, A^1 is the road network. For each arc a we denote by a_1 and a_2 the indexes of its initial and end node, respectively. Let E^1 be the set of (non-directed) edges obtained from A^1 , that is, $E^1 = \{a \in A^1 : a_1 < a_2\}$. Let $|E^1|$ denote the cardinality of this set. Note that $|E^1| \leq \binom{k}{2} - 1 = \frac{k(k-1)}{2} - 1$.
- (iii) Let A^2 be the two directed arcs joining the original high-speed train stations, that is, $A^2 = \{(n^1, n^k), (n^k, n^1)\}$.
- (iv) Let $N^2 = \{n^{k+1} = (x_{st}, 0)\}$ be the new station to be built, where $x_{st} \in (0, b)$ is a variable of the problem.
- (v) Let $N^3 = \{n^{k+2}, \dots, n^{k+|E^1|+1}\}$ be intermediate points at each of the edges in E^1 , which are to be determined. One of these points will be linked with the new station to be built, n^{k+1} . Therefore, assuming that the elements in E^1 can be approximated by straight segments, for each $i = k+2, \dots, k+|E^1|+1$, there exists $(n^j, n^{j'}) \in E^1$ and $\lambda \in [0, 1]$ such that $n^i = \lambda n^j + (1 - \lambda)n^{j'}$. In non straight-segment cases, other parameterizations may apply. Note that λ is a variable of the problem. For each $n^i \in N^3$, let na_1^i (na_2^i) $\in \{1, \dots, k\}$ denote the initial (end) node of the arc in E^1 to which n^i is intermediate.
- (vi) Let A^3 be the directed arcs joining the extreme points of each edge in E^1 with their corresponding intermediate points. Therefore $|A^3| = 4|E^1|$.
- (vii) Let A^4 be the set of directed arcs joining the old stations (n^1 and n^k) with the new one (n^{k+1}). Therefore $|A^4| = 4$.
- (viii) Let A^5 be the set of directed arcs joining the new station with the intermediate points N^3 . Therefore the cardinality of A^5 is $2|E^1|$. Define E^5 as the set of (non-directed) edges obtained from A^5 .
- (ix) For each arc $a \in A^1 \cup A^2$, there is a parameter d_a which is the necessary time to traverse the corresponding arc. We assume that speed using the high-speed corridor is faster than the road network, given by parameter $\alpha \in (0, 1)$. Distances for the arcs in (A^3, A^4, A^5) will be determined in the problem, and are calculated as the Euclidean distances between the corresponding endpoints. Distances for arcs in A^4 belong to the high-

speed train network, and are therefore multiplied times the speed factor.

- (x) The construction costs of arcs in A^3 and A^4 are not considered, because the old infrastructure will be used. Nevertheless, the construction cost of arcs joining the new station with the chosen intermediate point (those in A^5) will be considered, and is assumed to be proportional to its Euclidean distance. Construction costs for the new station and its connection with the road network are also considered, and parameterized by $c(x_{st})$ and $c(a)$. A bound C_{max} on the available budget is also given.
- (xi) The set of origin-destination (O/D) pairs is $W = \{(n^i, n^j) : i, j = 1, \dots, k\}$. The number of potential trips between nodes in N^1 is given by parameter g_w , for all $w \in W$. The traveling time for the O/D pair w without using the new station is also known and equal to u_w^{PREV} . We will denote by w_1 and w_2 the indexes of the initial and end node of O/D pair w , respectively.
- (xii) We assume that, if a passenger goes from n^1 to n^k (or from n^k to n^1), they will suffer a stop time at the new station to be built n^{k+1} , which is fixed and equal to β .

Example 2.1 Consider the following toy example to illustrate our problem. There is a high-speed train corridor between city $n^1 = (0, 0)$ and city $n^3 = (2, 0)$ and a third city n^2 located on $(1, 1)$. Edges (n^1, n^2) and (n^2, n^3) belong to the road network. All these edges can be used in both directions, see Figure 1, left network.

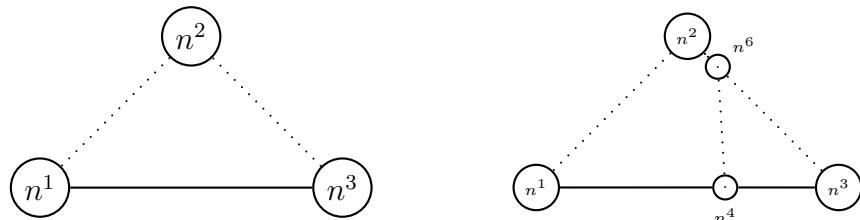


Fig. 1. Left hand side. Test Network. Dotted arcs belong to the road network and dashed arcs belong to the high-speed network. Right hand side. Solution to example. Dotted arcs belong to the road network, and dashed arcs belong to the high-speed train network. The new high-speed train station is located at n^4 . The connection between n^4 and the road network is located at n^6 .

The travel times between nodes in the current network using the roads were considered equal to their Euclidean distances. The high-speed factor considered was $\alpha = 0.25$. Therefore $d_{a=(n^1, n^2)} = d_{a=(n^2, n^3)} = \sqrt{2}$, $d_{a=(n^1, n^3)} = 0.5$. We assume that $g_{w_1} = 9$ and $g_{w_2} = 18$, with $w_1 = (1, 2)$ and $w_2 = (2, 3)$.

No other O/D pairs are considered in this example. The construction costs for the new station are constant and equal to two monetary units, that is, $c_{x_{st}} = 2 \forall x_{st} \in (0, 2)$. The construction costs for the new link is assumed to be proportional to its Euclidean length so that one length unit costs one monetary unit. The maximum budget is set to 2.8 monetary units. Note that two new potential nodes are considered: n^5 which is an intermediate point of arc (n^1, n^2) and n^6 , which is an intermediate point of (n^2, n^3) . In order to illustrate our notation, we will define for this example each of the defined sets: $N^1 = \{n^1, n^2, n^3\}$, $N^2 = \{n^4\}$, $N^3 = \{n^5, n^6\}$, $A^1 = \{(n^1, n^2), (n^2, n^1), (n^2, n^3), (n^3, n^2)\}$, $A^2 = \{(n^1, n^3), (n^3, n^1)\}$, $A^3 = \{(n^1, n^5), (n^5, n^1), (n^2, n^5), (n^5, n^2), (n^2, n^6), (n^6, n^2), (n^3, n^6), (n^6, n^3)\}$, $A^4 = \{(n^1, n^4), (n^4, n^1), (n^3, n^4), (n^4, n^3)\}$, $A^5 = \{(n^4, n^5), (n^5, n^4), (n^4, n^6), (n^6, n^4)\}$. The sets E are naturally defined from the sets A .

The problem then consists of choosing a location for the new station (node in N^2), and one intermediate node in N^3 , so that the objective function is maximized and the budget constraint is not violated. The objective is to maximize the number of travelers that will use the new station. This number will be estimated from a Logit function. Therefore, if u_w denotes the traveling time of O/D pair w when they use the new station, we assume that the proportion of travelers in w that will use this new station is given by:

$$\varphi(x) = \frac{1}{1 + e^{-(u_w^{PREV} - u_w)}}.$$

Note that, if $u_w^{PREV} - u_w$ is sufficiently large, then $\varphi(u_w^{PREV} - u_w)$ tends to one. Conversely, if $u_w^{PREV} - u_w$ is sufficiently small, $\varphi(u_w^{PREV} - u_w)$ tends to zero. Finally, if $u_w = u_w^{PREV}$, $\varphi(u_w^{PREV} - u_w)$ is 0.5. Other properties of φ is that it is monotone increasing. Unfortunately, this function is neither convex nor concave, which will make our problem harder to solve. Note that, in order to avoid illogical results, this function might be considered zero or one in its left tail and right tail, respectively.

3 The model

The objective is to find a location for the new station to be built and a point in the network (N^1, A^1) that attracts as many potential travelers as possible to using the new station to be built. Such problem will be modeled as a mixed integer non linear programming (MINLP) problem with the following variables:

- (i) $x_{st} \in (0, b)$ says where to locate the new station.
- (ii) $\lambda \in [0, 1]$ will decide where to locate the intermediate point.
- (iii) For each $n^i \in N^3$, x_a^i and y_a^i denote the first and second coordinate of the intermediate point n^i .
- (iv) Binary variable v_a says whether or not arc $a \in A^5$ (links between the new station and the intermediate points of arcs in A^1) is to be built.
- (v) d_a^3, d_a^4 and d_a^5 measure the length of arcs in A^3, A^4 and A^5 , respectively.
- (vi) Binary variable f_a^w says whether or not O/D pair w uses arc a when they are forced to use the new station to be built.
- (vii) Binary variable t_w says whether or not O/D pair w suffers a stop time at the new station to be built.
- (viii) u_w is the time needed for O/D pair w using the new station.
- (ix) The expected number of travelers using the new station is modeled by variable z .

Some of the variables are explicitly defined:

$$\begin{aligned}
x_a^i &= \lambda n_1^j + (1 - \lambda)n_1^{j'}, \quad \forall i \in N^3 : na_1^i = j, na_2^i = j'. \\
y_a^i &= \lambda n_2^j + (1 - \lambda)n_2^{j'}, \quad \forall i \in N^3 : na_1^i = j, na_2^i = j'. \\
d_a^3 &= \sqrt{(n_1^j - x_a^j)^2 + (n_2^j - y_a^j)^2}, \\
&\forall a \in A^1, j \in N^1, i \in N^3 : a = (i, j) \text{ or } a = (j, i). \\
d_a^4 &= \alpha \sqrt{(n_1^i - x_{st})^2 + (n_2^i - 0)^2}, \\
&\forall a \in A^4, i \in N^1, j = k + 1, a = (i, j) \text{ or } a = (j, i). \\
d_a^5 &= \sqrt{(x_a^i - x_{st})^2 + (y_a^i - 0)^2}, \quad \forall a \in A^5, i \in N^3 : a_1 = i \text{ or } a_2 = i. \\
u_w &= \sum_{a \in A^1 \cup A^2} d_a f_a^w + \sum_{a \in A^3} d_a^3 f_a^w + \sum_{a \in A^4} d_a^4 f_a^w + \sum_{a \in A^5} d_a^5 f_a^w + t_w \beta, \quad \forall w \in W.
\end{aligned}$$

From these variables and definitions, our objective function and constraints can be defined. The objective is

$$(1) \quad \max z = \sum_w g_w \frac{1}{1 + e^{-(u_w^{REV} - u_w)}},$$

subject to the following constraints:

$$(2) \quad c(x_{st}) + \sum_{a \in A^5 : a_1 < a_2} v_a c(a) \leq C_{max}.$$

- $$(3) \quad \sum_{a \in A^5 : a_1 < a_2} v_a \leq 1.$$
- $$(4) \quad \sum_{a : a_2 = w_1} f_a^w = 0, \forall w \in W.$$
- $$(5) \quad \sum_{a : a_2 = w_2} f_a^w = 1, \forall w \in W.$$
- $$(6) \quad \sum_{a : a_1 = w_1} f_a^w = 1, \forall w \in W.$$
- $$(7) \quad \sum_{a : a_1 = w_2} f_a^w = 0, \forall w \in W.$$
- $$(8) \quad \sum_{a : a_1 = r} f_a^w - \sum_{a : a_2 = r} f_a^w = 0, \forall w \in W, r \in N^1 \cup N^2 \cup N^3 : w_1 \neq r, w_2 \neq r.$$
- $$(9) \quad \sum_{a \in A^4 \cup A^5} f_a^w \geq 1, \forall w \in W.$$
- $$(10) \quad f_a^w \leq v_a, \forall a \in A^5, w \in W.$$
- $$(11) \quad f_{a'}^w \leq v_a, \forall a, a' \in A^5, w \in W : a_1 = a'_2, a_2 = a'_1.$$
- $$(12) \quad (\sum_{a \in A^4} f_a^w) - 1 \leq Mt_w, \forall w \in W.$$

Constraints (2) ensure that the maximum budget is not exceeded. Constraints (3) ensure that at most one connection between the road network and the new station to be built is done. Note that the right hand side of this constraint can be modified so that at most m connections are built. (4) to (8) are flow conservation constraints on the f variables. Constraints (9) force that the new station is used in order to calculate the time spent using such new station. Constraints (10) and (11) say that, if an arc joining the new station with one of the intermediate points is used by at least one O/D pair, then such arc must be built. Constraints (12) make $t_w = 1$ if two arcs in A^4 are used by the O/D pair w . M is a sufficiently large positive number.

Example 3.1 With the data in Example , the corresponding MINLP model has 73 variables (48 of which are binary) and 53 constraints. It is modeled and solved in GAMS 23.0 using CONOPT 3.14 and CPLEX 11.2.1 obtaining the following solution in less than 2 seconds:

The new station is located at $(1.251, 0)$, linked using a straight segment with point $n^6 = (1.202, 0.798)$, which is located on edge (n^2, n^3) (see Figure 1, right hand side.) Note that n^5 is not part of the solution network. Using the Logit function described in Section 3, with this solution it is expected that

50.1% of w^1 and 53.2% of w^2 will use the new station. When using the new station, the path followed by w^1 is $n^1 - n^4 - n^6 - n^2$, whereas the path followed by w^2 is $n^2 - n^6 - n^4 - n^3$. With this solution, $u_{w^1} = 1.398$ and $u_{w^2} = 1.272$. Note that these travel times are lower than the original ones ($\sqrt{2} \simeq 1.414$). This is why the expected proportion of travelers that will use the new station is, in both cases, larger than 50%.

4 Conclusions

In this paper we have introduced a MINLP model to locate and join with an already existing road network a new station on a high-speed train corridor. Such location maximizes the proportion of travelers for which the use of the new station is attractive, which is measured by a Logit function. Further research on this topic will focus on efficient procedures (possibly heuristics) to solve the proposed problem, because the introduced MINLP is expected to computationally explode for realistic instances.

Acknowledgments

This work was partially supported by Ministerio de Educación, Ciencia e Innovación (Spain) under project MTM2009-14243 (FEDER), and under project MTM2012-37048 and by Junta de Andalucía (Spain)/FEDER under excellence projects P09-TEP-5022 and FQM-5849. Special thanks are due to one anonymous referee for his/her valuable comments and advice.

References

- [1] H. Hamacher, A. Liebers, A. Schöbel, D. Wagner, F. Wagner, *Locating New Stops in a Railway Network*. Electronic Notes in Theoretical Computer Science, 50(1), 2001.
- [2] G. Laporte, J.A. Mesa, F.A. Ortega, *Locating Stations on Rapid Transit Lines*. Computers & Operations Research, 29, 741–759, 2002.
- [3] A. Schöbel, *Locating Stops Along Bus or Railway Lines-A Bicriteria Problem*. Annals of Operations Research, 136, 211–227, 2005.
- [4] M.-C. Körner, J.A. Mesa, F. Perea, A. Schöbel, D. Scholz, *A Maximum Trip Covering Location Problem with an Alternative Mode of Transportation on Tree Networks and Segments*. TOP, DOI 10.1007/s11750-012-0251-y, 2012.

Disjunctive combinatorial branch in a subgradient tree algorithm for the DCMST problem with VNS-Lagrangian bounds

Rafael Castro de Andrade¹

*Department of Statistics and Applied Mathematics, Federal University of Ceará,
Campus do Pici, BL 910, CEP 60.455-760 - Fortaleza, Ceará, Brazil.*

Adriano Tavares de Freitas²

*Department of Computer Science, Federal University of Ceará, Campus do Pici,
BL 910, CEP 60.455-760 - Fortaleza, Ceará, Brazil.*

Abstract

Consider an undirected graph $G = (V, E)$ with a set of nodes V and a set of weighted edges E . The weight of an edge $e \in E$ is denoted by $c_e \geq 0$. The degree constrained minimum spanning tree (DCMST) problem consists in finding a minimum spanning tree of G subject to maximum degree constraints $d_v \in \mathbb{N}$ on the number of edges connected to each node $v \in V$. We propose a Variable Neighborhood Search (VNS)-Lagrangian heuristic that outperforms the best known heuristics in the literature for this problem. We use an exact subgradient tree (SGT) algorithm in order to prove solution optimality. The SGT algorithm uses a combinatorial relaxation to evaluate lower bounds on each relaxed solution. Thus, we propose a new branching scheme that separates an integer relaxed solution from the domain of spanning trees while generating new disjoint SGT-node partitions. We prove optimality for many benchmark instances and improve lower and upper bounds for the instances whose optimal solutions remain unknown.

Keywords: subgradient tree, DCMST, VNS-Lagrangian heuristic.

1 Introduction

Let $G = (V, E)$ be an undirected graph, where V is the set of nodes and E is the set of weighted edges, with the weight of an edge $e \in E$ being denoted by $c_e \geq 0$. The degree constrained minimum spanning tree (DCMST) problem consists in finding a minimum spanning tree of G subject to a maximum degree constraint $d_v \in \mathbb{N}$ on the number of edges connected to each node $v \in V$. We know that when $d_v = 2$, for all $v \in V$, this problem is equivalent to find a minimum weight Hamiltonian path in G , which is NP-hard [5].

The reader is referred to [7], [2], [8], [1], [3] and [4] for a selective literature review on applications and solution approaches for this problem.

The contribution of this work is twofold. First, we develop a VNS-Lagrangian heuristic based on a dynamic variable neighborhood descent (VND) method. The core of our algorithm is the Lagrangian heuristic in [1] where we introduce new dynamic local search techniques to overcome local optima solutions, adapted from the VNS/VND techniques from [9]. Second, we propose an exact subgradient tree (SGT) method which uses a combinatorial relaxation of the problem obtained by dropping the degree constraints and incorporating them into the objective function of the relaxed problem by means of Lagrange multipliers. The relaxed subproblems in the SGT are solved by using a subgradient optimization algorithm [6]. As the solution of a given subproblem is integer (a spanning tree possibly violating the degree constraints of the original problem), we propose a new and novel branching technique for partitioning the space of feasible (for the subproblem) solutions in disjoint regions of spanning trees while separating any relaxed node solution. Observe that a classic branching by dichotomy for this problem means fixing some edge of the relaxed solution (a spanning tree) to zero or to one to create two new partitions, which is known to be very inefficient. The idea we use here is quite general and can be extended to any divide-and-conquer approach that uses a combinatorial relaxation. Our SGT approach is competitive with the one in [4] and the VNS-Lagrangian heuristic outperforms all the best known heuristics for the DCMST problem.

We present the problem formulation in the next section. Sections 3 and 4 present the solution approaches. Computational results are presented in

¹ Email: rca@lia.ufc.br

² Email: adrianoblue@gmail.com

Section 5. In Section 6 we present a brief conclusion.

2 Problem formulation

In model (P) below let $E(S) \subseteq E$, for $S \subseteq V$, represent the set of edges with both extremities in S . Let $\delta(v) \subseteq E$, for $v \in V$, represent the set of edges adjacent to v in G . Let x_e , for all $e \in E$, be binary variables representing if an edge e belongs ($x_e = 1$) or not ($x_e = 0$) to a minimum degree constrained spanning tree of G . The problem formulation of [1] is

$$(1) \quad (P) \quad \min \sum_{e \in E} c_e x_e$$

$$(2) \quad \text{s.t. } \sum_{e \in E} x_e = |V| - 1,$$

$$(3) \quad \sum_{e \in E(S)} x_e \leq |S| - 1, \quad S \subset V,$$

$$(4) \quad \sum_{e \in \delta(i)} x_e \leq d_i, \quad \forall i \in V.$$

$$(5) \quad x_e \in \{0, 1\} \quad \forall e \in E.$$

Associating Lagrange multipliers $\lambda \in \mathbb{R}_+^{|V|}$ with constraints (4) we obtain the relaxed problem

$$(6) \quad (R) \quad \min \sum_{e=(i,j) \in E} (c_e + \lambda_i + \lambda_j)x_e - \sum_{i \in V} \lambda_i d_i$$

s.t. (2), (3), (5).

Problem (R) gives lower bounds on the solution of (P) for a given $\lambda \geq 0$. In [1] the reader can find a subgradient optimization procedure to obtain lower bounds on the solution of (P) .

Now consider the Lagrangian relaxation (R_F) of a non empty restriction (P_F) of (P) below, where we fix some variables $x_e = 1$ for $e \in F_1$ and $x_e = 0$ for $e \in F_0$, with $F_1 \cap F_0 = \emptyset$ and $F_1, F_0 \subset E$, $|F_1| < |V| - 1$. Define primal multipliers $\beta_v = 0$ if $\sum_{u \in \delta(v) \cap F_1} x_{uv} < d_v$ and $\beta_v = \infty$, otherwise.

$$(7) \quad (R_F) \quad \min \sum_{e=(i,j) \in E} (c_e + \lambda_i + \lambda_j + \beta_i + \beta_j)x_e - \sum_{i \in V} \lambda_i d_i$$

*s.t. $x_e = 1, \forall e \in F_1, \quad x_e = 0, \forall e \in F_0$
(2), (3), (5),*

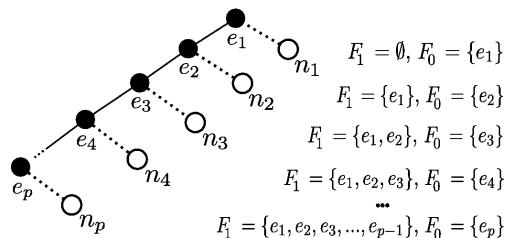
Proposition 2.1 *For a given fixing of variables as above, the solution value of (R_F) is a lower bound for the restricted version (P_F) of (P) . Moreover, it is at least equal to the value obtained by solving (R_F) without considering the primal multipliers β .*

The proof of Proposition 2.1 will appear in a full version of this paper. We explore it in the exact SGT scheme presented hereafter. The idea is to avoid that some non fixed edges incident to a saturated node appear in any solution of (R_F) . When considered during the branching process of the SGT algorithm, Proposition 2.1 reduces considerably the size of the set F_0 of edges that will not belong to any relaxed solution associated to a given partition of the space of feasible spanning trees. If the solution of (R_F) presents some node $u \in V$ that is not saturated by the edges in F_1 , we can divide that partition (associated to the fixed edges F_0 and F_1) into smaller ones as follows.

Proposition 2.2 *Let v be a node violating the degree constraint d_v in a Lagrangian relaxed solution (a tree T_P) related to a given partition P associated to F_0 and F_1 . Consider the p edges incident to v in T_P in the order they appear in the list $L_v = \langle e_1, e_2, \dots, e_p \rangle$. The partitions $P \cap \{x | x_{e_1} = 0\}$ and $P \cap P_j$, for each $j \in \{2, \dots, p\}$, where $P_j = \{x | x_{e_i} = 1, i = 1, \dots, j-1, x_{e_j} = 0\}$, are disjoint. Moreover, the partitions $P \cap P_j$, for $j \geq d_v + 1$ are infeasible for the original problem and can be discarded by an exact algorithm for the DCMST problem.*

Figure 1 illustrates the partitioning process of Proposition 2.2. Dotted lines represent that an edge is fixed at zero. Dashed lines represent that an edge is fixed at one. In this figure we assume $F_0 = F_1 = \emptyset$ for the root partition (F_0 and F_1 , for each child partition, are the sets of edges fixed at zero and at one, respectively). All possible partitions n_1, \dots, n_p are represented and some of them can be infeasible for the original problem (the ones with $|F_1| > d_v$).

Figure 1. Disjunctive combinatorial branch: the new disjoint partitions are the leaves in this tree representation.



This partitioning scheme is quite general and can be applied to exact al-

gorithms where the employed relaxation is a combinatorial problem (i.e. has an integer relaxed solution).

3 VNS-Lagrangian heuristic

The idea is to improve the Lagrangian heuristic in [1], with VNS/VND techniques from [9] to overcome local optima solutions of the Lagrangian procedure. First, we relax the degree constraints of the problem and incorporate them to the objective function by using Lagrange multipliers. A subgradient optimization procedure is used to obtain lower bounds on the problem solution. The Lagrange multipliers are used to perturb the original edge costs in order to obtain feasible degree constrained trees in the graph with modified Lagrange edge costs. For such, we use an adapted version of the Kruskal algorithm to deal with the degree constraints. The incumbent solution of this phase is submitted to a VNS procedure. Each iteration of the VNS is composed of a shaking phase, where a solution T' is obtained from an incumbent solution T of a neighborhood structure $N_1^{VNS}, \dots, N_{k_{max}}^{VNS}$, where $N_k^{VNS}(T)$ is the set of feasible trees which are neighbors to T having exactly $k \leq k_{max}$ different edges, with k_{max} given. In a second phase, we try to improve T' with a Dynamic VND procedure based on three neighborhood structures $N_{VND}^{(1)}, N_{VND}^{(2)}$ and $N_{VND}^{(3)}$ of a feasible solution T . For this, we classify the edges which are not in T into three groups: $E0$ of edges with no saturated extremity; $E1$ of edges with exactly one saturated extremity and $E2$ of edges with both saturated extremities, with $E(G) = E(T) \cup E0 \cup E1 \cup E2$. An edge exchange is defined by a pair of edges (e, \bar{e}) , where $e \in E(T)$ and $\bar{e} \in E(G) \setminus E(T)$.

Neighborhood $N_{VND}^{(1)}(T)$ is the set of feasible trees which can be obtained from T by applying one edge exchange (e, \bar{e}) . Basically, an edge $\bar{e} \in (E0 \cup E1)$ is inserted in T and a cycle C is generated. We remove and edge $e \in E(T)$ from C in order to reestablish feasibility of the solution.

In neighborhood $N_{VND}^{(2)}(T)$ we use two edge exchanges: (e_1, \bar{e}_1) and (e_2, \bar{e}_2) . The insertion of an edge $\bar{e}_1 \in (E1 \cup E2)$ generates a cycle. We remove the edge $e_1 \in E(T)$ from this cycle. The second exchange is used if we need to arrange the degree of a node v which remains violated after the first edge exchange.

Neighborhood $N_{VND}^{(3)}(T)$ uses three edge exchanges: (e_1, \bar{e}_1) , (e_2, \bar{e}_2) and (e_3, \bar{e}_3) . First, we insert an edge $\bar{e}_1 \in E2$ and a cycle is generated. An edge $e_1 \in E(T)$ is removed from this cycle. After the edge exchange (e_1, \bar{e}_1) , two nodes v_1 and v_2 may remain violated. The remaining two edge exchanges are used to arrange the degrees of v_1 and v_2 .

Note that for DCMST instances with degree constraints $d_v = 2$, for all $v \in V$, the three neighborhoods above have no effect for improving feasible solutions. To overcome such situations, we propose redefining the sets $E0$, $E1$ and $E2$ dynamically based on the current tree after each edge exchange.

In the last phase of each VNS iteration, absorption functions (see e.g. [9]) allow actualizing the incumbent solution. In this process, we can move to a feasible tree with worse cost in order to overcome local optima solutions.

The best incumbent solution is used as a cut-off value in the SGT method.

4 SGT scheme

The SGT is a branch-and-bound search tree algorithm composed of three operations: evaluation, pruning and branching of SGT nodes. In the evaluation process we have two types of lower bounds on each node solution - one given by the value of the Lagrangian relaxed solution (with edge costs perturbed by Lagrange multipliers) and the other by the original edge costs associated to that solution. We determine sequences of Lagrange multipliers in a tree structure following the generation of SGT nodes. These sequences are updated from a father to a child node by taking into account the fixing of the edges and the direction of the gradient obtained at the node relaxed solution. Internal to each SGT subproblem, we iterate the subgradient procedure a certain number of iterations to improve its relaxed bound. SGT nodes are organized in the search tree according to the best bound strategy. The pruning operation allows deleting SGT nodes from the search tree when the bound on the node solution is greater than the value of the best incumbent solution for the problem. In the branching process, a given partition (father node) is divided into new smaller and disjointed ones (children nodes). The node we choose to perform the branch is the one with smaller violation of its degree constraint. The novelty here is how we separate a feasible tree from the domain of spanning trees to obtain new disjoint subproblems partitions according to the Proposition 2.2. Observe that we do not use linear relaxation in our work (a combinatorial one is used instead). Moreover, when some edges are fixed at one (thus they must be in the node solution), it is possible that some node v becomes saturated (the number of edges connected to v is equal to d_v). In this case, all remaining unfixed edges incident to v can be fixed at zero. However, saving the information of the edges fixed at zero for each SGT node is not practical since the required memory increases very fast. To overcome this problem, we introduce the concept of primal multipliers in the Proposition 2.1 that are also used to perturb the Lagrange edge costs in order

to prevent a known saturated node of having additional edges being incident to it in the relaxed SGT node solution.

5 Computational Results

The C++ algorithms are linked with g++-4.4.3 for Linux Ubuntu in a Core 2 Duo PC with 2.4 GHz / 3GB RAM. In this short paper we report only results for two classes of benchmark instances and compare our results with the ones in [9] and [4]. A more extensive set of experiments including Hamiltonian [1], DE and DR [4] instances will be reported in a complete version of this paper to show the efficiency of our solution approach. The legend in the next tables is self-explained. *CPU* is in (*min : sec*) and *gap* = $100(UB - LB)/LB$.

Instance V - <i>(id)</i>	Results from [9]			VNS-Lagr-Heur				+SGT for gap > 0				Results from [4]			
	UB	cpu	LB	UB	cpu	gap	LB	UB	cpu	gap	LB	UB	cpu	gap	
100_1	3790	0:00	3790	3790	0:00	0.000					3790	3790	0:00	0.000	
100_2	3829	0:00	3829	3829	0:00	0.000					3829	3829	0:00	0.000	
100_3	3916	0:00	3916	3916	0:01	0.000					3916	3916	0:00	0.000	
200_1	5316	0:06	5316	5316	0:00	0.000					5316	5316	0:00	0.000	
200_2	5651	0:08	5647	5647	0:01	0.000					5647	5647	0:02	0.000	
200_3	5699	0:06	5698	5698	0:01	0.000					5698	5698	0:01	0.000	
300_1	6477	0:50	6475	6477	0:11	0.030	6477	6477	0:12	0.000	6477	6477	0:02	0.000	
300_2	6807	0:55	6803	6809	0:48	0.088	6807	6807	20:49	0.000	6805	6807	0:31	0.029	
300_3	6430	0:40	6430	6430	0:01	0.000					6430	6430	0:04	0.000	
400_1	7414	3:01	7414	7414	0:01	0.000					7414	7414	0:02	0.000	
400_2	7783	2:33	7776	7783	1:43	0.090	7782	7782	26:55	0.000	7779	7782	0:32	0.038	
400_3	7604	3:19	7602	7604	1:24	0.026	7604	7604	1:33	0.000	7603	7604	0:38	0.013	
500_1	8272	9:12	8272	8272	0:03	0.000					8272	8272	0:06	0.000	
500_2	8396	10:56	8392	8392	2:01	0.000					8392	8392	0:18	0.000	
500_3	8502	8:24	8502	8502	2:07	0.000					8502	8502	0:26	0.000	
600_1	9037	10:19	9037	9037	0:01	0.000					9037	9037	0:09	0.000	
700_1	9786	16:37	9786	9786	0:02	0.000					9786	9786	0:13	0.000	
800_1	10333	29:21	10333	10333	0:07	0.000					10333	10333	0:25	0.000	
900_1	10918	36:00	10918	10918	0:07	0.000					10918	10918	0:29	0.000	
1000_1	11408	36:05	11407	11408	2:58	0.008	11407	11408	*	0.008	11407	11408	2:48	0.008	
2000_1	15670	180:50	15670	15670	12:23	0.000					15670	15670	4:32	0.000	
2000_2	16255	208:06	16239	16243	59:03	0.024	16239	16242	*	0.018	16242	16242	45:12	0.000	
2000_3	16687	225:29	16668	16670	54:18	0.011	16668	16670	*	0.011	16669	16670	68:56	0.005	
2000_4	16445	34:36	16368	16370	64:12	0.012	16368	16370	*	0.012	16369	16370	70:32	0.006	
2000_5	16532	193:27	16520	16523	37:54	0.018	16520	16521	220:12	0.006	16520	16521	47:57	0.006	

(* Time limit exceeded (1 day)

Table 1
Computational results for Euclidean instances from [1]

In Table 5 we see that the VNS-Lagrangian Heuristic module outperforms the one in [9]. The SGT module acts when our heuristic gap is not null. It obtains the same upper bounds of [4] and proves optimality for three new instances. The SGT lower bounds are competitive with the ones of the non delayed relax and cut exact algorithm of [4].

6 Conclusions

We propose a new VNS-Lagrangian Heuristic and an exact SGT algorithm for the DCMST problem that are competitive with the best known algorithms for this problem. The SGT uses a novel partitioning technique that can be extended to other exact methods exploring combinatorial relaxations for the subproblems. We show how to strengthen the subproblems relaxed solution with the use of the primal multipliers. To the best of our knowledge, it is the first time these techniques are proposed for solving a NP-hard problem and they showed to be very efficient.

References

- [1] Andrade, R., A. Lucena and N. Maculan, *Using Lagrangian dual information to generate degree constrained spanning trees*, Discrete Applied Mathematics **154** (2006), pp. 703–717.
- [2] Caccetta, L. and S. Hill, *A branch and cut method for the degree-constrained minimum spanning tree problem*, Networks **37** (2001), pp. 74–83.
- [3] Cunha, A. S. and A. Lucena, *Lower and upper bounds for the degree-constrained minimum spanning tree problem*, Networks **50** (2007), pp. 55–66.
- [4] Cunha, A. S. and A. Lucena, *A hybrid relax-and-cut/branch and cut algorithm for the degree-constrained minimum spanning tree problem*, Technical report, Departamento de Administração, Universidade Federal do Rio de Janeiro (2008).
- [5] Garey, M. and D. Johnson, “Computers and Intractability, A Guide to the Theory of NP-Completeness,” Freeman, 1979.
- [6] Held, M., P. Wolfe and H. Crowder, *Validation of subgradient optimization*, Math. Programming **6** (1974), pp. 62–88.
- [7] Krishnamoorthy, M., A. Ernst and Y. Sharaiha, *Comparison of algorithms for the degree constrained minimum spanning tree*, Journal of Heuristics **7** (2001), pp. 587–611.
- [8] Ribeiro, C. and M. Souza, *Variable neighborhood search for the degree-constrained minimum spanning tree problem*, Discrete Appl. Math. **118** (2002), pp. 43–54.
- [9] Souza, M. and P. Martins, *Skewed vns enclosing second order algorithm for the degree constrained minimum spanning tree problem*, European Journal of Oper. Res. **191** (2008), pp. 677–690.

Heuristics for solving a capacitated multiple allocation hub location problem with application in German wagonload traffic

Julia Sender^{1,2}

*Institute of Transport Logistics
TU Dortmund
Dortmund, Germany*

Uwe Clausen³

*Fraunhofer Institute for Material Flow and Logistics
& Institute of Transport Logistics
TU Dortmund
Dortmund, Germany*

Abstract

In this paper, we present a capacitated multiple allocation hub location problem, which arose from a network design problem in German wagonload traffic. We develop heuristic solution approaches based on local improvements. We solve the problem with the heuristics and CPLEX on test data sets provided by our partner Deutsche Bahn AG. The computational results are presented and compared.

Keywords: hub location problems, IP, local search, logistics, network design

1 Introduction

Wagonload traffic is a specific production form in railway logistics, where the traffic volume (consisting of single wagons or small wagon groups) does not justify an entire direct train due to high fixed costs for running a train (e.g., track fees, personnel and traction costs). In order to exploit the effects of consolidation and economies of scale, the wagons are re-classified (bundled) in several formation yards with wagons of different origins and destinations. Although, the consolidation may lead to lower transportation costs, the reclassification of wagons in yards increases costs due to establishing and operating yards. Due to the strategic dimension of the network design, it has a great influence on future quality, production, and network costs of wagonload traffic. In order to route the origin-destination demand through the network, hub location problems take the location of hubs and the allocation of nodes into account. Thus, they decide about the structure of hub-and-spoke networks. Current reviews on hub location problems are given, e.g., in [1,2,3]. Most hub location problems can be divided into single and multiple allocation variants. In single allocation each non-hub node is allocated to a single hub node, whereas in multiple allocation each non-hub node may be allocated to multiple hub nodes. Multiple allocation versions increase the flexibility and complexity, but are expected to have lower costs. In most hub location problems the triangle inequality for costs or distances is assumed. Indeed, in wagonload traffic the costs per km may vary for different tracks. Since we consider a strategic problem, we introduce several capacity levels for dimensioning hubs. In wagonload traffic a shunting track in a yard is used to classify trains for a particular direction. Hence, the number of outgoing connections (directions) may also be limited for each yard. Thus, the considered capacitated multiple allocation hub location model differs in some ways from existing hub location problems. A similar model is also presented more in detail in our work [5], where we presented several modifications of the model in order to reduce the size and the complexity. These modifications include, e.g., the removal of redundant constraints, aggregation of constraints or adding of valid inequalities. However, the modifications done there have only small effects and therefore, are not sufficient to solve real-sized data sets with (commercial) solver, like CPLEX. In this paper, we develop heuristic solution approaches based on local

¹ We thank the German Federal Ministry of Education and Research (BMBF) for supporting this research (03MS640B).

² Email: julia.sender@tu-dortmund.de

³ Email: uwe.clausen@iml.fraunhofer.de

improvements for the original model.

In section 2, we present the optimization model. Solutions approaches for solving this model are developed in section 3. In section 4, computational results are given. We end with a short conclusion in section 5.

2 Optimization Model

Let N be the set of origin-destination nodes and potential hub nodes and let $i, j, k, l \in N$. The demand of origin-destination pair (i, j) is given by a number of wagons W_{ij} . Since the network structure is mainly determined by the number of wagons which have to be consolidated in yards, the data set given does not contain origin-destination demand where an entire train, that is a direct connection, would be provided. Let $\Gamma_k = \{1, \dots, s_k\}$ be the set of different capacity levels of hub k . The costs, capacity, and maximum number of outgoing connections of a hub k with level q ($q \in \Gamma_k$) are given by c_k^q , γ_k^q , and b_k^q . The costs per wagon on arc (k, l) are represented by c_{kl} . We consider a discount factor β for inter-hub transportation.

We introduce three different kinds of integer flow variables for collection, transfer, and distribution movements. The collection variables X_{ik} represent the flow from origin i to hub k . Transfer movements from hub k to hub l , where the flow originates at origin i , are given by Y_{kl}^i . The distribution variables Z_{lj}^i represent the flow from hub l to destination j originating at origin i . Moreover, we use binary variables for locating hubs and capacity levels. The location variable H_k is one if node k is a hub and zero otherwise. If node k is a hub with capacity level q ($q \in \Gamma_k$), the binary variable H_k^q is one. To model the limitation on outgoing hub arcs, we introduce the binary variable A_{kl} which is one if hub node k is connected to node l and zero otherwise. The capacitated multiple allocation hub location model is given as follows:

$$\min \sum_{k \in N} \sum_{q \in \Gamma_k} c_k^q H_k^q + \sum_{k \in N} \sum_{l \in N} c_{kl} (X_{kl} + \sum_{i \in N} (\beta Y_{kl}^i + Z_{kl}^i)) \quad (1)$$

$$\text{s. t. } \sum_{i \in N} (X_{ik} + \sum_{l \in N} Y_{lk}^i) \leq \sum_{q \in \Gamma_k} \gamma_k^q H_k^q \quad \forall k \in N \quad (2)$$

$$\sum_{q \in \Gamma_k} H_k^q = H_k \quad \forall k \in N \quad (3)$$

$$Y_{kl}^i + Z_{kl}^i \leq O_i A_{kl} \quad \forall i, k, l \in N \quad (4)$$

$$\sum_{l \in N} A_{kl} \leq \sum_{q \in \Gamma_k} b_k^q H_k^q \quad \forall k \in N \quad (5)$$

$$A_{kl} \leq H_k \quad \forall k, l \in N \quad (6)$$

$$\sum_{k \in N} X_{ik} = O_i \quad \forall i \in N \quad (7)$$

$$X_{ik} + \sum_{l \in N} Y_{lk}^i = \sum_{l \in N} Y_{kl}^i + \sum_{j \in N} Z_{kj}^i \quad \forall i, k \in N \quad (8)$$

$$\sum_{l \in N} Z_{lj}^i = W_{ij} \quad \forall i, j \in N \quad (9)$$

$$X_{kk} = O_k H_k \quad \forall k \in N \quad (10)$$

$$Z_{kk}^i = W_{ik} H_k \quad \forall i, k \in N \quad (11)$$

$$X_{ik} \leq O_i H_k \quad \forall i, k \in N \quad (12)$$

$$Y_{kl}^i \leq O_i H_k \quad \forall i, k, l \in N \quad (13)$$

$$Y_{kl}^i \leq O_i H_l \quad \forall i, k, l \in N \quad (14)$$

$$Z_{lj}^i \leq W_{ij} H_l \quad \forall i, j, l \in N \quad (15)$$

$$X_{ik}, Y_{kl}^i, Z_{kj}^i \in \mathbb{Z}^+, H_k, H_k^s \in \{0, 1\} \quad \forall i, j, k, l \in N, s \in \Gamma_k \quad (16)$$

The objective function (1) minimizes the total costs, consisting of hub and transportation costs, where inter-hub flow is discounted. Constraints (2) assure that all flow into a hub may not exceed the capacity assigned to it. Exactly one capacity level has to be chosen for each established hub (cf. constraints (3)). In order to restrict the number of outgoing connections constraints (4), (5), and (6) are used. Constraints (7), (8), and (9) are flow related constraints. If an origin-destination node becomes a hub node, the demand has to be (re-)classified and thus, influences the capacity of the hub. Due to this, we assure that if a node becomes a hub, we have “artificial” collection or distribution movements from the hub to itself (cf. constraints (10) and (11)). Constraints (12)-(15) ensure usable solutions: a hub has to be established for each collection, transfer, and distribution movement. Constraints (16) are domain constraints. In account for practical requirements, it might be useful to assume a maximum distance between non-hub and hub nodes. This can be done by fixing corresponding collection and distribution variables to zero.

3 Local Search Heuristics

Heuristics based on local improvements are well known in hub location literature since they can provide good solutions without consuming a great amount of CPU time [1,2]. Due to the publicity of local search algorithms, we only describe the main details of the heuristic developed here. The approach exploits some ideas of possible moves of the simulated annealing approach in [4] for a (classical) capacitated hub location problem with single allocation.

One main element of applying local search approaches to specific problems is the definition of the neighborhood and, in particular, the definition of moves to get from one solution to another. We define different hub-based and flow-based moves to solve the problem which exploit the structure of the problem: location and allocation decisions. In some moves so called *nearby* non-hub or hub nodes of the current node are used. These nearby nodes are determined as follows: According to a sorted list (in terms of transportation costs), the best nodes are chosen to be nearby nodes of the given node. Additionally, a randomly chosen candidate is added. Finally, one nearby node is randomly chosen out of this list.

We consider the following hub-based moves: *Relocate Hub* exchanges the location of a hub with a nearby non-hub node. The corresponding flow is allocated to the new hub. The move *New Hub* establishes a new hub node; a non-hub node becomes a hub node. By doing so, the corresponding collection and distribution flow of the new hub is allocated to itself. *Close Hub* closes a hub node. The corresponding flow is allocated to a nearby hub node. The move *Split Hub* reallocates some origin-destination flows via a hub to an allocated non-hub node, which becomes a hub node. By doing so, the flow of a hub is split and a new hub is established.

Moreover, we consider some flow-based moves: By applying the move *Reallocate Flow Collection*, the collection flow of an origin (non-hub) node to a hub node is reallocated to a nearby hub node. Similarly, *Reallocate Flow Distribution* changes the distribution hub of an origin-destination flow. Corresponding transfer and collection movements are modified. The move *Swap Flows Collection* chooses flows from two origin-destination pairs in the collection phase (with different, but nearby collection hubs) and swaps the allocation of the two collection flows. In other words, each flow is allocated to the hub of the other one. Similarly, the move *Swap Flows Distribution* reallocates the distribution flow of two origin-destination pairs to the nearby (distribution) hub of the other origin-destination pair. In both cases, affected movements are modified. By applying *Shift Flow*, an origin-destination flow is transported via an additional hub. Thus, for a chosen origin-destination pair a movement between two hubs is replaced by two new movements from the first hub to a (randomly) chosen hub and from the chosen hub to the second hub.

The main structure of our local search approach is given as follows. At first, we generate a feasible initial solution. According to a sorted list (in terms of transport volumes) an initial hub set is chosen, whereat the capacity of these hubs exceeds the total demand. Moreover, we add some supplementary hubs. Starting with this initial hub set, the solution is built up successively. For each

origin-destination pair collection, transfer, and distribution movements are determined. If all origin-destination nodes are allocated, the initial heuristic stops; otherwise another hub node is added to the hub list, all movements are deleted, and the prior steps are repeated. In the following iterations, we generate multiple neighbor solutions of the current solution. The best feasible solution is compared to the current best solution. If the new solution is better than the current best solution, we replace it. Infeasible solutions created by the moves are not considered as starting points for further search. In order to generate a new solution in the neighborhood of the current one, we choose one of the moves described above. The (preliminary chosen) probabilities that one move is chosen are set to 15%, 5%, 5%, 5%, 15%, 15%, 10%, 10%, and 20%, respectively. The setting is based on previous tests and the results of [4].

We implemented three versions of the local search approach described above. In order to escape from local minima, we implemented a local search with multistarts **MLS** and a simulated approach **SA**. In **MLS** we start the local search procedure several times to escape from local optima. This is done by applying moves to the current best solutions and accepting all feasible solutions for a given number of iterations. Moreover, we accept a new solution for the further search if it is at most 10 percent worse than the current best solution. This is a version of threshold accepting, i.e. accepting worse solutions within a given threshold. Based on the same input parameters, we also implemented a simulated annealing version, where the probability to accept worse solutions decreases with an increasing run-time. Solutions increasing the objective value are accepted according to the Boltzmann's probability (depending on the decreasing temperature). The two heuristics are also compared to a pure local search (descent) variant **DLS**. While the heuristic **MLS** and **SA** allow uphill moves, the descent approach accepts only improvements. The stop criterion is given by a time limit (here 43,200 sec) or by a given number of iterations (here 500 iterations) with no improvement of the objective value. Moreover, we test a simple constructive heuristic **CH**. This heuristic is similar to the heuristic used for generating the initial solution, whereat we do not add additional hub nodes.

4 Computational Results

All heuristics were implemented in MATLAB R2011b. In order to compare the quality of the heuristic solutions, the model was implemented in GAMS 23.9.1 and solved with CPLEX 12.4.0.1. All tests were carried out on a PC under Windows 7 (64 bit) with an Intel Core i5 CPU, 3.2 GHz, and 16 GB

RAM. Aiming at an approach applicable in practice, the time limit was set to 12 hours.

In the computational experiments we test the effectiveness of the heuristics with different data sets provided by Deutsche Bahn AG. The results are presented in Table 1. The columns *gap* represent the relative gap. According to CPLEX, we define the gap as $gap = 100 \cdot \frac{BF - BP}{BF}$, where we use the lower bounds (*BP*) found by GAMS/CPLEX and the best feasible solution (*BF*) found by the corresponding solution approach. Since **DLS** and **CH** do not reach the time limit of 12 hours, we also present the time used. Since the random elements have an influence on the results, the heuristics are repeated five times for each data set. The average gap is based on the average *BF*; the presented deviations are based on the original *BF* in comparison to the average *BF*.

Instance no. nodes	GAMS/CPLEX		MLS		SA		DLS		CH	
	gap (%)	ø gap (%)	ø gap (%)	ø gap (%)	ø sec	gap (%)	sec			
1 37	13.98	12.20 ($\pm 3.20\%$)	14.47 ($\pm 3.66\%$)	16.53 ($\pm 3.45\%$)	72	41.43	< 1			
2 49	21.59	15.70 ($\pm 3.65\%$)	21.87 ($\pm 2.32\%$)	23.22 ($\pm 4.20\%$)	297	49.91	< 1			
3 74	-	20.77 ($\pm 5.81\%$)	26.43 ($\pm 2.82\%$)	26.32 ($\pm 2.54\%$)	4,250	50.74	< 1			
4 74	4.60	9.97 ($\pm 1.21\%$)	12.23 ($\pm 1.41\%$)	21.24 ($\pm 7.04\%$)	5,855	36.50	< 1			

Table 1
Computational results

Even for small test data sets no optimal solutions are found by CPLEX within a run-time of 12 hours. Although, no heuristic can find (proven) optimal solutions, the results in Table 1 are promising. In most cases tested, **MLS** can outperform the results found by CPLEX – even for small scenarios. By comparing the heuristic approaches, we observe that **MLS** outperforms the other heuristics. In particular, the solutions found by **MLS** outperform the best solutions found with heuristics **SA** and **DLS**. Although, the simulated annealing approach accepts solutions worse than the current best solutions, we observe once "good" best solutions are found, the heuristic cannot improve these values (for hours). We observed this effect for varying input parameters for the acceptance criterion, i. e., the initial temperature and the annealing schedule. One reason might be that the defined neighborhoods are not appropriate for simulated annealing. Due to the (random) multistart, which accepts all feasible solutions for a given numbers of iterations, we get a greater chance to escape from local optima. In particular, we observe that the limitation of outgoing connections increases the difficulty to find feasible (and better) solutions in the neighborhood of a solution. By considering the solution process, we noticed that satisfying solutions are found within a small amount of run

time and hence, can be used as upper bounds within in an exact algorithm.

5 Conclusion and future work

In this paper, we developed different versions of a local search approach for solving a capacitated multiple allocation hub location problem derived from a practical application in network design of German wagonload traffic. These heuristics are based on local improvements and exploit the structure of the problem: location and allocation decisions. First computational results, in particular of the local search with multistart, are promising for solving real data sets. All heuristic approaches can be used to find satisfying solutions in a small amount of run time. From a practical point of view, these heuristics can be easily extended to other practical requirements, like vehicle-based costs, due to the modular construction. Although, first computational results are promising, in future work we consider further tests on the parameters, e.g. probabilities, in order to improve the results. Moreover, future work is dedicated to improving the heuristics in terms of the final solution, i.e. closing the gap, as well as solving larger hub location problems.

References

- [1] Sibel Alumur and Bahar Y. Kara. Network hub location problems: The state of the art. *European Journal of Operational Research*, 190:1–21, 2008.
- [2] James F. Campbell, Andreas T. Ernst, and Mohan Krishnamoorthy. Hub location problems. In Zvi Drezner and Horst H. Hamacher, editors, *Facility location. Applications and theory.*, chapter 12, pages 373–407. Springer, 2002.
- [3] James F. Campbell and Morton E. O’Kelly. Twenty-five years of hub location research. *Transportation Science*, 45(2):153–169, 2012.
- [4] Andreas T. Ernst and Mohan Krishnamoorthy. Solution algorithms for the capacitated single allocation hub location problem. *Annals of Operations Research*, 86:141–159, 1999.
- [5] Julia Sender and Uwe Clausen. Optimizing the network in German wagonload traffic. submitted to: Proceedings of 5th International Conference on Railway Operations Modelling and Analysis Planning, Simulation and Optimisation Approaches.

2-InterConnected Facility Location: Specification, Complexity, and Exact Solutions

Markus Chimani^{1,4}

Faculty of Mathematics and Computer Science, Uni Jena, Germany

Maria Kandyba^{2,4}

Jena, Germany

Maren Martens^{3,4}

Munich, Germany

Abstract

Connected facility location combines cost-efficient facility placement and the requirement to connect the facilities among each other. Such problems arise, e.g., in telecommunication applications where networks consist of a central core and local clients connected to it. Reliability of the core is a central issue, and we may hence require the core to be at least 2-connected.

We establish the problem class of 2-interConnected Facility Location (2-iCFL), categorize its central variants, and prove that they are hard to approximate. However, our computational results show that our orientation-based ILPs allow us to effectively solve such problems to optimality for hundreds of nodes. We also establish constructive characterizations for feasible problem instances, to be used for algorithmic feasibility checks, preprocessing, and heuristics.

Keywords: facility location, 2-connectivity, inapproximability, exact algorithms, integer programming, experiments

¹ MC was funded via a junior professorship by the Carl-Zeiss foundation.

² MK was funded by the German Telecom foundation.

³ MM was funded by the Zuse Institute Berlin (ZIB).

⁴ Emails: markus.chimani@uni-jena.de, maria.kandyba@gmail.com, m.mts@gmx.de

1 Introduction

Combined facility location and connectivity problems have gained a lot of attention in the past decade. Herein, we consider cost-minimization problems where we have to open facilities (choosing from a given set of locations) and connect each other location (clients) to them. Our motivation stems from various industrial cooperations where the goal is to cost-efficiently design optical telecommunication networks. A typical real-world network contains an especially reliable core (using higher-grade hardware) and clients that are directly connected to it. For reliability, it is common to require that the core network is 2-connected whereas a client needs only a simple connection to some node in the core (a facility).

Let $G = (V, E)$ be an undirected graph, $n := |V|$, $m := |E|$, $P \subseteq V$ a set of possible facilities, $f : P \rightarrow \mathbb{R}_0^+$ non-negative opening costs, and $c : E \rightarrow \mathbb{R}_0^+$ non-negative edge costs. A *2-interconnected facility location problem* (2-iCFL) is to choose a subset of facilities $\mathcal{F} \subseteq P$ and to connect each node $u \notin \mathcal{F}$ to some node $v \in \mathcal{F}$ directly via an edge $\{u, v\} \in E$. Furthermore, for each pair of nodes $u, v \in \mathcal{F}$ we require two disjoint paths, only visiting facilities. In the resulting network $N = (V, E' \subseteq E)$, we minimize the function $\sum_{v \in \mathcal{F}} f_v + \sum_{e \in E': |e \cap \mathcal{F}|=1} c_e + M \cdot \sum_{e \in E': |e \cap \mathcal{F}|=2} c_e$, where $M \geq 1$ reflects possibly higher costs for core connections (observe that edges can be interpreted as two-element vertex-subsets). We call the network $N[\mathcal{F}]$ spanned purely by the facilities, the *core* of the solution. Since our edge costs are non-negative, all nodes $v \notin \mathcal{F}$ have degree 1 in N .

We are dealing with a *2-edge-interconnected facility location problem* (2E-iCFL) if we require two edge-disjoint paths between each pair of facilities, i.e., a 2-edge-connected core. Requiring node- instead of edge-disjointness (2-node-connected core), we obtain the *2-node-interconnected facility location problem* (2N-iCFL). We write simply *2-iCFL* when our results hold for both versions of connectivity. In real-world applications the input may already contain a fixed facility r . We refer to such 2-iCFL problems as *rooted 2-iCFL* (r -2-iCFL), or more specifically r -2N-iCFL and r -2E-iCFL. A problem is *degree-bounded* if each node may have degree at most Δ in the solution. This comes from the practical background that, e.g., a router in a telecommunication network can only be attached to some maximum number of other devices, due to its number of physical slots, plugs, or sockets.

Connected facility location and variants thereof have been considered in a wide range of publications, both from the approximation and the mathematical programming point of view; see [3] for a more detailed discussion and

bibliography. However, explicit two-connectivity requirements for the core have not yet gained large attention. A special case of 2-iCFL are the *Ring Star* or *tour-CFL* problems [4,5], where the facilities are required to lie on a simple cycle. To the best of our knowledge, 2-iCFL has only been considered in [6]—in the context of complete cores and for $M = 1$. Therein, an undirected ILP formulation has been used to solve instances of up to 25 nodes with a fixed number of open facilities.

Contribution. We establish the problem class of 2-iCFL and categorize its central variants. While all such variants are clearly NP-hard, we show that even strong approximations are NP-hard (Sect. 2). For degree-bounded 2-iCFL, already checking if an instance allows any feasible solution is NP-hard. On the other hand, feasibility of 2-iCFL variants without degree bounds can be tested in linear time (Sect. 3), giving rise to preprocessing routines and heuristics. In Sect. 4, we show how 2-iCFL can be solved to optimality using strong orientation-based ILP formulations and conduct experiments in Sect. 5. The obtained ILP is stronger (also theoretically) than the above mentioned [6] for a special case of 2-iCFL. Due to space limitations, we refer the reader to the technical report [3] for detailed proofs omitted herein.

2 Complexity

For the following results on the complexity of 2-iCFL, we can restrict our considerations to $M = 1$ without loss of generality.

Theorem 2.1 *It is NP-hard to approximate 2-iCFL within a factor $c \log |V|$, for some constant $c > 0$. This is true even in complete graphs.*

Proof (Sketch) (cf. [3]). We can show this result using a reduction of MINIMUM DOMINATING SET (MDS), for which the above inapproximability is known [7]. Given a graph H for MDS, we construct a 2-iCFL instance by considering a graph G that contains two nodes for each vertex in H (forming the node subsets V_H, V'_H) and one node for each edge (forming the node subset V_E). The edges in G are given by the incidence relation between nodes and edges in H . Furthermore, G contains two special nodes, one of which is connected to all nodes of V_H , the other one to all nodes of V_E . All edge costs are 0, and P contains all nodes except V'_H , with all opening costs 0 except for those of the nodes V_H . It can be shown that any $(c \log |V|)$ -approximation of 2-iCFL gives rise to an approximation of equivalent factor for MDS. \square

When we add a degree bound, 2-iCFL becomes much harder:

Theorem 2.2 *For any constant degree bound $\Delta \geq 2$, it is NP-hard to approximate degree-bounded 2-iCFL within a factor $c^{p(n)}$, where $c > 1$ is a constant and $p(n)$ is a polynomial in n . This is already true in complete graphs.*

Proof (Sketch) (cf. [3]). We first consider only $\Delta = 2$ and a reduction of HAMILTONIAN CYCLE. Given an instance of the latter problem, we set the cost of each edge to 1, and then extend the graph to a complete graph with cost $K := (c^{p(n)} - 1)n + 2$ for each new edge. All vertices are potential facilities with opening costs 0. It can be shown that any feasible solution within the desired approximation ratio would have to resemble a Hamiltonian path. The case $\Delta > 2$ can be handled by adding $\Delta - 2$ many nodes (not to be included in P) for each original node v , with 0-cost edges to this v . \square

3 Feasibility Checking, Preprocessing, and Heuristic

If we leave out every edge of cost K in the above proof, there is a feasible solution to 2-iCFL iff H contains a Hamiltonian cycle. Hence:

Corollary 3.1 *It is NP-hard to decide whether an instance of degree-bounded 2-iCFL admits a feasible solution.*

In contrast to this, we can test feasibility of instances without degree bound in linear time. Herein, we mention only the central ideas, see [3] for details. It can be observed that (except for some trivial special cases) an instance is feasible iff there is a unique special component B in the graph induced by P , with the property that all other nodes are incident to at least one node in B . For 2N-iCFL, the candidates for B are the 2-node-connected components, while for 2E-iCFL we have to consider maximal unions of overlapping two-node-connected components (disregarding bridges). The two-connectivity test and all the subsequent tests require only linear time.

The testing algorithm can also be used as a preprocessing routine to fix and delete edges and fix nodes as clients or facilities. Finally, the above observations give rise to a trivial greedy heuristic: Knowing the special component B , we select all of B into \mathcal{F} and attach each other node to a facility neighbor with smallest edge cost. Then, we incrementally try to improve this solution by removing edges and changing facilities into clients, in decreasing order of cost improvement, checking feasibility after each step.

4 Exact Algorithms

We can formulate the variants of 2-iCFL as integer linear programs using directed cuts. The 2-connectivity model of the formulation is based on the currently strongest ones known for $\{0, 1, 2\}$ -survivable network design [2]. We start with considering r-2-iCFL problems, as they form the basis of all our ILP formulations. We then extend these formulations to the unrooted problems.

r-2-iCFL. Given an undirected graph $G = (V, E)$, we say that a directed graph $\hat{G} = (V, A)$ is an *orientation* of G if it contains exactly one of the directed arcs $(v, u), (u, v)$ for every edge $\{v, u\} \in E$. Based on the orientation properties of 2-node/edge-connected graphs [2], we can recognize feasible networks (including an optimal one) as follows (proof omitted, see [3]).

Lemma 4.1 *A network N with facilities \mathcal{F} is feasible for r-2E-iCFL (or r-2N-iCFL) if and only if there exists an orientation \hat{N} of N with properties (P1)–(P2) (or (P1)–(P4), respectively):*

- (P1) *Each node $v \notin \mathcal{F}$ has in-degree 1 and out-degree 0.*
- (P2) *For each $v \in \mathcal{F} \setminus \{r\}$, \hat{N} contains directed paths $(r \rightarrow v)$ and $(v \rightarrow r)$.*
- (P3) *For each $v \in \mathcal{F} \setminus \{r\}$, the two paths of (P2) are internally node-disjoint.*
- (P4) *The in-degree of r is 1.*

Instead of $G = (V, E)$ we consider its bidirected counterpart $G' = (V, A)$ which contains the two arcs $(v, u), (u, v)$ for each edge $\{v, u\} \in E$. As implied in Sect. 3, we can assume $\{v, u\} \cap P \neq \emptyset$ for all $\{v, u\} \in E$. We use binary indicator variables x_a for each $a \in \mathcal{X} := A \cap (P \times P)$, z_a for each $a \in \mathcal{Z} := A \cap (P \times V)$, and y_v for each $v \in P$. We set $x_a = 1$ iff a is in the solution's core, $z_{(v,u)} = 1$ iff the client u is attached to the facility v , and $y_v = 1$ iff v is chosen as a facility. Let $S \subset V$, then $\delta^+(S)$ and $\delta^-(S)$ denote the arcs leaving and entering S , respectively. We may use one or more nodes U as a subscript to denote that any arcs incident to U shall be ignored. Furthermore, we use the shorthands $x(B) := \sum_{a \in B \cap \mathcal{X}} x_a$ and $z(B) := \sum_{a \in B \cap \mathcal{Z}} z_a$ for $B \subseteq A$.

We can write the following ILP for r-2N-iCFL (Figure 1). Selected arcs emanate only from facilities, (2). By (3), every node is either attached to a facility or belongs to the core. Each edge within the core is uniquely oriented, (4). If a node is a facility, there exists a path from the root to it and vice versa, (5), (6). We guarantee node-disjointness of these two paths by forbidding any common internal node w , (7). Finally, (8) guarantees a single block containing all facilities. By omitting the latter two constraints, we obtain the formulation for r-2E-iCFL.

$\min \sum_{i \in P} f_i y_i + \sum_{a \in \mathcal{Z}} c_a z_a + M \cdot \sum_{a \in \mathcal{X}} c_a x_a$	(1)	
$x_{vw} \leq y_v, \quad z_{vu} \leq y_v \quad \forall (v, w) \in \mathcal{X}, (v, u) \in \mathcal{Z}$	(2)	
$z(\delta^-(\{v\})) + x(\delta^-(\{v\})) \geq 1 \quad \forall v \in V$	(3)	
$x_{vw} + x_{wv} \leq 1 \quad \forall (v, w), (w, v) \in \mathcal{X}$	(4)	
$x(\delta^-(S)) \geq y_v \quad \forall S \subseteq V \setminus \{r\}, \forall v \in S \cap P$	(5)	
$x(\delta^+(S)) \geq y_v \quad \forall S \subseteq V \setminus \{r\}, \forall v \in S \cap P$	(6)	
$x(\delta_w^+(S_1)) + x(\delta_w^-(S_2)) \geq y_v \quad \forall w \in V \setminus \{r\}, \forall S_1, S_2 \subseteq V \setminus \{r, w\},$ $\forall v \in S_1 \cap S_2 \cap P$	(7)	
$x(\delta^-(\{r\})) = 1$	(8)	
$x_a, z_b, y_u \in \{0, 1\} \quad \forall a \in \mathcal{X}, b \in \mathcal{Z}, u \in P$	(9)	
$x(\delta^+(\{r\})) = 1$		(10)
$x(\delta_r^+(S)) \geq y_u + y_v - 1 \quad \forall u \in P \setminus \{r\}, S \subseteq V \setminus \{r, u\},$ $\forall v \in S \cap P$	(11)	
$x(\delta_{w,r}^+(S_1)) + x(\delta_{w,r}^-(S_2)) \geq y_u + y_v - 1 \quad \forall w \in V \setminus \{r\}, \forall u \in P \setminus \{r, w\},$ $\forall S_1, S_2 \subseteq V \setminus \{r, w, u\},$ $\forall v \in S_1 \cap S_2 \cap P$	(12)	
$\sum_{i: \{v,i\} \in E} x_{vi} + x_{iv} + z_{vi} + z_{iv} \leq \Delta \quad \forall v \in P.$	(13)	

Fig. 1. ILP for r-2N-iCFL (top), and additional constraints for derivatives, cf. text.

2-iCFL, degree-bounds, etc. We can extend the formulation to the unrooted problem family: We extend G via an artificial root node $r \in P$ and introduce zero-cost edges $r \times (P \setminus \{r\})$ together with their x -variables, but without z 's. We restrict the solution space via (10) to networks where r requires a unique out-going arc. It remains to guarantee 2-connectivity even after the removal of r , which can be done via (11) for 2E-iCFL solutions and via (12) for 2N-iCFL.

A maximum degree restriction Δ can be added to each of the above formulations straight-forwardly by requiring (13). In the special case $M = 1$, our ILPs can be simplified by carefully unifying the x and z variables.

Overall, the above formulations give optimal solutions for their respective problems (i.e., any combination of edge- vs. node-connectivity; rooted vs. unrooted; degree-bounded vs. unrestricted). Observe that, although requiring an exponential number of constraints, the ILPs can be solved straight-forwardly by separating the cut-type constraints in polynomial time via max flow. Furthermore, it follows analogously to known proofs [2]:

Observation 4.2 *Our directed cut formulations are stronger than undirected cut formulations, and are equally strong as, but practically preferable over, formulations based on multicommodity flow.*

5 Experiments

As a proof-of-concept, we briefly evaluate the general practicability of our approach, and want to estimate the ILPs' dependance on various instance properties. We implemented a Branch-and-Cut algorithm using the above r-2-iCFL formulations, using CPLEX and separation via C++, facilitating the efficient max-flow code of [1]. The experiments were conducted on an Intel Xeon 2.33Ghz CPU with 2GB of RAM per process (32bit).

We consider the following generated test suite⁵, to allow us to observe the algorithms' dependency on different input parameters, denoted by n , λ , ϱ , and δ in the following: We distribute n points uniformly at random on a 1000×1000 grid (taking the first chosen node as the root), and compute the Euclidean distances $d_{\{u,v\}}$ for each pair of nodes $\{u,v\}$. Based thereon, we generate (random) edge costs via a Gauss distribution, using $d_{\{u,v\}}$ as the mean and $\lambda \cdot d_{\{u,v\}}$ as the variance. Clearly, $\lambda = 0$ gives Euclidean instances. Using a uniform distribution, we select a percentage ϱ of the nodes into the set P of potential facilities with uniformly distributed costs from the interval $[250, 750]$ ⁶. Overall, we generate a percentage δ of all possible edges. To guarantee feasible instances, we thereby start with a random cycle through P ; for every node $V \setminus P$ we introduce an edge connecting it to some (randomly chosen) node in P . The remaining edges are generated randomly over all remaining node pairs. In the light of Sect. 3, we consider parameter values such that the obtained graph is a single block and hence not easily preprocessable. For each parameter setting $(n, \lambda, \varrho, \delta)$, we generate 5 instances.

Table 1⁷ shows the central findings for $M = 1$; the running times for $M = 10, 20$ are virtually the same. Small graphs and few routers are easily solvable. Bold entries are those where the algorithmic behavior starts to degrade. Interestingly, degradation is not mainly due to the graph size, but the size of P ; degradation occurs when $|P| \in [40, 50]$. It is potentially fortunate that in most practical settings (e.g, in telecommunications), P seems to be rather small. Also, the graph's density does not play a crucial role. While 2E-iCFL and 2N-iCFL are computationally comparable, the Gauss variance is of moderate importance; Euclidean instances are the simplest.

⁵ See <http://ls11-www.cs.tu-dortmund.de/people/kandyba/2iCFL-instances>.

⁶ This interval constitutes the sweet spot with respect to the expected edge lengths to avoid degenerate cases where either all or only a single potential facility would be opened.

⁷ The first specification in each cell is for r-2E-iCFL, the second for r-2N-iCFL. We describe the average computation time as < 1sec, < 1min, < 10min, or < 1h, denoted by s, m, T, and H, respectively. If not all 5 instances could be solved within one hour, we also give the number of solved instances (“—” denotes no success at all).

Table 1
Overview on the experimental results for $M = 1$ (cf. text and footnote 7)

n	λ	$\varrho = 10\%, \delta =$			$\varrho = 25\%, \delta =$			$\varrho = 50\%, \delta =$		
		30%	70%	100%	30%	70%	100%	30%	70%	100%
50	0.0	s s	s s	s s	s s	s s	s s	m s	m m	m m
	0.1	s s	s s	s s	s s	s s	s s	m s	m m	m m
	0.3	s s	s s	s s	s s	s s	s s	m s	m m	m m
100	0.0	s s	s s	m s	m m	m m	m m	T4 T4	H T	T T
	0.1	s s	s s	m m	m s	m m	m m	H3 H3	T4 T4	T T
	0.3	s s	m m	m m	m m	m m	m m	H2 H2	H2 H2	H3 H3
200	0.0	m m	m m	m m	H4 H4	H4 H4	H H	— —	— —	— —
	0.1	m m	m m	m m	H3 H2	H4 H3	H H	— —	— —	— —
	0.3	m m	m T	T T	— H1	H3 H3	— —	— —	— —	— —
400	0.0	H H3	H3 H4	H H	— —	— —	— —	— —	— —	— —
	0.1	T H	H4 H4	H1 H2	— —	— —	— —	— —	— —	— —
	0.3	H H4	H3 H1	H1 H1	— —	— —	— —	— —	— —	— —

While the problem's complexity may seem daunting at first, our experiments show that exact approaches are promising. Even in their current simple form, without requiring strong primal heuristics or algorithmic tricks, they are applicable to some real-world problems with not too many potential routers.

References

- [1] B. V. Cherkassky and A. V. Goldberg. On implementing push-relabel method for the maximum flow problem. *Algorithmica*, 19:390–410, 1997.
- [2] M. Chimani, M. Kandyba, I. Ljubić, and P. Mutzel. Orientation-based models for {0,1,2}-survivable network design: Theory and practice. *Mathematical Programming B*, 124:413–439, 2010.
- [3] M. Chimani, M. Kandyba, and M. Martens. 2-interconnected facility location: Specifications, complexity results, and exact solutions. Tech.Rep. TR-FSUJ-CS-AE-11-00 (v.2). Algorithm Engineering, Computer Science, FSU Jena. https://www.ae.uni-jena.de/alenmedia/de/dokumente/TR_2iCFL_pdf.pdf.
- [4] F. Eisenbrand, F. Grandoni, T. Rothvoß, and G. Schäfer. Approximating connected facility location problems via random facility sampling and core detouring. In *Proc. SODA '08*, pages 1174–1183, 2008.
- [5] M. Labb  , G. Laporte, I. R. Mart  n, J. Jos  , and S. Gonz  lez. The ring star problem: Polyhedral analysis and exact alg. *Networks*, 43(3):177–189, 2004.
- [6] M. Pioro and D. Medhi. *Routing, Flow, and Capacity Design in Communication and Computer Networks*. Morgan Kaufmann Publishers, 2004.
- [7] R. Raz and S. Safra. A sub-constant error-probability low-degree test, and sub-constant error-probability PCP characterization of NP. In *Proc. STOC '97*, pages 475–484, 1997.

A polynomial formulation for the stochastic maximum weight forest problem

Pablo Adasme¹

*Departamento de Ingeniería Eléctrica, Universidad de Santiago de Chile
Avenida Ecuador 3519, Santiago, Chile.*

Rafael Andrade²

Departamento de Estatística e Matemática Aplicada, Universidade Federal do Ceará, Campus do Pici - Bloco 910, 60455-760, Fortaleza, Ceará, Brasil.

Marc Letournel, Abdel Lisser³

*Laboratoire de Recherche en Informatique, Université de Paris-Sud XI
Bât. 650, 91405, Orsay Cedex France.*

Abstract

Let $G = (V, E_D \cup E_S)$ be a non directed graph with set of nodes V and set of weighted edges $E_D \cup E_S$. The edges in E_D and E_S have deterministic and uncertain weights, respectively, with $E_D \cap E_S = \emptyset$. Let $S = \{1, 2, \dots, P\}$ be a given set of scenarios for the uncertain weights of the edges in E_S . The stochastic maximum weight forest (SMWF) problem consists in determining a forest of G , one for each scenario $s \in S$, sharing the same deterministic edges and maximizing the sum of the deterministic weights plus the expected weight over all scenarios associated with the uncertain edges. In this work we present two formulations for this problem. The first model has an exponential number of constraints, while the second one is a new compact extended formulation based on a new theorem characterizing forests in graphs. We give a proof of the correctness of the new formulation. It generalizes existing related models from the literature for the spanning tree polytope. Preliminary results evidence that the SMWF problem can be NP-hard.

Keywords: compact formulation, stochastic maximum weight forest problem.

1 Introduction

Consider a non directed graph $G = (V, E_D \cup E_S)$, with set of nodes V and set of weighted edges $E_D \cup E_S$, with $E_D \cap E_S = \emptyset$. The edges in E_D and E_S have deterministic and uncertain weights, respectively, with $E_D \cap E_S = \emptyset$. The edges in E_D and E_S have deterministic and uncertain weights, respectively. Let $S = \{1, 2, \dots, P\}$ be a given set of scenarios for the uncertain weights of the edges in E_S . The stochastic maximum weight forest (SMWF) problem [3] consists in determining a forest of G , one for each scenario $s \in S$, sharing the same deterministic edges and maximizing the sum of the deterministic weights plus the expected weight over all scenarios associated with the uncertain edges.

For $|S| = 1$, the problem is to find a forest of maximum weight in G and can be done in polynomial time [5]. There is no similar result concerning the complexity of the SMWF problem, but we present some instances that have not the totally dual integral (TDI) property. This evidences that this problem can be NP-hard. In fact, it has been practically impossible to generate randomly non TDI instances for this problem.

One of the main difficulties in investigating the SMWF problem concerns the large dimensions of a linear model for solving the problem instances. We can transform accordingly SMWF instances to obtain a complete graph so that any scenario solution be a spanning tree and use existing spanning tree formulations [4,6,1] to model the SMWF problem. But we can overcome transforming forests into trees. Thus, saving memory space and execution time, by introducing a new characterization theorem for forests in graphs that allows to obtain a compact extended formulation for the problem. This result is novel for the SMWF problem and generalizes the ideas from the spanning tree polytope (see e.g. [4,6,1]), constituting the main contribution of this work. We prove the correctness of the new formulation. With this compact extended formulation we can now handle instances with up to 40 nodes.

2 Problem formulation

Let $G = (V, E_D \cup E_S)$ be an undirected graph and $S = \{1, 2, \dots, P\}$ be the set of scenarios as described in Section 1. Let π_s be the probability associated with a given scenario $s \in S$, with $\sum_{s=1}^P \pi_s = 1$ and $\pi_s \geq 0$, for all $s \in S$. Let $c_{uv}^s \in \mathbb{R}$, with $u, v \in V$, represent the weight of an edge uv in scenario $s \in S$,

¹ Email: pablo.adasme@usach.cl

² Email: rca@lia.ufc.br

³ Email: {marc.letourneau, abdel.lisser}@lri.fr

with $c_{uv}^1 = c_{uv}^2 = \dots = c_{uv}^P$ given for all $uv \in E_D$. The uncertain weights c_{uv}^s of each scenario $s \in S$, for all $uv \in E_S$, are randomly generated. Let represent a forest of G in scenario $s \in S$ by a vector $x^s = (x_D^s, x_S^s) \in \{0, 1\}^{|E_D \cup E_S|}$, where $x_{uv}^s = 1$ if uv belongs to the forest of that scenario, and $x_{uv}^s = 0$, otherwise. We must have $x_D^1 = x_D^2 = \dots = x_D^P$, thus we replace these variables by a unique vector $y \in \{0, 1\}^{|E_D|}$. Let $E(H)$ represent the set of edges with both extremities in the subset of nodes $H \subset V$, such that $|H| \geq 3$. The SMWF problem is

$$(1) \quad (P_1) \quad \max \sum_{uv \in E_D} c_{uv}^1 y_{uv} + \sum_{s \in S} \pi_s \left(\sum_{uv \in E_S} c_{uv}^s x_{uv}^s \right)$$

$$(2) \quad \text{s.t.} \quad \sum_{uv \in E(H) \cap E_D} y_{uv} + \sum_{uv \in E(H) \cap E_S} x_{uv}^s \leq |H| - 1, \quad \forall H \subset V, \forall s \in S$$

$$(3) \quad x_{uv}^s \in \{0, 1\}, \quad \forall uv \in E_S, \quad \forall s \in S, \quad y_{uv} \in \{0, 1\}, \quad \forall uv \in E_D$$

Constraints (2) impose the non existence of cycles in any solution.

Proposition 2.1 *The maximum weight forest problem (i.e. the SMWF with $|S| = 1$) is polynomially solvable by a greedy algorithm [2].*

Proposition 2.2 *Formulation (P_1) is not totally dual integral (TDI).*

In [3] the authors present a small example for a six nodes graph and three scenarios of edges weights where the linear relaxed solution is not integer. This proves Proposition 2.2. The fact that the problem may present fractional linear relaxed solution makes it difficult to deal with. Moreover, the exponential number of constraints in (P_1) makes the problem intractable, thus we propose a new compact formulation for this problem.

3 Polynomial formulation

Before presenting the new formulation, we need a new characterization for forests in undirected graphs. Let, for a given node $k \in V$ and a given scenario $s \in S$, F_k^s represent an abstract orientation of the edges of a general subgraph F of G such that the node k in the scenario s is taken as a referential in F . In this case, if an edge uv belongs to F , then the referential node k observes u preceding $v \neq k$ or $v \neq k$ preceding u , but not both, in F_k^s .

Theorem 3.1 *Let F^s be a subgraph of G for scenario $s \in S$. For all $k \in V$, if there exist independent abstract orientations F_k^s of the edges of F^s , for $k = 1, \dots, |V|$, verifying simultaneously the following conditions in each F_k^s*

- (i) *There is no arc entering each referential node k ;*
 - (ii) *There is at most one arc entering a node $u \neq k$;*
- then F^s is an acyclic forest.*

Proof. Suppose that F^s contains a cycle $C = (V(C), E(C))$ for some scenario s , with $V(C)$ and $E(C)$ being the set of nodes and edges of C , respectively. We show that F^s cannot satisfy both conditions above for all $k \in V$ in scenario $s \in S$. To see this, first consider the nodes not in $V(C)$ as referential nodes and assume without loss of generality that they do not induce a cycle in F^s . For these nodes, if the arcs of the cycle C were symbolically oriented in the clockwise sense, both conditions (i) and (ii) should be satisfied. But when we consider the nodes in $V(C)$ as referential in any abstract orientation of the edges of $E(C)$, there are at least two arcs connected to any node of $V(C)$ (two leaving or two entering or one leaving and the other entering each node). We distinguish two possible situations. First, if all arcs connected to any referential node of C leave that node, then at least some other node in this cycle presents two arcs entering it, which violates the condition (ii). Second, if the arcs of C were symbolically oriented in the clockwise sense, then the condition (i) should be violated for all referential node in C . Thus, if F^s contain a cycle, there is no possibility of orienting the edges of F_k^s , $k = 1, \dots, |V|$, in order to satisfy the conditions (i) and (ii) simultaneously. Therefore F^s must be a forest, thus concluding the proof. \square

See that when F^s is a forest, then always exist abstract orientations of the edges of F_k^s , $k = 1, \dots, |V|$, satisfying the two conditions of Theorem 3.1.

Corollary 3.2 *If subgraph F^s in the Theorem 3.1 contains $|V(G)| - 1$ edges, then F^s is a spanning tree of G in scenario s .*

Now we present a new model for this problem. Consider the sets $E := E_S \cup E_D$ and $E^+ := E \cup \{uv \in V \times V \mid u \neq v, uv \notin E\}$. Note that E^+ contains all possible edges for G . Let, for all scenario $s \in S$, $x^s \in \{0, 1\}^{|E^+|}$ be the characteristic vector of a general solution (forest) F^s in scenario s for the SMWF problem with coordinates x_{uv}^s for all $uv \in E^+$. Define, for every $s \in S$, $k \in V$ and $uv \in E^+$, binary variables λ_{kuv}^s and λ_{kvu}^s , such that $\lambda_{kuv}^s = 1$ if edge uv belongs to the solution of the scenario s and, for the referential node k , node v precedes node u in an abstract orientation F_k^s of the edges of F^s ; and $\lambda_{kvu}^s = 0$, otherwise. The SMWF problem can be modeled as

$$(4) \quad (P_2) \quad \max \sum_{uv \in E_D} c_{uv}^1 x_{uv}^1 + \sum_{uv \in E_S, s \in S} \pi_s c_{uv}^s x_{uv}^s$$

$$\begin{aligned}
(5) \quad & s.t. \quad \lambda_{kuv}^s + \lambda_{kvv}^s = x_{uv}^s, \quad \forall s \in S, \forall k \in V, \forall uv \in E \\
(6) \quad & \sum_{v \in V - \{u\}} \lambda_{kuv}^s \leq 1, \quad \forall s \in S, \forall u, k \in V, u \neq k \\
(7) \quad & \lambda_{kkv}^s = 0, \quad \forall s \in S, \forall v, k \in V, k \neq v \\
(8) \quad & x_{uv}^s = 0, \quad \forall s \in S, \forall uv \in E^+ - E \\
(9) \quad & x_{ED}^1 = x_{ED}^2 = \dots = x_{ED}^P \\
(10) \quad & \lambda \in \{0, 1\}^{|V \times E^+ \times S|}, \quad x \in \{0, 1\}^{|S \times E^+|}
\end{aligned}$$

Constraints (5) state that if an edge uv is in a solution of the scenario s (i.e. $x_{uv}^s = 1$), then for all referential node $k \in V$ either v precedes u or u precedes v in F_k^s . Constraints (6) limit the number of predecessors of any node u in F_k^s to at most one, with $u \neq k$, in order to satisfy the condition (ii) of the Theorem 3.1. Constraints (7) state that no node v can precede the referential node k in F_k^s (the condition (i) of the Theorem 3.1). Constraints (8) fix at zero all the corresponding variables related to the extra edges we add to make G a complete graph. Note that the λ variables of each scenario (those associated with existing edges of E) induce an abstract orientation (one for each referential k) of the edges present in any feasible solution for problem (P_2) and these orientations satisfy the two conditions of the Theorem 3.1. Equalities in (9) are the non anticipativity constraints.

In this work the set of scenarios S has a constant number of realizations of the edges weights. Thus, as the number of constraints and variables of (P_2) is polynomial, then the resulting model is polynomial. For a non constant $|S|$, the model (P_2) is pseudo-polynomial.

Proposition 3.3 *The projection on the space of the x variables of an optimal solution $(\bar{\lambda}, \bar{x})$ for (P_2) , if it is non empty, is a set of forests, one for each scenario, maximizing (4).*

Proof. Suppose, by contradiction, that F^s is feasible for (P_2) and contains a cycle $C^s = (V(C^s), E(C^s))$ in some scenario $s \in S$ induced by the variables \bar{x}^s . For all the edges uv in $E(C^s)$ we have that $\bar{x}_{uv}^s = 1$. Thus, by (5), we must have $\lambda_{kuv}^s + \lambda_{kvv}^s = 1$, for all $k \in V(C^s)$ and for all $uv \in E(C^s)$. In particular, these λ^s must verify (6) and (7). But (see the proof of the Theorem 3.1) finding one evaluation for these variables means orienting the edges in $E(C^s)$ such that every referential node $k \in V(C^s)$ have no incoming arc and the remaining nodes in $V(C^s) - \{k\}$ have at most one incoming arc. Therefore, this is not possible and thus the related constraints correspond to an infeasible system of linear inequalities, contradicting the assumption that F^s is feasible for (P_2) . The reader may notice that when a solution F^s is a

forest, then there exist variables $\bar{\lambda}^s$ and \bar{x}^s whose values correspond to such solution. From the edges in F^s we obtain the \bar{x}^s values and if we apply an algorithm to remove iteratively the leaves from every sub-tree of the forest F^s we can identify which node can be considered as sub referential. If this deletion process reaches an edge, any extremity of this edge can be considered as the sub-referential node \hat{k} to obtain the required orientation with respect to the main referential node k . This orientation can be obtained from the \hat{k} -rooted pending sub-tree structure. The solution optimality follows from (4). \square

4 Computational experiments

Numerical experiments have been carried out on a Pentium IV, 1 GHz with 2G-RAM under windows XP. The source codes are generated with Matlab and the optimization models are solved by CPLEX 12. We report preliminary results for randomly (1–21) and arbitrarily (22–25) generated instances.

The dimensions of the instances are in the Table 1. (RP_1) and (RP_2) correspond to the linear relaxations of the models (P_1) and (P_2) , respectively. The first column identifies the instance number. Columns 2-5 provide the parameters $|V|$, $N = |E_D|$, $M = |E_S|$ and $P = |S|$. Columns 6-7 and 8-9 show the number of constraints and variables for (RP_1) and (RP_2) , respectively. For the model (RP_1) , the number of variables is of the order of $\mathcal{O}(N + MP)$ and the number of constraints, $\mathcal{O}(2^{|V|}P)$. For the model (P_2) , the number of variables and constraints are of the order of $\mathcal{O}(P|V|^3)$. In the first part of Table 2 we report numerical results for random generated instances. Column *Inst.* identifies the instance from the Table 1. Columns 2-3, 4-5, 6-7 and 8-9 give the optimal solution value and cpu time in seconds for (P_1) , (RP_1) , (P_2) and (RP_2) , respectively. We have that the optimal relaxed solutions are all integer. For the random generated instances, the data for columns 6 and 7 are equal to those for columns 8 and 9, respectively. This means that these random generated instances proved easy. The model (P_1) is very limited in solving those instance with more than 13 nodes. However, generally solving their corresponding linear relaxation (RP_1) takes less cpu time than by using the linear relaxation (RP_2) for these instances. Note that CPLEX finds integer solutions for (RP_1) in smaller cpu time than for (P_1) for these instances. In the second part of Table 2 we report numerical results for the arbitrarily generated instances of the Table 1. These instances intend to show the problem difficulty. These results give some insights about the complexity of this problem (this is an open question). Note that despite the reduced dimensions of these instances, and the fact we do not have many instances to

Inst.	Instance parameters				(RP_1)		(RP_2)	
	$ V $	N	M	P	# Constr.	# Var.	# Constr.	# Var.
1	5	5	5	5	130	30	675	550
2	5	5	5	50	1300	255	6750	5500
3	5	5	5	100	2600	505	13500	11000
4	8	14	14	5	1235	84	2730	2380
5	8	14	14	50	12350	714	27300	23800
6	8	14	14	100	24700	1414	54600	47600
7	10	22	23	5	5065	137	5285	4725
8	10	22	23	50	50650	1172	52850	47250
9	10	22	23	100	101300	2322	105700	94500
10	12	33	33	5	20415	198	9075	8250
11	12	33	33	50	204150	1683	90750	82500
12	12	33	33	100	408300	3333	181500	165000
13	15	52	53	5	163760	317	17585	16275
14	15	52	53	50	1637600	2702	175850	162750
15	15	52	53	100	3275200	5352	351700	325500
16	20	95	95	5	5242775	570	41325	38950
17	20	95	95	10	10485550	1045	82650	77900
18	20	95	95	25	26213875	2470	206625	194750
19	30	217	218	5	5368708965	1307	138110	132675
20	30	217	218	10	10737417930	2397	276220	265350
21	40	390	390	5	5497558138675	2340	325650	315900
22	6	3	6	3	-	171	21	684
23	8	4	10	4	-	988	44	2144
24	8	4	11	5	-	1235	59	2680
25	14	6	30	4	65476	126	11308	10556

Table 1
Dimensions of the instances.

Inst.	(P_1) cpu(s)		(RP_1) cpu(s)		(P_2) cpu(s)		(RP_2) cpu(s)	
	Random generated instances							
1	163.0000	0.34	163.0000	0.32	163.0000	0.36	163.0000	0.36
2	178.6012	0.36	178.6012	0.36	178.6012	1.50	178.6012	1.50
3	176.4886	0.41	176.4886	0.41	176.4886	2.94	176.4886	2.94
4	460.0961	0.42	460.0961	0.39	460.0961	0.53	460.0961	0.53
5	395.5182	0.81	395.5182	0.75	395.5182	3.98	395.5182	3.98
6	460.7503	3.38	460.7503	1.36	460.7503	12.91	460.7503	12.91
7	585.3537	0.73	585.3537	0.69	585.3537	0.78	585.3537	0.78
8	589.5790	5.22	589.5790	2.66	589.5790	6.81	589.5790	6.81
9	612.3388	14.06	612.3388	5.50	612.3388	27.20	612.3388	27.20
10	808.9535	2.97	808.9535	2.36	808.9535	0.92	808.9535	0.92
11	809.6658	44.58	809.6658	19.00	809.6658	28.12	809.6658	28.12
12	855.6202	107.17	855.6202	37.00	855.6202	57.48	855.6202	57.48
13	1182.2429	44.23	1182.2429	27.05	1182.2429	2.83	1182.2429	2.83
14	-	-	-	-	1142.6433	161.94	1142.6433	161.94
15	-	-	-	-	1108.7019	1025.52	1108.7019	1025.52
16	-	-	-	-	1616.3587	15.67	1616.3587	15.67
17	-	-	-	-	1586.4263	37.13	1586.4263	37.13
18	-	-	-	-	1579.3486	759.53	1579.3486	759.53
19	-	-	-	-	2537.2853	254.66	2537.2853	254.66
20	-	-	-	-	2649.1216	3425.19	2649.1216	3425.19
21	-	-	-	-	3540.0685	4682.42	3540.0685	4682.42
Arbitrarily generated instances								
22	41	1.484	43.500	0.422	41	0.011	43.500	0.010
23	94	0.515	97.333	0.391	94	0.028	97.333	0.006
24	118	0.406	120.500	0.391	118	0.032	120.500	0.006
25	202	14.375	204.500	11.563	202	0.110	204.500	0.044

"-": No solution found due to CPLEX shortage memory.

Table 2
Numerical results for random and arbitrarily generated instances.

perform exhaustive numerical experiments, the new compact model obtains all their optimal solutions in considerable fewer cpu time than the model (P_1) .

5 Conclusion

In this paper we propose a polynomial size formulation for the stochastic maximum weight forest problem. This formulation is based on the one for the spanning tree polytope of complete undirected graphs [4]. We extend the model in [4] to deal with forests in non complete graphs. For this, we propose a new theorem characterizing forests in undirected graphs that is important to prove the correctness of the new model.

The numerical experiments evidence that this problem can be NP-hard. Thus, further research will be dedicated to answer this open question concerning the complexity of the SMWF problem.

Acknowledgments

Thanks to the Chilean National Commission for Scientific and Technological Research (Conicyt grant 79100020) and to the Brazilian National Council for the Research and Development (CNPq grant 201347/2011-3) for supporting this work. We are grateful to the referees by their valuable comments to improve this paper.

References

- [1] Conforti, M., and G. Cornuéjols, and G. Zambelli, *Extended Formulations in Combinatorial Optimization*, 4OR - A Quarterly Journal of Operations Research **8** (2010), 1–48.
- [2] Edmonds, J., *Matroids and the greedy algorithm*, Mathematical Programming **1** (1971), 127–136.
- [3] Letournel, M., and A. Lisser, and R. Schultz, *Is the polytope associated with a two stage stochastic level problem TDI?*, Université de Paris Sud - LRI, Report RR-1546, 2010.
- [4] Martin, R. K., *Using separation algorithms to generate mixed integer model reformulations*, Operations Research Letters **10(3)** (1991), 119–128.
- [5] Pulleyblank, W. R., “Polyhedral combinatorics,” In Nemhauser et al., editor, Optimization, volume 1 of Handbooks in Operations Research and Management Science, North-Holland, chapter 5, 371–446, 1989.
- [6] Yannakakis, M., *Expressing Combinatorial Optimization Problems by Linear Programs*, Journal of Computer and System Sciences, **43(3)** (1991), 441–466.

Strategic planning in LTL logistics – increasing the capacity utilization of trucks

J. Fabian Meier^{1,2}

*Institute of Transport Logistics
TU Dortmund, Germany*

Uwe Clausen³

*Fraunhofer Institute for Material Flow and Logistics
& Institute of Transport Logistics
TU Dortmund, Germany*

Abstract

A “less than truckload” (LTL) network organises the transport of small shipping volumes by truck between given depots. To be cost-efficient it is necessary to bundle and unbundle goods on their way, using depots as so-called *hubs*. Our aim is to develop a strategic plan which is cost-optimal for given average shipping volumes. We consider transshipment and transport costs; to give a realistic estimate of the economies of scale, we charge each truck on a specific route equally, whether it is full or (nearly) empty.

Real-sized problems become too hard for standard solvers so that we develop a combination of heuristic strategies (which can, in the end, be combined with solvers like CPLEX). We consider the problem in two flavours: MAPIT requires to transport unsplit goods from one depot to another, using at most two intermediate depots as hubs. IO-MAPIT furthermore considers the circulation of trucks.

Keywords: Heuristics, IP, Hub Location, LTL Logistics, Economies of Scale

1 Introduction

LTL networks form the main leg of many transport networks; they organise the carriage of (usually small) shipping volumes between given depots, often over large distances. To increase capacity utilization, the goods are usually not transported directly, but bundled and unbundled at specially equipped depots which we then call hubs. Maintaining these hubs generates costs (depending on the transshipment capacity which is provided) which have to be balanced with the transportation costs. A general discussion of hub location problems can, e.g., be found in [1].

In the following discussion we consider a set D of n of depots distributed all over Europe with given average shipping volumes between them. All depots have the capability to act as hubs; we have to pay for the installed transshipment capacity and the actual turnover costs by a given factor t . We have two further practical requirements:

- Each shipping volume may be turned over at most twice (but may also be sent directly if this is cheaper).
- Each shipping volume may only use one route (so it cannot be split and transported on different routes between the same origin and destination).

Since the cost of using a truck hardly depends on its load, we measure the transportation costs by the number of trucks on each route. We consider only one type of truck. Our aim is to find a routing of all shipping volumes which is cost-optimal. It is not meant to be a vehicle routing plan on a day-to-day basis; it should provide the information for strategic decisions as the planning of hub capacities, truck numbers and truck circulation.

Mathematically, the decisions in the problem can be described by n^4 binary variables $\text{Route}(i, k, l, j)$ which indicate how the shipping volumes $w(i, j)$ are routed over two depots k and l ($k = l$ or $k = l = j$ represent one-stop or no-stop routes). We introduce integer variables $\text{Truck}(k, l)$ which describe the number of trucks on the connection $k \rightarrow l$, determined by the chosen Routes .

There are also other models which measure vehicles by integer variables and have only approximately n^3 variables (see e.g. [2,3]). These flow-based models allow splitting of the shipping volumes $w(i, j)$ and routing over an arbitrary number of hubs. Although these assumptions stem from rail and

¹ This research is part of the DFG project CL 318/14-1 “Lenkung des Güterflusses in durch Gateways gekoppelten Logistik-Service-Netzwerken mittels quadratischer Optimierung”

² Email: meier@itl.tu-dortmund.de

³ Email: Uwe.Clausen@iml.fraunhofer.de

air traffic and are not realistic for LTL, these 3-index-models can provide relaxations of our model; hence their solutions are lower bounds.

In contrast to classical models, our model is undoubtedly more realistic than just “discounting” the hub-hub connections by a fixed factor. But:

- For realistic n (in the range from 25 to 100) we get a huge number of integer variables, so that standard MIP solvers give up quickly.
- The LP relaxation is very poor. Actually, without any further restrictions, the LP solution is just given by transporting every $w(i, j)$ directly and charging it with the appropriate fraction of a truck size. Hence it is difficult to get a useful lower bound. Some preliminary results to improve this situation are presented in [2.1](#).

Our approach is as follows. Starting from the “direct transportation” solution, we apply three different heuristic improvement strategies after each other (see section [3](#)):

- Firstly, a tree-based method which quickly gives a medium-good solution
- Secondly, a “destroy and locally repair” strategy
- Thirdly, a strict descend to a local optimum

This approach produces a good solution to start an optimization in CPLEX; furthermore, it gives us information about the “quality” of the possible hubs. We use this information to restrict the number of possible **Routes** to those expected to be “good”. We never delete **Routes** belonging to our initial solution so that it is still valid in the restricted model. This restricted problem can then be given to CPLEX (see [3.3](#)).

We will also shortly glance at the problem of truck circulation. In realistic scenarios every truck should eventually come back to its home depot. While it is easy to state this property in the mathematical description ($\text{incomingtrucks}(i) = \text{outgoingtrucks}(i)$ for $i \in D$), it is difficult to include this behaviour in the heuristics.

2 The models MAPIT and IO-MAPIT

In this section we define our model, which we call MAPIT: The “Multi-Allocation Problem with Integer Trucks”. For the set D of depots, we define (with i, j, k, l always from D):

- $w(i, j)$ is the shipping volume (the truck capacity is set to 1).
- $c(i, j)$ is the cost for one truck to get from i to j . We assume the triangle

inequality for $c(i, j)$ since every truck should always choose the cost-optimal connection between two given points.

Furthermore, we charge the transshipment costs with a factor t . The most important variable is $\text{Route}(i, k, l, j)$ which is binary and indicates the route for the shipping volume from i to j (k and l may be equal and/or equal to j to indicate transport over less than two hubs). The **Routes** are used to calculate the variables $\text{Transport}(i, j)$ for the actual transport volume and the integer variables $\text{Truck}(i, j)$ for the number of trucks to be used. Then the model looks like this :

$$\text{Minimize} \quad \sum_{i,j} \text{Truck}(i, j) \cdot c(i, j) + \sum_{i,j} (\text{Transport}(i, j) - w(i, j)) \cdot t \quad (1)$$

$$\begin{aligned} \text{Transport}(i, j) = & \sum_{k,l} \text{Route}(i, j, k, l) \cdot w(i, l) + \sum_{k,l} \text{Route}(k, l, i, j) \cdot w(k, j) \\ & + \sum_{k,l} \text{Route}(k, i, j, l) \cdot w(k, l) \end{aligned} \quad (2)$$

$$\text{Truck}(i, j) \geq \text{Transport}(i, j) \quad (3)$$

$$\sum_{k,l} \text{Route}(i, k, l, j) = 1 \quad (4)$$

$$\text{Transport}(i, j) \in \mathbb{R}_{\geq 0}, \quad \text{Truck}(i, j) \in \mathbb{Z}_{\geq 0}, \quad \text{Route}(i, k, l, j) \in \{0, 1\} \quad (5)$$

The model IO-MAPIT has one additional constraint:

$$\sum_k \text{Truck}(k, l) = \sum_k \text{Truck}(l, k) \quad \text{for all } l \quad (6)$$

The objective function (1) consists of transportation costs (first part) and transshipment costs (second part). In constraint (2), we determine the total transport volume on each edge, consisting of three sums for the three (possible) parts of a **Route**. The inequality (3) determines the number of trucks (from the transport volume). Finally, (4) indicates that only one **Route** is used. The additional requirement (6) keeps the number of trucks at each depot constant, meaning that we could handle the **Routes** with round-trips for each truck.

We want to remind the reader that although we did not mention the transshipment capacities in the model and did not introduce special “hub variables”, the costs for building and maintaining transshipment capacities at

each depot are included in the cost parameter t assuming a simplified linear cost structure. More complex cost structures make the MIP model harder but could be easily incorporated into the heuristics explained in section 3.

2.1 LP relaxation, lower bounds and possible improvements

The optimal LP solution of MAPIT is given by direct transport for every commodity with the objective value $\sum_{i,j \in D} c(i,j) \cdot w(i,j)$. As most of our shipping volumes are smaller than one tenth of a truck, rounding the LP solution increases the objective function by a factor of ten to twenty. How can we improve this?

For every subset $I \subset D$ we can compute the minimal amount that must be shipped from I to $D \setminus I$ using the given values $w(i,j)$. We then generate a lower bound for the number of trucks between these sets by rounding up the minimal shipping volume. Especially, if we divide the set of depots into 1 and $(n - 1)$ depots or 2 and $(n - 2)$ depots, these inequalities give the LP relaxation additional strength. Nevertheless, the LP relaxation is still weak.

A more fruitful approach could be derived from the 3-index-model (mentioned in the introduction) allowing splitting and arbitrary rerouting. If we further relax the scenario by replacing the truck cost step function by a piecewise linear function, we can drastically improve the branch-and-cut solvability. We aim to give numerically evidence for this in future publications.

3 The heuristics

Our initial heuristic consists of three steps which are detailed below. The first two use parallel (sometimes interacting) threads; we prefer between 4 and 8 threads since less would increase the risk of running into local minima and more would lead to significantly longer running times (as modern computers have about 4 to 8 CPU kernels).

3.1 The three steps

Tree heuristic

The tree heuristic is based on the additional assumption that the transport of all $w(i,j)$ for a fixed destination j forms a “sending tree”. This means that shipping volumes for j that meet at one hub will never be split again. Computationally, this has the advantage that we can represent **Route** by a data structure consisting of n trees; hanging a subtree to a different node is an $O(1)$ operation but enables us to reroute a lot of traffic.

We use an improvement strategy: We start from an arbitrary solution (like the rounded LP solution), and search for edges where empty space is transported over large distances. These “bad edges” are checked in every of the n trees: If we could save costs by avoiding the given edge we reroute the respective subtree. The cost estimates are mildly randomized to avoid local minima. The tree heuristic quickly produces reasonable solutions (e.g. less than 30% gap for $n = 60$) and provides the basis for the next step.

Local-Global heuristic

We call a single route $i \rightarrow k \rightarrow l \rightarrow j$ for a shipping volume $w(i, j)$ *locally optimal*, if changing just this route cannot decrease the overall costs. Furthermore, in our given solution we calculate *individual costs* for each shipping volume $w(i, j)$, meaning the costs we would save by not transporting $w(i, j)$. The general idea of the Local-Global heuristic is now given by repeating the following two steps:

- (i) Remove shipping volumes with high individual costs from the solution; furthermore, remove some arbitrarily chosen shipping volumes.
- (ii) Find new routes for the removed shipping volumes (one after the other); the routes are chosen to be locally optimal (in the sense defined above). Since our routes have at most three edges, we can find such a route in $O(n^2)$, but a clever saving algorithm for already computed values can drastically speed up the procedure in the average case.

We do not only save the best solutions but maintain a solution pool. Unsuccessful threads choose from this pool to keep up with the better threads.

Local improvement

In the last step we undertake minor improvements until our solution is locally optimal, i.e. changing only one Route cannot decrease the overall costs. Coming from the strongly optimized solution of the local-global heuristic, this is usually fast. From the complexity point of view, this is identical to part 2 of the local-global heuristic applied to just one route in each step.

3.2 IO-MAPIT

Our heuristic approaches do not include the circulation of trucks. The easiest way to add this aspect is to take the heuristic solution of MAPIT and balance it with the help of a min-cost flow. In our test data, we got reasonable results with this approach but in general this may not be the case.

Instead of introducing “artificial shipping volumes”, which was difficult, we prefer to let CPLEX do this correction afterwards (see also section 4).

3.3 Giving the solution to CPLEX

If we take our heuristic result as initial solution for CPLEX we cannot expect CPLEX to improve it (for high n) since the gap is still high. Hence, we delete a lot of **Routes** from the model which are probably not helpful (our equipment can handle about 250,000 **Routes**, meaning that we need further restrictions if $n > \sqrt[4]{250,000} \cong 22$). We apply two strategies:

- Avoid **Routes** which are “long” compared to the direct connection.
- Ignore **Routes** which include turnover at hubs with low quality: By this we mean hubs at which nearly no transshipment happens and which are therefore probably far away from the main connections.

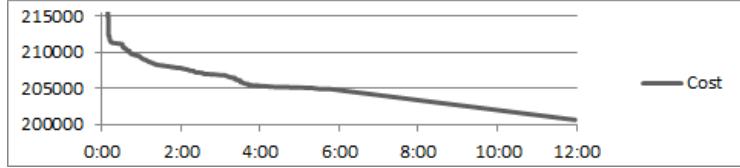
Together with our initial solution we give CPLEX the chance to apply its branch-and-cut strategies effectively. The numerical results are discussed below, also for IO-MAPIT.

4 Experiments

We use a real world data set for 60 depots, from which we derive scenarios I_n with $n = 15$, $n = 30$ and $n = 60$. The full data set (I_{60}) is the most interesting case since CPLEX completely fails on it and therefore heuristics become necessary. At first we test good parameter choice by running four different configurations over relatively short time periods (20 minutes). To incorporate statistical spread we made 10 tests each and give the average objective value together with the maximal deviation:

	Best found for I_{60}	Short tree heuristic	Long tree heuristic
High thread interaction	211,167 ($\pm 1.6\%$)	211,038 ($\pm 2.0\%$)	
Low thread interaction	210,751 ($\pm 2.4\%$)	210,293 ($\pm 0.8\%$)	

We see that the method is stable and only mildly depends on the choice of the parameters. For our long time experiments we combine 6 hours of heuristic with 6 hours of CPLEX as described in section 3.3. For I_{60} , we show the development of the solution in figure 1. For the table we use the configuration described above. The “Pure CPLEX” indicates the result we get by applying CPLEX directly to our mathematical model (12h runtime). For IO-MAPIT



	Instance 15	Instance 30	Instance 60	IO-MAPIT for I_{60}
Pure CPLEX	45,630	111,587	951,424	none
Best found	49,027	102,494	200,635	227,967
Lower bound	44,456	86,645	139,991	167,918
Gap	9%	15%	30%	26%

Fig. 1. Experimental results

we took the same heuristic, corrected the result by a min-cost flow and gave it to CPLEX. Unfortunately, it is difficult to compare the results to those from the literature: The “usual” models consider linear (rather than truck-based) transportation costs with a discount factor for hub-hub-connections. Some papers from the Nineties (like [2]) have comparable models, but the computational results are outdated. The recent work [3] has a comparable model, but considers only problem sizes up to 100,000 decision variables (like our “Instance 15”), for which a gap of about 20% is encountered.

5 Conclusion

The experiments show that the heuristics have good strength for short and long running times. Our aim is to incorporate them into more complex models including time slices. Furthermore we need to raise the lower bounds to get a better theoretical understanding of the problem.

References

- [1] Campbell, J.F. and M.E. O’Kelly, *Twenty-five years of hub location research*, Transportation Science, **46**, 2, (2012), 153–169,
- [2] Jaillet, P. and G. Song and G. Yu, *Airline network design and hub location problems*, Location Science, **4** (1996), 195–212,
- [3] Sender, J. and U. Clausen, *Hub location problems with choice of different hub capacities and vehicle types*, Network Optimization (2011), 535–546,

Branch-and-Price for a European variant of the Railroad Blocking Problem

Robert Voll¹

*Institute of Transport Logistics
TU Dortmund
Germany*

Uwe Clausen²

*Fraunhofer Institute for Material Flow and Logistics
& Institute of Transport Logistics
TU Dortmund
Germany*

Abstract

In wagonload traffic, a production form in railway freight traffic, small groups of wagons have to be transported. In order to decrease transportation costs, wagons from different relations are consolidated. In railyards trains can be separated and rearranged to new trains. The costs arising from this process, which is called reclassification, must be balanced with the transportation costs. The minimization of total costs can be formulated as a network optimization model. We introduce a branch-and-price approach for the considered problem. Moreover, we present specialized cuts, which can be incorporated into the branching scheme. Solutions obtained from our implementation can keep up with results computed by CPLEX.

Keywords: column generation, branch-and-price, railway freight traffic, optimization, railroad blocking problem

1 Introduction

Railway is one of the most important modes of transport world-wide. Despite all liberalization efforts and a stronger awareness of ecological needs, railroad freight traffic was not able to increase its share in European modal split significantly in the last decades. Especially wagonload traffic is confronted with strong economic issues. In wagonload traffic a customer orders the transport of single railcars or a small group of railcars. Due to the small number of railcars, direct transport is rarely economic. Therefore, it is reasonable to consolidate cars from different relations, i.e., origin-destination pairs, to decrease transportation costs. Trains can be separated and rearranged to new trains in reclassification yards [5]. This process also results in high costs. The sequence of yards for each railcar is determined by so called blocking plans. Transportation and reclassification costs depend on the blocking plan.

The generation of efficient blocking plans can be modelled as a large-scale network optimization problem and is referred to as the Railroad Blocking Problem (RBP). The resulting models have a multicommodity flow structure with elements of network design problems. We extend a column generation approach published earlier [12] to a branch-and-price algorithm (BaP).

The remainder of this paper is structured as follows: Section 2 gives a very short survey on optimization of blocking plans. In section 3 our model is presented, followed by a description of our BaP approach in 4. Afterwards, we introduce cuts for the presented model in 5. Section 6 provides numerical results. The paper is concluded by a summary.

2 Survey

Several contributions in the field of blocking plans deal with North American railway systems ([1,4,7,11]). In contrast to Europe, the networks are characterized by a strict separation of passenger and freight traffic. Due to mainly infrastructural reasons, freight trains are limited in total length and weight in Europe. Consequently, corresponding constraints are incorporated into models designed for application in Europe ([6,9]). A Model for Iranian([13]) railways has recently been published. The differences between North American and Chinese planning strategies are explained in [10].

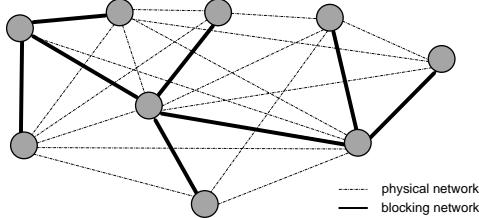
Most of the solution approaches are customized to the particular models and can hardly be transferred to other models.

¹ Email: voll@citl.tu-dortmund.de

² Email: uwe.clausen@iml.fraunhofer.de

3 Model

The RBP can be modeled as a multicommodity capacitated network design problem (MCNDP). The underlying problem of determining a sequence of yards for each railcar is a routing problem. Nevertheless, it has strong network design characteristics. The considered graph \mathcal{G} is complete and consists of the reclassification yards \mathcal{N} and railway connections \mathcal{A} between them. While finding an optimal routing for all railcars, we implicitly want to find the so called blocking network, which is a subset of the physical network, see graphic 1. The arcs of this blocking network are established by trains in the physical network.



Graphic 1: blocking network

Operating a train induces high fixed costs, which dominate the transport costs in European wagonload traffic. So, the cost function must focus rather on train costs than on costs per wagon. Hence, we introduce two kinds of decision variables: path variables λ_p^k and design variables n_{ij} . The path variables select a particular path p for relation k , i.e. wagons with a common origin $o(k)$ and destination $d(k)$, from the set of all feasible paths $\mathcal{P}(k)$. A path is feasible for a relation, if it connects origin and destination of the relation and does not traverse more than S_k arcs. The design variables give the number of trains established on each arc.

$$\lambda_p^k = \begin{cases} 1, & \text{if relation } k \text{ uses path } p \in \mathcal{P}(k) \\ 0, & \text{otherwise} \end{cases} \quad (1)$$

$$n_{ij} = \text{number of trains established on arc } (i,j) \quad (2)$$

Moreover, some parameters are necessary. Due to aspects of robustness, we do not look for a solution for a single day, but for a blocking plan, which works well on a couple of days \mathcal{D} . Accordingly, particular demand parameters for each day of the planning horizon are given. Further information about this

setup can be found in [6]. A path $p \in \mathcal{P}(k)$ can be characterized by the arcs (i, j) it traverses. So, x_{ij}^{kp} is one, iff (i, j) is an arc on path $p \in \mathcal{P}(k)$. The transportation costs for a train on arc (i, j) and the classification costs per wagon in yard i are given by c_{ij} and U_i , respectively. The parameters v_k^d , l_k^d , and w_k^d give the number of wagons on relation k on day d and their aggregated lengths and weights, respectively. W_{ij} is the maximal train weight allowed on (i, j) and, accordingly, L_{ij} the maximal length. Parameters R_i restrict the number of trains which can be built in yard i per day. The following optimization model (OPT_{path}) is defined:

$$\min_{x,n} \sum_d \sum_{i,j} [c_{ij}n_{ij} + \sum_k U_i \sum_{p \in \mathcal{P}(k)} v_d^k \lambda_p^k x_{ij}^{kp}] \quad (3)$$

$$s.t. \quad \sum_k l_d^k \sum_{p \in \mathcal{P}(k)} \lambda_p^k x_{ij}^{kp} \leq L_{ij} n_{ij} \quad \forall d \in \mathcal{D} \quad \forall (i, j) \in \mathcal{A} \quad (4)$$

$$\sum_k w_d^k \sum_{p \in \mathcal{P}(k)} \lambda_p^k x_{ij}^{kp} \leq W_{ij} n_{ij} \quad \forall d \in \mathcal{D} \quad \forall (i, j) \in \mathcal{A} \quad (5)$$

$$\sum_j n_{ij} \leq R_i \quad \forall i \in \mathcal{N} \quad (6)$$

$$\lambda_p^k \in \{0, 1\} \quad \forall k \in \mathcal{K} \quad \forall p \in \mathcal{P}(k) \quad (7)$$

$$n_{ij} \in \mathbb{N} \quad \forall (i, j) \in \mathcal{A} \quad (8)$$

The objective function sums up train and reclassification costs. Constraints (4) and (5) reflect the aforementioned restrictions on trains length and weight, respectively. The reclassification capacity of each node is measured by the number of outgoing trains in (6). All path variables must be binary, because relations shall not be splitted, and the number of trains on an arc should be integral. The model (OPT_{path}) is equivalent to a compact arc-based formulation (OPT_{arc}) with arc-flow variables x_{ij}^k . The following transformation shows their equivalence.

$$x_{ij}^k = \sum_{p \in \mathcal{P}(k)} \lambda_p^k x_{ij}^{kp} \quad (9)$$

The decision problem corresponding to (OPT_{path}) can be proven to be \mathcal{NP} -complete, because it includes the integer multicommodity flow problem (IMCFP) as a special case. Following Garey and Johnson ([8]), IMCFP is known to be \mathcal{NP} -complete even for two commodities.

4 Branch-And-Price

(OPT_{path}) can hardly be solved by standard solvers. Small instances can be solved to optimality, but medium size instances (up to ~ 250 relations) result in excessively high run times. For real-world instances, it was not even possible for CPLEX 12.4 to solve the LP in the root node. So, we developed a column generation (CG) algorithm in [12], which was reasonable, because our problem has a block diagonal structure, which is vital to CG (e.g. [3]). A set of paths for each relation could be generated in polynomial time using the Bellman-Ford shortest path algorithm. Our implementation outperforms the CPLEX LP solver, although it was not able to solve real-world instances, either. However, we are not (only) interested in LP solution, but want to solve the integer problem. So, it is necessary to embed CG into a BaP framework. The success of BaP strongly depends on the branching decisions. Thus, we want to discuss, which branching is reasonable for (OPT_{path}) . The branching must incorporate two aspects. On the one hand, the branching should have an impact on the objective function value of the LP. Otherwise the size of the branching tree can hardly be restricted. On the other hand, decisions should lead the search into branches with promising solutions. Therefore, it is important to eliminate fractional solutions. Considering the structure of (OPT_{path}) , a branching on the train number variables n_{ij} seems reasonable, because of their strong impact in the objective function. Unfortunately, our tests showed that it does not help to eliminate fractional solutions. Hence, no good primal solutions were found. Branching on the path variables has undesired effects, which lead to an unsymmetric partition of the decision tree, as stated in [2]. Using transformation (9), the equivalence of compact and extensive formulation, we can branch on the arc-flow variables x_{ij}^k of the compact formulation. Fixing a variable x_{ij}^k to zero is easy. We can simply delete the arc from the graph, when generating new paths. Fixing $x_{ij}^k = 1$ is complicated, because it forces a relation to use the particular arc (i, j) . The resulting constrained shortest path problem cannot be solved in polynomial time anymore. We overcome this problem by solving the pricing problem by dynamical programming instead of Bellman-Ford algorithm. This is not difficult, because paths, which must include certain arcs, can be composed from regular shortest paths.

One more decision is important for the implementation of Branch-and-Price: Which sequence of x_{ij}^k is used for branching? We follow an idea of Barnhart et al. ([2]). The largest relation k_{max} with fractional path variables $\lambda_p^{k_{max}}$ is chosen. The size of a relation can be measured by the number of trains induced by it. It can be computed from the demands l_d^k and w_d^k . We search

for the first yard i , where the flow of the relation is split up and branch on the arc (i, j) with the highest amount of flow of relation k_{max} .

5 Cuts

Using the branching scheme mentioned above, it is possible to incorporate problem tailored cuts. With respect to constraints (4) and (5) it is obvious that trains are needed on an arc which is used by at least one relation. If x_{ij}^k is fixed to one, it is necessary to establish at least enough trains on (i, j) to transport the wagons of k . Denoting the set of relations which are forced to use (i, j) by branching by \mathcal{K}_{fix} , we obtain the following cuts:

$$n_{ij} \geq \max_{d \in \mathcal{D}} \left\lceil \left\{ \frac{\sum_{\mathcal{K}_{fix}} l_d^k}{700}, \frac{\sum_{\mathcal{K}_{fix}} w_d^k}{1600} \right\} \right\rceil \quad (10)$$

Cuts (10) include the following well-known cuts (11) from multicommodity network design as a special case.

$$n_{ij} \geq x_{ij}^k \quad (11)$$

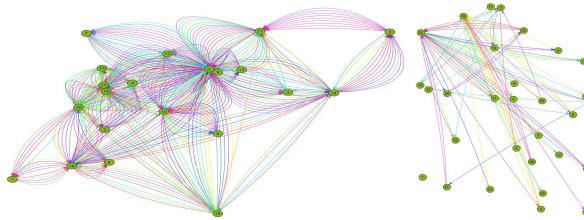
6 Numerical Results

Combining BaP with the cuts (10), we obtained a Branch-and-Price-and-Cut (BaPaC) algorithm. BaPaC was implemented using CPLEX LP solver for the RMPs as described in [12]. We compare our BaPaC approach to CPLEX on the arc-based model (OPT_{arc}). The results can be found in Table 1. The instances are derived from scenarios provided by Deutsche Bahn. They vary in terms of network size and density of demands, i.e., the ratio of relations with demand and the number of OD-pairs in the network. That value is an indicator for the possibility of flow consolidation, because a higher density permits more synergetic effects between relations, see Graphic 2. The column ‘Root’ contains the gap resulting from BaPaC using only the paths generated in the root node of the tree. The columns ‘BaPaC’ and ‘CPLEX’ give the relative gaps of the best solutions obtained by our BaPaC and CPLEX 12.4, respectively. They are compared to the best lower bound (LB) provided by CPLEX. Due to its cutting planes, CPLEX mostly obtains tighter bounds than other approaches we tested. All computations were executed on a mod-

ern workstation computer.

Table 1: Computational results

#nodes/ #relations	density of demands	time limit	Root	BaPaC	CPLEX
30/65	7,4 %	7200s	2,00%	<u>1,55%</u>	1,76%
50/224	9,1 %	21600s	<u>23,53</u>	<u>23,53%</u>	no feas
21/254	60,5 %	21600s	21,97%	<u>14,08%</u>	14,21%
21/254	60,5 %	21600s	53,64%	25,38%	<u>22,35%</u>



Graphic 2: visualization of first and last instances' solutions

7 Conclusions

We presented a model for a network design problem arising from railway freight traffic. Incorporating a column generation scheme into a branch-and-bound framework, we obtained a branch-and-price method. We discussed several ways of branching and traversing the tree. Moreover, we were able to derive cuts, which fit into the branching scheme. The solutions computed by our implementation could keep up with the solutions provided by CPLEX. Nevertheless, we are neither able to solve medium-sized instances to optimality nor to find feasible solutions for larger instances. Exact approaches are not appropriate for solving real-world instances. Hence, our future efforts will be concentrated on finding heuristic methods.

Acknowledgement

Presented results arose mostly from our project “A User-Guided Planning-Tool For Car Grouping in Railroad Freight Traffic” with Deutsche Bahn AG funded by German Federal Ministry for Science and Education (01IC10L26A).

References

- [1] R.K Ahuja, K. C. Jha, and J. Liu. Solving real-life railroad blocking problems. *Interfaces : INFORMS journal on the practice of OR*, 37(5):404–419, 2007.
- [2] C. Barnhart, C. Hane, and P. Vance. Using branch-and-price-and-cut to solve origin- destination integer multicommodity flow problems. *Operations Research*, 2000(Vol. 48, No. 2):318–326, 2000.
- [3] C. Barnhart, E.L Johnson, G.L Nemhauser, M.W.P Savelsbergh, and P. Vance. Branch-and-price: Column generation for solving huge integer programs. *Operations Research*, 1998(Vol. 46, No. 3), 1998.
- [4] L.D Bodin, B.L Golden, and A.D Schuster. A model for the blocking of trains. *Transportation Research B*, 1980(Vol. 14B):115–120, 1980.
- [5] M. Bohlin, F. Dahms, H. Flier, and S. Gestrelius. *Optimal Freight Train Classification using Column Generation*. 12th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems, 2012.
- [6] U. Clausen and R. Voll. A blocking model with bundling effects respecting multiple od-matrices. *Proceedings of the 4th IC-EPSMSO*, 2011(1):5–12, 2011.
- [7] J.-F. Cordeau, P. Toth, and D. Vigo. *A survey of optimization models for train routing and scheduling*. Groupe d'études et de recherche en analyse des décisions, Montréal, 1998.
- [8] M.R Garey and D. S. Johnson. *Computers and intractability: A guide to the theory of NP-completeness*. Freeman, New York and NY, 19 edition, 1997.
- [9] H. Homfeld. *Consolidating car routes in rail freight service by discrete optimization*. München, 2012.
- [10] B. Lin, Z. Wang, L. Ji, Y. Tian, and G. Zhou. Optimizing the freight train connection service network of a large-scale rail system. *Transportation Research Part B: Methodological*, 46(5):649–667, 2012.
- [11] H.N Newton, C. Barnhart, and P. Vance. Constructing railroad blocking plans to minimize handling costs. *Transportation Science*, 1998(Vol. 32, No.4), 1998.
- [12] R. Voll and U. Clausen. Column generation for multi-matrix blocking problems. In *Operations research proceedings 2011 - Selected papers of the International Conference on OR*, pages 281–286.
- [13] M. Yaghini, A. Foroughi, and B. Nadjari. Solving railroad blocking problem using ant colony optimization algorithm. *Applied Mathematical Modelling*, (35):5579–5591, 2011.

Home Care optimization: impact of pattern generation policies on scheduling and routing decisions

Paola Cappanera¹

*Dipartimento di Ingegneria dell'Informazione
Università degli Studi di Firenze
Firenze, Italy*

Maria Grazia Scutellà²

*Dipartimento di Informatica
Università di Pisa
Pisa, Italy*

Abstract

In Home Care optimization, operators have to be assigned to patients by taking into account compatibility skill constraints, and patient visits have to be scheduled in a given planning horizon. Moreover, operator tours have to be determined. Integer Linear Programming models have been proposed which use the concept of patterns, i.e. a priori scheduling profiles, to combine the diverse decision levels. Computational results on real instances show that pattern generation policies are crucial to address scheduling and routing in large Home Care instances.

Keywords: Home Care, Integer Linear Programming, computational experiments.

¹ Email: paola.cappanera@unifi.it

² Email: scut@di.unipi.it

1 The problem

Nowadays, the ever increasing average age of population and the increased costs for the consequently required care, compel the medical care units (hospitals and so on) to offer Home Care Services in an attempt to limit costs. Even more important, medical treatments carried out at patients home impact favorably on their quality of life. Therefore, Home Care Services are a cost-effective and flexible instrument in the social system.

Here we address a relevant optimization problem arising in Home Care; specifically, given a planning horizon W , usually a week, a set of patients with an associated *care plan*, i.e. weekly requests each of them demanding a specific level of skill to be operated, and a set of operators also characterized by a specific skill, the problem asks to schedule the patient requests during the planning horizon, to assign the operators to the patients by taking into account the compatibility between request and operator skills, and to determine the tour each operator has to perform in every day of the planning horizon.

More formally, the Home Care Problem (HCP) under investigation is defined on a complete directed network $G = (N, A)$, having n nodes, where each node j corresponds to a patient. There is an extra node (node 0), which is used to denote the basis of the operators. A set K of \bar{k} levels of skill is assumed for both patients and operators, where skill \bar{k} corresponds to the highest ability and skill 1 to the lowest one. A care plan is associated with each patient j that specifies the number of visits required by j in the planning horizon W relatively to each skill level $k \in K$. Specifically, r_{jk} , with $k \in K$, gives the number of visits of skill k required by j in W . A set O of (skilled) operators is available in the planning horizon. In addition, a subset $O_d \subseteq O$ of the operators is available on day d , for each $d \in W$. A hierarchical structure of the skill levels is assumed for the operators, so that an operator with skill k can work all the requests characterized by a skill up to k .

In HCP the scheduling of the patient requests in W , the operator assignment and the routing decisions are offered through a new modelling device, called *pattern*. We assume in fact that the patient requests are operated according to a set P of a priori given patterns. In particular, for each pattern $p \in P$, we define $p(d) = 0$ if no service is offered at day d , while it is $p(d) = k$ if a visit of skill k is operated according to pattern p on day d .

Given the input data above, HCP thus consists in assigning one pattern from P to each patient j , so scheduling the requests of j during the planning horizon (*care plan scheduling*), in assigning operators to each patient j , for each day where a request of j has been scheduled (*operator assignment*), and

in determining the tour of each operator for each scheduled day (*routing decisions*). In addressing these three groups of decisions, the skill constraints (i.e. the compatibility between the skills associated with the patient requests and the skills of the operators) have to be taken into account. Other relevant Quality of Service requisites are considered.

An objective function typically used to guide the Home Care decisions is the balancing of the workload among the operators. Hence we maximize the minimum operator utilization factor, where the *operator utilization factor* is the total workload of the operator in W over his/her maximum possible workload.

In the state-of-the-art literature Home Care problems are usually solved in cascade: first the operators are assigned to the patients on a geographical basis; second, the schedule of each operator is determined, usually operator-wise. See for example [4]. Some Vehicle Routing Problem (VRP)-like formulations exist in the literature, which however generally deal with a daily planning horizon. To the best of our knowledge there are only three exceptions ([1,5,6]). However, no exact approach is proposed there to solve the overall problem, but two-stage solution approaches are presented. On the other hand, here HCP is solved by jointly addressing assignment, scheduling and routing decisions over W , by incorporating the skill hierarchy structure introduced in ([2]) where, however, only daily routing decisions have been addressed.

The HCP solution approach is based on new Integer Linear Programming (ILP) formulations, which have been proposed in [3]. As previously indicated, the innovative modelling device proposed to combine the various levels of decisions is the use of a priori given *patterns*. Three policies to generate patterns have been designed in [3] and will be discussed in this paper, by emphasizing their impact on the efficiency of the optimization approach and on the quality of the returned solutions.

2 Pattern generation policies

In order to formulate HCP, the main decision variables ([3]) are:

$$z_{jp} = \begin{cases} 1 & \text{if patient } j \text{ is assigned to pattern } p \\ 0 & \text{otherwise} \end{cases} \quad j \in N \ (j \neq 0), p \in P$$

$$x_{ij}^{td} = \begin{cases} 1 & \text{if operator } t \text{ uses } (i, j) \text{ on day } d \\ 0 & \text{otherwise} \end{cases} \quad (i, j) \in A, d \in W, t \in O_d$$

The proposed formulations to HCP include the following constraints:

$$\sum_{p \in P} z_{jp} = 1 \quad \forall j \in N \setminus \{0\} \quad (1)$$

$$\sum_{i \in N} \sum_t x_{ij}^{td} \leq \sum_{p: p(d) \geq 1} z_{jp} \quad \forall j \in N \setminus \{0\}, \forall d \in W \quad (2)$$

$$\sum_{i \in N} \sum_{t: s_t \geq k} x_{ij}^{td} \geq \sum_{p: p(d) = k} z_{jp} \quad \forall j \in N \setminus \{0\}, \forall d \in W, \forall k \in K \quad (3)$$

$$\sum_{i \in N} x_{ij}^{td} = \sum_{i \in N} x_{ji}^{td} \quad \forall j \in N \setminus \{0\}, \forall d \in W, \forall t \in O_d \quad (4)$$

Constraints (1) assure that each patient is assigned exactly to a pattern. Constraints (2) state that at most one operator per day can visit patient j , if a visit has been scheduled on that day for node j . By denoting with s_t the skill level of operator t , constraints (3) guarantee that, on day d , exactly one operator, of adequate skill, must visit patient j if a service has been scheduled for j on day d . This is true for each skill level k . In particular, the least skilled operators can perform only visits of skill 1 (case $k = 1$), whereas the most skilled operators can perform all types of visits (case $k = \bar{k}$). (4) are the classical flow conservation constraints on the routing variables.

(1)-(4) reveal that the pattern device has an impact on both scheduling and routing decisions, and therefore it is crucial for the efficiency of the overall optimization process, and also in determining the quality of the returned solutions. To investigate this impact three pattern generation policies have been analysed. *Heur* is a greedy heuristic procedure based on the frequency of the request types: it firstly orders the patient requests according to their numbers of requirements for increasing levels of skill and then, by scanning the ordered list, generates patterns that can accomplish with the frequency of such requests. *ImplSol* is based on the extraction of pattern information from the solution actually implemented at the Home Care provider. The third policy, called *FlowBased* or simply *FB*, is a multicommodity flow based approach defined on an auxiliary layered network $G_W = (N_W, A_W)$, with $|N_W| = n_w$, having one layer L_d for each considered day d in W , plus a source node (say 1) and a destination node (say n_w). Each layer is composed of $\bar{k} + 1$ nodes: node 0, which indicates that no visit is scheduled in the day corresponding to the layer, and a node k , for each $k \in K$, which represents the scheduling of a visit of skill k . In G_W there exists a directed arc from the source node to the nodes in first layer, from each node in the last layer to the destination node, and from each node in layer L_d to each node in the next layer, for each $d \in W$.

Any directed source-destination path in G_W corresponds to a potential pattern. Therefore, we introduce a binary commodity for each patient j , having node 1 as the origin and node n_w as its destination, and state the following multicommodity flow problem on G_W as a tool to generate patterns:

$$\min \sum_{(h,i) \in A_W} q_{hi}$$

$$\sum_{(h,i) \in A_W} f_{hi}^j - \sum_{(i,h) \in A_W} f_{ih}^j = \begin{cases} -1, & \text{if } i = 1, \\ 1, & \text{if } i = n_w, \\ 0, & \text{otherwise} \end{cases} \quad \forall i \in N_W, \forall j \in N \setminus \{0\} \quad (5)$$

$$\sum_{d \in W} \sum_{(h,k):k \in L_d} f_{hk}^j = r_{jk} \quad \forall j \in N \setminus \{0\}, \forall k \in K \quad (6)$$

$$\sum_{j \in N \setminus \{0\}} t_j \sum_{(h,k):k \in L_d} f_{hk}^j \leq \sum_{t \in O_d: st \geq k} D_t \quad \forall d \in W, \forall k \in K \quad (7)$$

$$\sum_{j \in N \setminus \{0\}} f_{hi}^j \leq n q_{hi} \quad \forall (h,i) \in A_W \quad (8)$$

$$f_{hi}^j \in \{0, 1\} \quad \forall (h,i) \in A_W, \forall j \in N \setminus \{0\}$$

$$q_{hi} \in \{0, 1\} \quad \forall (h,i) \in A_W$$

For each patient j , the flow variables $\{f_{hi}^j\}$ model a directed path in G_W from node 1 to node n_w (constraints (5)). These variables model a feasible pattern for j , that is a pattern which is compatible with the care plan of j , thanks to constraints (6). Constraints (7) take into account, skill by skill, the operators availability in each day of the planning horizon. In fact, denoting by t_j the service time at patient j , (7) impose that the total service time of scheduled visits of skill k per day does not exceed the daily availability of the operators of skill at least k (here D_t denotes the workday length of operator t). Finally, constraints (8) link together the flow variables f_{hi}^j with the design variables q_{hi} . Such auxiliary variables $\{q_{hi}\}$ are introduced to discover which arcs are used to design the patterns: by minimizing the total number of used arcs, the model thus tends to minimize, in an implicit way, the number of generated patterns.

In the experimental campaign, policy *FB* has been used in combination with a parameter that reduces the operators availability. In fact, since the flow based model neglects the traveling times, it may occur that the patterns

thus provided generate an infeasible solution when the routing issue is also considered. A reduction of the operators availability is then used as a means to prevent some undesirable infeasibilities. Specifically, three values of the aforesaid parameter are considered: 0.5 which halves the operators availability, 0.75 which reduces the availability by 25%, and 1 which maintains the real availabilities. The corresponding policies will be denoted by *FB-50*, *FB-75* and *FB-100*, respectively.

3 Computational results

We generated a set of Home Care instances starting from real data provided by one of the largest Italian public medical care unit. The considered provider operates in the north of Italy and its services cover a region that is organized in divisions, in turn organized in districts. The instances we used consider the largest district of the Merate area and comprise 10 municipalities. Two skills are considered for operators and patient requests: *ordinary*, corresponding to the lowest ability or skill 1, and *palliative*, corresponding to the highest ability or skill 2. In regards to the patients, we had access to the care profile of 4123 patients in the time period [2004 - 2008] and we selected two weeks, i.e. a week in January 2006 (hereafter denoted by *January 2006*) with 129 patients and a week in April 2007 (hereafter denoted by *April 2007*) with 163 patients. Patient demands had been computed starting from historical series. The district under consideration is characterized by 11 operators, 8 of which of skill 1 and 3 of skill 2. In all the generated instances, the traveling times have been computed via Google Maps for the inter-municipalities distances, while they have been set equal to 3 minutes for the intra-municipalities distances, consistently with the provider indications. Furthermore, according to the medical care unit indications the service time has been fixed to 30 minutes.

In regards to the patterns, which are a peculiarity of our approach, we used the three generation policies described in Section 2, by considering a weekly time horizon. The three policies may produce a number of patterns very different the one from each other; these values are reported in Table 1.

Summarizing, the experimental campaign analyzes the impact of the pattern generation policies (5 choices) in the selected weeks, both in terms of efficiency of the optimization process and quality of the solutions provided. The experiments related to the five policies have been performed on a AMD Opteron(tm) Dual Core Processor 246 (CPU MHz 1991.060). The solver is CPLEX 12.4 with a time limit of 12 hours and a memory limit for the branch and bound tree of 1 GB. In the following the computational times are ex-

Table 1
Number of patterns used

Week	Heur	ImplSol	FB-50	FB-75	FB-100
January 2006	20	29	13	11	11
April 2007	27	33	17	14	14

Table 2
LP results

		Heur	ImplSol	FB-50	FB-75	FB-100
January 2006	LPTime	9044.38	17805.47	1337.04	219.17	266.21
	LPValue	0.3552	0.3552	0.3552	0.2922	0.2922
April 2007	LPTime	17450.49	17689.80	1942.01	1702.29	1785.68
	LPValue	0.5393	0.5393	0.5393	0.5393	inf

pressed in seconds of CPU time.

Partial (and preliminary) computational results are reported below. Precisely, Table 2 reports the performance of the models in terms of time required to solve the Linear Programming relaxation (LPTime) and in terms of LP objective function (LPValue), with string “inf” denoting infeasibility. It is worth observing that the LPTime required for policies *Heur* and *ImplSol*, i.e. the policy implemented by the provider, can be quite high. On the contrary the LPTime is much more shorter for the flow based policies which are characterized by a much smaller number of generated patterns with respect to the other policies (see Table 1). Determining a good and limited set of patterns seems thus to be crucial. Particular attention should be given to the LPValue: observe that the LPValue is the same (when available) for policies *Heur*, *ImplSol* and *FB-50*, thus suggesting that the selected patterns are sufficient to obtain the same solution quality in terms of (relaxed) minimum operator utilization factor (but, as observed, with very different computational times).

In Table 3 the results obtained when the integer problem is solved are reported for all the pattern selection policies in terms of percentage relative gap computed with respect to the best upper bound obtained in the branch and bound tree (string “n.a.” is used to point out that no integer solution is found). The stopping criterion that determines the algorithm termination is given in columns Stop, where T is used to indicate that the time limit has been exceeded while M is used to indicate an out of memory condition.

It is possible to observe that the *FB-50* policy allows one to determine better solutions, in terms of percentage gaps, than the ones computed by

Table 3
IP results

	Heur		ImplSol		FB-50		FB-75		FB-100	
	%Gap	Stop	%Gap	Stop	%Gap	Stop	%Gap	Stop	%Gap	Stop
January 2006	1.02	T	0.66	T	0.07	T	n.a.	T	n.a.	T
April 2007	0.48	T	0.41	T	0.17	M	0.17	T	n.a.	inf

using policies *Heur* and *ImplSol* (recall, in fact, that the LPValue is the same, see Table 2), although the problem solution required large computational time and memory resources. However a large but affordable consumption of such resources does not seem to be an issue when the focus is a difficult planning problem that has to be solved once a week, or for a still longer time horizon.

More complete computational results, which comprise the analysis of an alternative objective function, and the assessment of the quality of the returned solutions in terms of operator utilization factor and travelled time, will be discussed.

References

- [1] Begur, S. V., D. M. Miller, and J. R. Weaver, *An Integrated Spatial DSS for Scheduling and Routing Home-Health-Care Nurses*, Interfaces **27**(4) (1997), 35–48.
- [2] Cappanera, P., L. Gouveia, and M. G. Scutellà, *Models and valid inequalities to Asymmetric Skill-Based Routing Problems*, EURO Journal on Transportation and Logistics, doi: 10.1007/s13676-012-0012-y (2012).
- [3] Cappanera, P., and M. G. Scutellà, *Joint assignment, scheduling and routing models to Home Care optimization: a pattern based approach*, Technical Report TR-13-05, Dipartimento di Informatica, Università di Pisa, 2013.
- [4] Cheng E., and J. L. Rich, *A Home Health Care Routing and Scheduling Problem*, Technical Report, 1998.
- [5] Jensen, T. S., “Application of Metaheuristics to Real-life Scheduling Problems“, Ph.D. thesis, Department of Mathematics and Computer Science, University of Southern Denmark, 2012.
- [6] Nickel, S., M. Schroder, and J. Steeg, *Planning for Home Health Care Services*, Technical Report, Berichte des Fraunhofer ITWM, 173, 2009.

Multipolar routing: where dynamic and static routing meet

Walid Ben-Ameur¹

*Samovar, CNRS UMR 5157
Télécom SudParis
Evry, France*

Mateusz Żotkiewicz²

*Institute of Telecommunications
Warsaw University of Technology
Warszawa, Poland*

Abstract

Assuming that the traffic matrix belongs to a polytope, we describe a new routing paradigm where each traffic matrix is routed using a combination of a number of extreme routings. This combination depends on the current traffic matrix. Multipolar routing can be seen as a generalization of both dynamic routing and robust static routing. Moreover, the time complexity of multipolar routing is under control since it depends on the number of poles (i.e., the number of extreme routings) which can be defined by the network planner.

Keywords: Robust optimization, network optimization, routing, polyhedral model.

¹ Email: walid.benameur@telecom-sudparis.eu

² Email: mzotkiew@tele.pw.edu.pl

1 Introduction

The traffic matrix in modern communication networks is uncertain. There are mainly two sources of traffic uncertainty. The first one is that traffic is variable. This variability occurs at different time scales. It is known that traffic varies in a diurnal way. Some periodicity is also observed over the weeks. This variability is now accentuated by the development of many new telecommunication services and the mobility of users. Traffic may also vary if there are network failures or even modifications in routing between neighboring networks. The second source of uncertainty is in fact the lack of information. Even if we assume that the traffic matrix is constant over some period of time, it is generally not possible to measure the traffic between each pair of origin-destination in the network. In fact, collecting data related to each pair of nodes can be prohibitively expensive.

Routing strategies should be adapted to take into account traffic uncertainty. This paper is proposing some new strategies where a certain delay function is minimized for uncertain traffic. In order to handle the traffic uncertainty, we will employ the polyhedral model introduced in [5, 7, 8]. In this model, all possible scenarios belong to a traffic demand polytope D . The polyhedral model can be seen as a generalization of the model where a limited number of traffic matrices is considered (see, e.g., [3, 15]). It also generalizes the Hose model introduced by [12] where an upper bound is specified for each node to limit the total traffic originating at the node. Bounds related to the traffic going to a given node can also be considered. Another special case of the polyhedral model is proposed in [10]. In this model, a volume of each demand varies in the range given by a mean value and a known deviation. The uncertainty set is such that the number of demands (commodities) deviating simultaneously from their mean value is bounded by a given constant.

In the cited papers [5, 7, 8] where the polyhedral model is considered, authors were working with the so called *Static Routing*, i.e., they assumed that the routing schemes are provided before a network is made operational, and a task is to provide such routing schemes that are compatible with all traffic matrices of D but not directly depending on the current traffic matrix. In other terms, *Static Routing* assumes that traffic matrices are always routed in the same way regardless of their locations in the traffic demand polytope.

Static Routing can be considered conservative in terms of cost. Therefore, in [6] some improvements were proposed. The main idea is to divide the uncertainty set D into some subsets, and use *Static Routing* for each of them independently. This class of problems was subsequently solved in [9, 18] where

different variants of the partitioning are studied. This strategy may be called *Multistatic Routing*. The idea behind partitioning the polytope D is to get closer to the optimal *Dynamic Routing* strategy where routing depends on a current traffic matrix. Unfortunately, *Dynamic Routing* is difficult to implement in networks and is also generally difficult to compute as shown by [11] and [14] answering a question raised in [5, 8]. The traffic polytope can also be partitioned according to time assuming that time is one of the dimensions of the traffic polytope. This makes the connection with work on multi-period routing in the context of switched networks, e.g., [3].

A distributed technique called *Volume-Oriented Routing* where each traffic demand is routed independently of the other demands is proposed in [19]. For each demand, the traffic below a certain threshold is routed in a certain way, and the rest of the traffic is routed according to a different scheme. The thresholds and the routing schemes should be of course computed by the network planner to optimize some global performance criteria.

Another strategy, denoted by *Affine Routing*, is proposed in [16]. The affine routing assumption consists in imposing that the resources reserved to route a traffic matrix is an affine combination of the traffic components [17].

In this paper we present a new routing paradigm, called *Multipolar Routing*, that merges the simplicity of *Static Routing* with the efficiency of *Dynamic Routing*. Moreover, it does not involve abrupt changes in network flows. In fact, *Multipolar Routing* can be seen as a generalization of both *Static Routing* and *Dynamic Routing*, as by increasing the time complexity of *Multipolar Routing*, we can obtain an optimal *Dynamic Routing*.

The paper is organized as follows. In Section 2 we present a notation and formulate *Static Routing*, which is a starting point for our research. *Dynamic Routing* is quickly recalled in Section 3. In Section 4 *Multipolar Routing* is introduced and discussed. Due to page limitations, numerical experiments could not be included in the paper.

2 Static routing

In this section, the notation and the basic *Static Routing* problem are presented. We consider a directed graph $G = (\mathcal{V}, \mathcal{A})$, where \mathcal{V} is a set of nodes and \mathcal{A} is a set of arcs. The graph represents a backbone and the arcs depict unidirectional transmission links. For each arc $a \in \mathcal{A}$ installed capacity is denoted by $c_a \in \mathbb{R}_+$. Let $t = (t_{ij})_{i,j \in \mathcal{V}}$ be a vector of $\mathbb{R}^{|\mathcal{V}|^2}$ that specifies values of traffic demands (or capacity requirements) between pairs of nodes of \mathcal{V} . This vector will be called a *traffic matrix*. A demand between nodes u and

v is called “demand uv ”, and its value is denoted by t_{uv} . The traffic matrix is supposed to be variable, and can be any point of a *traffic demand polytope* D . D is generally defined by some linear constraints involving the variables t_{ij} , for $i, j \in \mathcal{V}$. It can also be defined as a convex hull of a finite number of extreme points. More generally, D can be given by an oracle.

Let $\mathcal{P}(i, j)$ be a set of acyclic paths of G from i to j ($i, j \in \mathcal{V}$). Let $x_p^{(i,j)}$ be a proportion of the traffic demand from i to j ($i, j \in \mathcal{V}$) carried through a path $p \in \mathcal{P}(i, j)$. Note that $0 \leq x_p^{(i,j)} \leq 1$. For a current traffic matrix $t \in D$, the traffic carried through p is then given by $t_{ij}x_p^{(i,j)}$. We also use $y_a^{(i,j)}$ to denote a proportion of the traffic from i to j ($i, j \in \mathcal{V}$) flowing through an arc $a \in \mathcal{A}$. Let f_a be the maximum amount of traffic carried on an arc $a \in \mathcal{A}$. It can also be considered as the minimum capacity that has to be reserved on an arc a to satisfy all the constraints. The cost function to be minimized is denoted by w^{ST} . We assume that $w^{ST} = \sum_{a \in \mathcal{A}} h(f_a, c_a)$ where h is a convex piecewise linear function.

The problem of computing the minimum-cost *Static Routing* of an uncertainty domain D is given by (1).

$$\text{Minimize: } w^{ST} = \sum_{a \in \mathcal{A}} h(f_a, c_a) \\ \sum_{p \in \mathcal{P}(i,j)} x_p^{(i,j)} \geq 1 \quad \forall i, j \in \mathcal{V} \quad (1a)$$

$$\sum_{p \in \mathcal{P}(i,j): p \ni a} x_p^{(i,j)} \leq y_a^{(i,j)} \quad \forall i, j \in \mathcal{V}, \forall a \in \mathcal{A} \quad (1b)$$

$$\sum_{i,j \in \mathcal{V}} y_a^{(i,j)} t_{ij} \leq f_a \quad \forall a \in \mathcal{A}, \forall t \in D \quad (1c)$$

$$f_a \leq c_a \quad \forall a \in \mathcal{A} \quad (1d)$$

$$x_p^{(i,j)} \geq 0 \quad \forall p \in \mathcal{P}(i,j), \forall i, j \in \mathcal{V} \quad (1e)$$

$$y_a^{(i,j)} \geq 0 \quad \forall a \in \mathcal{A}, \forall i, j \in \mathcal{V} \quad (1f)$$

Observe that (1c) must be valid for each traffic matrix t in the polytope D .

Problem (1) can be easily solved using an algorithm based on constraint and path generation (see [7, 8]). Another approach based on duality is proposed in [1]. Notice that *Static Routing* was also considered in [2, 4] where it was called oblivious routing.

3 Dynamic routing

In this situation, the problem consists in providing routing schemes (possibly different) for all traffic matrices t from a traffic demand polytope. More specifically, variables $x_p^{(i,j)}$ and $y_a^{(i,j)}$ depend on t and can be denoted by $x_p^{(i,j)}(t)$ and $y_a^{(i,j)}(t)$. Since the problem formulation is very close to (1), i.e., we should only replace $x_p^{(i,j)}$ (resp. $y_a^{(i,j)}$) by $x_p^{(i,j)}(t)$ (resp. $y_a^{(i,j)}(t)$), we do not give it explicitly.

The problem of *Dynamic Routing* is obviously easy to solve when D is defined as a convex hull of a given number of traffic matrices. It can also be solved in polynomial time when D is defined by its facets and the number of demands is less than a constant [9]. Some other special cases where dynamic routing is easy to compute are presented in [13, 14].

Unfortunately, *Dynamic Routing* is generally difficult to compute as shown by [11] and [14]. In other words, *Dynamic Routing* is not only difficult to implement but also difficult to compute.

4 Multipolar routing

In this section we present *Multipolar Routing*. It is a new routing paradigm that merges simplicity (polynomially solvable) of *Static Routing* with efficiency of *Dynamic Routing*. In *Multipolar Routing* each meaningful (not dominated) traffic demand matrix t from a traffic demand polytope D can be represented as a convex combination of poles (special traffic demand matrices) from a given set. Each pole has a routing scheme associated with it, and each $t \in D$ is routed using a combination of those routing schemes.

4.1 Notation and formulation

In order to formulate *Multipolar Routing* we introduce additional sets of constants. Let \mathcal{K} be a set of poles. Each pole $k \in \mathcal{K}$ is a traffic matrix and can be seen as a vector. k_{ij} denotes the traffic volume between nodes i and j for a pole k .

The poles are given beforehand. However, they have to satisfy the constraints that each traffic demand matrix $t \in D$ has to belong to the convex hull of the poles and the zero matrix. In order to formally state this condition let us introduce the following set of variables. λ_t^k represents the importance of a pole k from a point of view of a traffic demand matrix t . Poles should be

chosen in such a way that the following conditions are satisfied.

$$\sum_{k \in \mathcal{K}} \lambda_t^k k_{ij} = t_{ij}; \quad \sum_{k \in \mathcal{K}} \lambda_t^k \leq 1; \quad \lambda_t^k \geq 0; \quad \forall t \in D, \forall k \in \mathcal{K}, \forall i, j \in \mathcal{V} \quad (2)$$

For each $t \in D$ the system (2) has in general an infinite number of solutions λ_t . We will denote the set of those solutions by Λ_t .

In *Multipolar Routing* each pole has a routing associated with it. Therefore, the following sets of variables have to be introduced. Let $x_p^{(i,j)k}$ be the proportion of a traffic demand from i to j ($i, j \in \mathcal{V}$) carried through a path $p \in \mathcal{P}(i, j)$ for a pole $k \in \mathcal{K}$. $y_a^{(i,j)k}$ denotes the proportion of the traffic demand from i to j ($i, j \in \mathcal{V}$) flowing through an arc $a \in \mathcal{A}$ for a pole $k \in \mathcal{K}$.

Having the variables defined, it is possible to formulate *Multipolar Routing* as follows, where the cost function is denoted by w^{MPR} .

$$\begin{aligned} \text{Minimize: } w^{MPR} &= \sum_{a \in \mathcal{A}} h(f_a, c_a) \\ \sum_{p \in \mathcal{P}(i,j)} x_p^{(i,j)k} &\geq 1 \quad \forall i, j \in \mathcal{V}, \forall k \in \mathcal{K} \end{aligned} \quad (3a)$$

$$\sum_{p \in \mathcal{P}(i,j): p \ni a} x_p^{(i,j)k} \leq y_a^{(i,j)k} \quad \forall i, j \in \mathcal{V}, \forall a \in \mathcal{A}, \forall k \in \mathcal{K} \quad (3b)$$

$$\sum_{i,j \in \mathcal{V}} \sum_{k \in \mathcal{K}} y_a^{(i,j)k} \lambda_t^k k_{ij} \leq f_a \quad \forall a \in \mathcal{A}, \forall t \in D, \lambda_t \in \Lambda_t \quad (3c)$$

$$f_a \leq c_a \quad \forall a \in \mathcal{A} \quad (3d)$$

$$x_p^{(i,j)k} \geq 0 \quad \forall p \in \mathcal{P}(i, j), \forall i, j \in \mathcal{V}, \forall k \in \mathcal{K} \quad (3e)$$

$$y_a^{(i,j)k} \geq 0 \quad \forall a \in \mathcal{A}, \forall i, j \in \mathcal{V}, \forall k \in \mathcal{K} \quad (3f)$$

In the formulation $|\mathcal{K}|$ independent sets of routing variables are used (one for each pole). Therefore, the size of the problem highly depends on the number of poles. For each traffic demand matrix $t \in D$, the lambda values represent possible convex combinations of poles that form t . The lambda values also state how t should be routed (3c), i.e., how the routing should be divided between $|\mathcal{K}|$ independent routings associated with the poles.

When the number of poles is polynomial in the size of data, then *Multipolar Routing* problem can be solved in polynomial time using techniques for *Static Routing* mentioned in Section 2, i.e., using an algorithm based on constraint generation (see [7,8]), and generating paths in an iterative way. More precisely, at each iteration of the algorithm we have a routing associated to each pole and we want to check for each arc a whether there is a matrix $t \in D$ violating

constraint (3c). This can be done by solving a simple linear program where we aim to maximize $\sum_{i,j \in \mathcal{V}} \sum_{k \in \mathcal{K}} y_a^{(i,j)k} \lambda_t^k k_{ij}$ under the constraints $t \in D$, $\sum_{k \in \mathcal{K}} \lambda_t^k k_{ij} = t_{ij} \forall i, j \in \mathcal{V}$, $\sum_{k \in \mathcal{K}} \lambda_t^k \leq 1$ and non-negativity constraints $\lambda_t^k \geq 0 \forall k \in \mathcal{K}$. Notice that this immediately implies that problem (3) can be solved in polynomial time.

4.2 Discussion

Multipolar Routing is very general and encompasses a number of interesting special cases. We will briefly discuss some of them in this section.

First, *Multipolar Routing* encompasses *Dynamic Routing* as a special case. It happens when the set of poles is equivalent to the set of extreme points of D . Note however that the number of extreme points of a polytope can be very large. That is why the implementation of dynamic routing is almost impossible in a general case.

We want to emphasize that *Multipolar Routing* is generally different to *Dynamic Routing*. We should understand that while there is a routing associated with each pole, we do not require each pole to be routable. Only matrices that are inside the polytope must be routed. Poles are used only to allow for combinations of different routing strategies. In other words, *Multipolar Routing* does not consist in defining a polytope including D and routing the matrices belonging to this polytope. We only route matrices inside D but we gain more flexibility by using the multipolar concept.

The number of poles should be at least equal to the dimension of the polytope since conditions (2) require that D is inside the convex hull of the pole and the zero matrix. The number of poles can even be larger making the implementation of *Multipolar Routing* difficult in practice. However, even if the number of poles is high, one can require that some of these poles have the same routing. We can for example require that the routings corresponding to all poles are exactly the same. In this case, we simply get *Static Routing*. The set of poles can be also partitioned into a small number of subsets with the same routing for poles belonging to the same subset.

Let us see here another variant of *Multipolar Routing* leading again to *Static Routing*. One natural way to define poles consists in considering for each traffic component (i.e., each $i \in \mathcal{V}$, and $j \in \mathcal{V} \setminus \{i\}$) a pole k defined by $k = (0, \dots, 0, k_{ij}, 0, \dots, 0)^T$ where only this traffic component is different to 0. The numbers k_{ij} should be of course large enough to allow the satisfaction of conditions (2). Since for each one of these poles there is only one nonzero traffic component, the routing associated to each pole is equivalent to routing only

one demand. Then we can build a *Static Routing* just by routing each traffic demand similarly to how it is routed when we consider the pole associated to this traffic demand (traffic component).

References

- [1] Altin, A., E. Amaldi, P. Belotti, and M.C. Pinar, Provisioning virtual private networks under traffic uncertainty. *Networks*, 49(1), 100-115, 2007.
- [2] Applegate, D., and E. Cohen, Making intra-domain routing robust to changing and uncertain traffic demands: understanding fundamental tradeoffs. *Proceeding of ACM SIGCOMM*, 2003.
- [3] Ash, G., *Dynamic routing in telecommunications networks*, McGraw-Hill, 1997.
- [4] Azar, Y., E. Cohen, A. Fiat, H. Kaplan, and H. Rache, Optimal oblivious routing in polynomial time. *Proceedings of the 35th ACM Symposium on the Theory of Computing*, 2003.
- [5] Ben-Ameur,W., and H. Kerivin, Routing of uncertain demands. *Proceedings of Informs*, 2001.
- [6] Ben-Ameur, W., Between fully dynamic routing and robust stable routing. *Proceedings of DRCN*, 2007.
- [7] Ben-Ameur, W., and H. Kerivin, New economical virtual private networks. *Communication of the ACM*, 46(6), 69-73, 2003.
- [8] Ben-Ameur, W., and H. Kerivin, Routing of uncertain traffic demands. *Optimization and Engineering*, 6(3), 283-313, 2005.
- [9] Ben-Ameur, W., and M. Żotkiewicz, Robust routing and optimal partitioning of a traffic demand polytope. *International Transactions in Operational Research*, 18(3), 307-333, 2011.
- [10] Bertsimas, D., and M. Sim, Robust discrete optimization and network flows. *Mathematical Programming*, 98, 49-71, 2003.
- [11] Chekuri, C., , G. Oriolo, M.G. Scutellà, and F.B. Shepherd, Hardness of robust network design. *Networks*, 50(1), 50-54, 2007.
- [12] Fingerhut, A.J., S. Suri, and J. Turner, Designing least-cost nonblocking broadband networks. *Journal of Algorithms*, 24(2), 287-309, 1997.
- [13] Frangioni, A., F. Pascali, M.G. Scutellà, Static and dynamic routing under disjoint dominant extreme demands. *Operations Research Letters*, 39(1), 36-39, 2011.
- [14] Minoux, M., Robust network optimization under polyhedral demand uncertainty is NP-hard. *Discrete Applied Mathematics*, 158, 597-603, 2010.
- [15] Ouorou, A., Robust capacity assignment in telecommunications. *Computational Management Science*, 3(4), 285-305, 2006.
- [16] Ouorou, A., and J-Ph. Vial, A model for robust capacity planning for telecommunications under demand uncertainty. *Proceedings of DRCN*, 2007.
- [17] Poss, M., and C. Raack, Affine recourse for the robust network design problem: Between static and dynamic routing. *Networks*, published online Doi 10.1002/net.21482.
- [18] Scutellà, M.G., On improving optimal oblivious routing. *Operations Research Letters*, 37(3), 197-200, 2009.
- [19] Żotkiewicz, M., and W. Ben-Ameur, Volume-oriented routing and its modifications. *Telecommunication Systems*, published online Doi 10.1007/s11235-011-9602-5.

Extended Cutset Inequalities for the Network Power Consumption Problem

Arie Koster¹, Truong Khoa Phan², Martin Tieves¹

¹*Lehrstuhl II für Matehmatis
RWTH Aachen
Aachen, Germany*

²*Joint Project Mascotte
I3S(CNRS-UNS), INRIA
Sophia Antipolis, France*

Abstract

In this paper, we enhance the MIP formulation for the Network Power Consumption problem, proposed by Giroire et al. We derive cutting planes, extending the well-known cutset inequalities, and report on preliminary computations.

Keywords: Green Networking, Mixed Integer Programming, Cutting Planes

1 Introduction

Green networking and energy-efficient routing are major aspects of today's backbone networks. Recent studies [8] indicate that the power consumption of the network mostly depends on the number of active links and routers. Based on this idea, energy-efficient routing [9] has been proposed so that traffic flows are aggregated into fewer links while preserving connectivity and QoS. Thus energy consumption can be reduced. Recently, Anand et al. [10] have analyzed data redundancy elimination opportunities between certain routers. Extending on this, Cisco, Juniper, and others have proposed the so called WAN Optimization Controllers (WOC). These can be enabled at each router

¹ Email:{koster,tieves}@math2.rwth-aachen.de

² Email:truong_khoa.phan@inria.fr

allowing for caching/compressing of traffic flows and thus potentially reducing traffic amounts [11]. Such a compressed flow can not be compressed again and it must be uncompressed somewhere on the network so that the receiver receives uncompressed flow. Based on this idea, Giroire et al. have proposed the GreenRE model [1], in which the necessary amount of active links can be reduced further via compressions, compared to standard energy aware routing (minimal #link usage), and thus, more energy can be saved.

In this paper, we will extend the work in [1] from a methodical point of view. As shown in [1], heuristic algorithms are used commonly to find approximate solutions since it is time-consuming to find optimal solutions. We will focus on enhancements of the ILP formulation to benchmark these heuristic solutions. Therefore, we propose two classes of valid inequalities and rate their potentials with an exact separation algorithm.

2 The Network Power Consumption Problem

Let $G = (V, E)$ be an undirected graph. The nodes V describe routers and the edges E describe potential immediate connections between those routers. The (energy) costs of an active WOC at node v is given by $PN_v \geq 0$ and the costs of using edge $e = \{v, w\}$ are given by $PE_e \geq 0$. We denote by D^{vw} the demand of (uncompressed) traffic to be routed between v and $w \in V$, $v \neq w$. Further, we assume that the capacity of all edges is a constant $c \in \mathbb{R}_{>0}$ and that the compression rate is given by $\gamma \in \mathbb{R}$, $\gamma > 1$. The task is to find a feasible routing between all participants, fulfilling all demands and being minimal in energy consumption.

We formulate the above presented problem as (mixed) integer linear program. Hereby, we use variables $x_{uv} \in \{0, 1\}$, indicating the usage of link $\{u, v\} \in E$ and variables $w_u \in \{0, 1\}$ denoting the activation of a WOC in $u \in V$. Further, we use variables $f_{uv}^{st}, g_{uv}^{st} \geq 0$ for describing the (un-) compressed flows for demand $s \neq t$ on edge uv from u to v of a feasible routing. Hereby f describes the amount of uncompressed flow on edge uv of demand st whereas g describes the amount of compressed flow respectively. Then, denoting by $N(v)$ all neighbors of a node v , the problem can be stated as

$$\begin{aligned} \min \quad & \sum_{(uv) \in E} PE_{uv} x_{uv} + \sum_{u \in V} PN_u w_u \\ \text{s.t. } & \sum_{v \in N(u)} (f_{vu}^{st} + \gamma g_{vu}^{st} - f_{uv}^{st} - \gamma g_{uv}^{st}) = \begin{cases} -D^{st} & \text{if } u = s, \\ D^{st} & \text{if } u = t, \\ 0 & \text{else} \end{cases} \forall u \in V, \forall s \neq t \in V \times V \end{aligned} \quad (1)$$

$$\sum_{s \in V} \sum_{\substack{t \in V \\ t \neq s}} (f_{uv}^{st} + f_{vu}^{st} + g_{uv}^{st} + g_{vu}^{st}) \leq cx_{uv} \quad \forall uv \in E \quad (2)$$

$$\sum_{v \in N(u)} (g_{uv}^{st} - g_{vu}^{st}) \leq \frac{D^{st}}{\gamma} w_u \quad \begin{aligned} &\forall u \in V, \\ &\forall s \neq t \in V \times V \end{aligned} \quad (3)$$

$$\sum_{v \in N(u)} (g_{vu}^{st} - g_{uv}^{st}) \leq \frac{D^{st}}{\gamma} w_u \quad \begin{aligned} &\forall u \in V, \\ &\forall s \neq t \in V \times V \end{aligned} \quad (4)$$

$$x_{uv} \in \{0, 1\}, w_u \in \{0, 1\}, f_{uv}^{st} \geq 0, g_{uv}^{st} \geq 0$$

The first class of constraints (1) consists of flow conservation constraints (either compressed or not). The second class (2) describes link capacity constraints, and the remaining class (3),(4) constitutes the possibility to (de-) compress flow at a node if a WOC is enabled. In the following, we will refer to this MIP as the NPC (Network Power Consumption) formulation.

3 Valid Inequalities

We present the following inequalities for strengthening the NPC formulation. Hereby we align ourselves closely to the notation of Raack [4]. Given a set $S \subset V$, the total demand, which needs to be routed between S and $V \setminus S =: \bar{S}$ is denoted by

$$D^S := \sum_{v \in S} \sum_{w \in \bar{S}} D^{vw} + \sum_{v \in \bar{S}} \sum_{w \in S} D^{vw}.$$

Further, let $\delta(S, \bar{S})$ be the corresponding cut between both sets. Now, the well known cutset inequality for network design [2], [3] can be adapted.

Theorem 3.1 *Let $S, \bar{S} \subset V$ be a partition of V . Then the **cutset** inequality*

$$\sum_{uv \in \delta(S, \bar{S})} x_{uv} \geq \left\lceil \frac{D^S}{c\gamma} \right\rceil. \quad (5)$$

holds for NPC.

This inequality assumes that at least one WOC is available in each of the two subsets. Assuming the contrary, we could increase the right-hand side. The following result takes the actual number of WOCs into account.

Theorem 3.2 Let $S, \bar{S} \subset V$ be a partition of V . Then the **extended cutset inequality**

$$\left(\left\lceil \frac{D^S}{c} \right\rceil - \left\lceil \frac{D^S}{c\gamma} \right\rceil \right) \sum_{u \in S} w_u + \sum_{uv \in \delta(S, \bar{S})} x_{uv} \geq \left\lceil \frac{D^S}{c} \right\rceil \quad (6)$$

holds for NPC.

Proof. Let S be given. Since w_u is integer, we distinguish two cases:

Case 1: Let $w_u = 0$ for all $u \in S$. Then (6) becomes

$$\sum_{uv \in \delta(S, \bar{S})} x_{uv} \geq \left\lceil \frac{D^S}{c} \right\rceil,$$

which is equivalent to the cutset inequality in absence of WOCs.

Case 2: Let $\sum_{u \in S} w_u \geq 1$. We obtain

$$\begin{aligned} & \left(\left\lceil \frac{D^S}{c} \right\rceil - \left\lceil \frac{D^S}{c\gamma} \right\rceil \right) \sum_{u \in S} w_u + \sum_{uv \in \delta(S, \bar{S})} x_{uv} \\ & \geq \left(\left\lceil \frac{D^S}{c} \right\rceil - \left\lceil \frac{D^S}{c\gamma} \right\rceil \right) + \sum_{uv \in \delta(S, \bar{S})} x_{uv} \geq \left\lceil \frac{D^S}{c} \right\rceil \end{aligned}$$

which holds, because of the cutset inequality (5). \square

Comparing inequality (5) with (6), we conclude: both inequalities are equal if exactly one WOC is deployed (in S). If no WOC is deployed, the latter one strictly dominates the first and vice versa if more than one WOC is available. In fractional solutions, no dominance relation can be given: the latter inequality has a weaker left-hand side while the first inequality has a weaker right-hand side. If S contains WOCs but \bar{S} does not, an exchange of S and \bar{S} yields the stronger inequality.

4 Recognizing violated Inequalities

Employing all inequalities for all possible cuts in the NPC formulation is not a realistic option. In the following, we will present a straightforward approach for separating these inequalities via an integer linear program generalizing the separation of cutset inequalities in [4]. While the objective function ‘rebuilds’ the extended cut inequality (6) for the current LP solution (w^*, x^*) , we need the following variables:

For each $v \neq w \in V$, let $z_{vw} \in \{0, 1\}$ denote, whether v and w are in separate sides of the cut. Further, let $d, d_\gamma \in \mathbb{Z}_{\geq 0}$, which represent the values $\lceil \frac{D^S}{c} \rceil$ and $\lceil \frac{D^S}{c\gamma} \rceil$, respectively. For every node $v \in V$, $\alpha_v \in \{0, 1\}$ denotes whether v is in S or in \bar{S} . Finally, given a sufficiently large constant $M \in \mathbb{N}$, for $k = 1, \dots, M$, and $v \in V$, let $\alpha_v^k \in \{0, 1\}$ denote if k equals $\lceil \frac{D^S}{c} \rceil - \lceil \frac{D^S}{c\gamma} \rceil$ and v is in S , i.e., α_v^k is an enumeration of all possible coefficients of the WOC-variable coefficients in the extended cut. Consequentially, we have that

$$\sum_{k=1}^M k \alpha_v^k w_v^* = \left(\left\lceil \frac{D^S}{c} \right\rceil - \left\lceil \frac{D^S}{c\gamma} \right\rceil \right) \sum_{v \in V} \alpha_v w_v^* = \left(\left\lceil \frac{D^S}{c} \right\rceil - \left\lceil \frac{D^S}{c\gamma} \right\rceil \right) \sum_{v \in S} w_v^*.$$

Then the problem of finding a violated extended cut can be written as an extension of the triangle-formulation of the cut-polytope [12] as

$$\begin{aligned} \min & \sum_{k=1}^M k w_v^* \alpha_v^k + \sum_{vw \in E} x_{vw}^* z_{vw} - d \\ \text{s.t. } & -1 + \epsilon \leq \frac{1}{c} \sum_{v \in V} \sum_{\substack{w \in V \\ w \neq v}} D^{vw} z_{vw} - d \leq 0 \end{aligned} \quad (7)$$

$$-1 + \epsilon \leq \frac{1}{c\gamma} \sum_{v \in V} \sum_{\substack{w \in V \\ w \neq v}} D^{vw} z_{vw} - d_\gamma \leq 0 \quad (8)$$

$$z_{vw} + z_{uw} + z_{uv} \leq 2 \quad \forall \{u, v, w\} \subset V \quad (9)$$

$$z_{vw} + z_{uw} \geq z_{uv} \quad \forall \{u, v, w\} \subset V \quad (10)$$

$$\alpha_v + \alpha_w \leq 2 - z_{vw} \quad \forall v, w \in V, v \neq w \quad (11)$$

$$\alpha_v + \alpha_w \geq z_{vw} \quad \forall v, w \in V, v \neq w \quad (12)$$

$$\sum_{k=1}^M \alpha_v^k = \alpha_v \quad \forall v \in V \quad (13)$$

$$M(1 - \alpha_v) + \sum_{k=1}^M k \alpha_v^k \geq d - d_\gamma \quad \forall v \in V \quad (14)$$

$$z_{vw}, \alpha_v, \alpha_v^k \in \{0, 1\}, d, d_\gamma \in \mathbb{Z}_{\geq 0}$$

In this model, the inequalities (9), (10) establish a feasible cut and the inequalities (7) and (8) recognize the rounded traffic across this cut. (11) and (12) determine which nodes are within the one side (S) of the cut, and the inequalities (13) - (14) choose the correct α_v^k values for each α_v .

If the resulting objective value is strictly smaller than zero, the optimal solution of the corresponding MIP describes a partition of V , violating an extended cutset inequality (the z_{vw} correspond to the edge variables and the α_v to the WOC variables in S). If the contrary holds, none of these extended cuts is violated.

This integer program can be adapted to separate cutset inequalities (5) easily by omitting the unnecessary parts (d , α_v , α_v^k) and restricting to the constraints (9), (10) and (8).

5 Preliminary Computational Experiments

In this section, we want to show benefits of incorporating the inequalities (5) and (6) within the (standard) MIP-solution process. In this preliminary study, for every LP solution, the inequalities (5) are separated. Only if no violated cut is found, the inequalities (6) are separated. We used modified instances of the SNDlib [7]. Taking care of the great variety within that library, all demands have been scaled such that a routing (without compression) is feasible at a capacity of 10,000 per link and infeasible at a capacity of 9,000. In this study, we used the instances ABILENE (scaling factor: 102), ATLANTA (2.6), DFN-BWIN (4.5), FRANCE (1.1) and POLSKA (0.17).

All computations have been done with CPLEX 12.4, with CPU-/thread-usage limited to one. For obtaining clear results, CPLEX internal cutting-planes have been disabled. We report on progress after the root node and compare ourselves to the usage of plain CPLEX (with the same settings). All scenarios have been tested with capacities 5,000 (halved), 10,000 (normal) and 20,000 (doubled) and a compression factor of $\gamma = 2$. The price for an edge has been determined as 200 [9] and the price of an WOC as 30 [1]. Success of the separation routine is measured by the relatively closed gap at the root, i.e. let DB denote the LP relaxation, DB_s the best dual bound obtained by our separation approach and PB the best primal bound available. Then we define the gap closed as $GC := \frac{DB_s - DB}{PB - DB}$. Clearly, an improvement is given, as soon as this value is positive and a higher value corresponds to a bigger improvement.

The results presented in Figure 1 are throughout positive. The gain from separating one or both of the two classes of inequalities is shown. The total improvement amounts to an average gap closed of 46.1%. This improvement was achieved by an average amount of 45 cuts per instance, 33 cutset inequalities (5) and 12 extended cuts (6). While the relation between the amounts of both cutting plane families is clear by the lazy separation approach of the extended

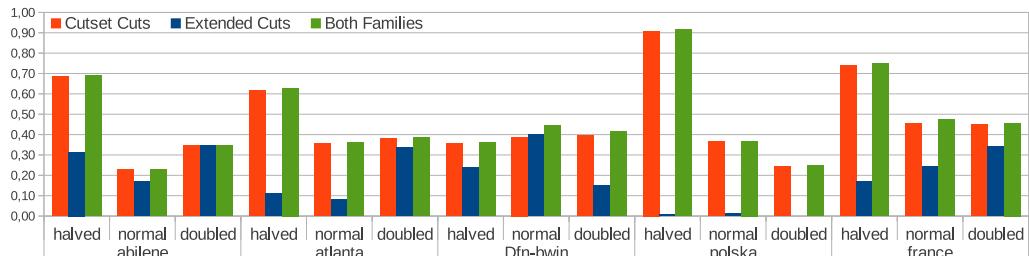


Fig. 1. Relative Gap Closing

cuts, it appears that the extended cuts (6) can still improve on the cutset inequalities (5). However, the success and the relation between both classes is highly dependant on the underlying network topology (and the edge/WOC prices). For example the POLSKA (doubled) instance seems to be very unfortunate for extended cuts, while the DFN-BWIN (normal) seems to favor exactly these. On the ABILENE (normal) instance, both behave equal.

The drawback of this separation approach is it's time consumption. The time needed for finding violated cuts and re-optimizing the linear relaxation is significant. Depending on the amount of cuts found, this procedure can amount to an increase of time consumption of more than 100%.

6 Conclusions

Concluding our paper, we rate our findings and point out potential further research. We derived two different classes of valid inequalities for the NPC-problem. These inequalities have been used in a straightforward cutting plane approach. Exemplary calculations showed that these inequalities are helpful in strengthening the formulation of the root node. However, the increase in time-consumption is a major drawback.

We believe that further research should be focused on a more sophisticated (but probably heuristic) separating algorithm. Clearly, this has to be joint with a more detailed analysis of success and effect of these inequalities. Finally, we still believe that there is potential for further improvement in the form of additional classes of valid inequalities and we hope to extend our finding to the robust counterpart of our model in the future.

References

- [1] F. Giroire, J. Moulierac, and T. Phan, and F. Roudaut, *Minimization of Network Power Consumption with Redundancy Elimination*, International IFIP TC 6 Networking Conference, Vol. 7289, 2012, 247–258.
- [2] A. Atamtürk, *On capacitated network design cut-set polyhedra*, Math. Program., Vol. 92, 2002, 425–437.
- [3] T. Magnanti, P. Mirchandani and R. Vachani, *The convex hull of two core capacitated network design problems*, Math. Program., Vol. 60, 1993, 233–250
- [4] C. Raack, A. Koster, S. Orlowski, and R. Wessäly, *On Cut-Based Inequalities for Capacitated Network Design Polyhedra*, Networks 57, Vol. 2, 2011, 141–156.
- [5] J. Chabarek, J. Sommers, P. Barford, C. estan, D. Tsiang, and S. Wright, *Power Awareness in Network Design and Routing*, INFOCOM 2008. The 27th Conference on Computer Communications. IEEE, 2008 , 457–465.
- [6] P. Mahadevan, P. Sharma, and S. Banerjee, *A Power Benchmarking Framework for network Devices*, NETWORKING 2009, Proceedings of the 8th International IFIP-TC 6 Networking Conference, 795–808.
- [7] S. Orlowski, M. Pióro, A. Tomaszewski and R. Wessäly, *SNDlib 1.0-Survivable Network Design Library*, Networks, Vol. 55, 2010, 276–286.
- [8] P. Mahadevan, P. Sharma and S. Banerjee, “*A Power Benchmarking Framework for Network Devices*”, Proceedings of IFIP Networking, 2009, 795–808.
- [9] L. Chiaraviglio and M. Mellia and F. Neri, “*Minimizing ISP Network Energy Cost: Formulation and Solutions*”, IEEE/ACM Trans. Netw, Vol. 20, 2012, 463–476.
- [10] A. Anand, A. Gupta, A. Akella, S. Seshan and S. Shenker, “*Packet Caches on Routers: the Implications of Universal Redundant Traffic Elimination*”, Proceedings of ACM SIGCOMM, 2008, 219–230.
- [11] T. Grevers and J. Christner, “*Application Acceleration and WAN Optimization Fundamentals*”, Cisco Press, 2007.
- [12] F. Barahona and A. Mahjoub, “*On the cut polytope*”, Math. Prog., Vol. 36, 1986, 157–173.

On the use of multiple sinks to extend the lifetime in connected wireless sensor networks

F. Castaño^{a,b,1} A. Rossi^a M. Sevaux^a N. Velasco^b

^a Université de Bretagne-Sud, Lab-STICC, Lorient, France

^b Universidad de los Andes, Departamento de Ingeniería Industrial, Bogotá, Colombia

Abstract

This paper addresses the maximum network lifetime problem in wireless sensor networks. In this problem, the purpose is to schedule a set of wireless sensors, keeping them connected and guaranteeing that all targets are covered, while network lifetime is maximized. Two variants of this problem are considered; in the first case it is assumed that a single sink (base station) is available, the second case considers the presence of several sinks. To solve the problem, a hybrid Column Generation-GRASP heuristic is proposed. The method is shown to be able to find optimal or near optimal solutions efficiently in both cases.

Keywords: Column Generation, GRASP, Wireless Sensor Networks, Lifetime, Multiple Sink

1 Introduction

A wireless sensor network (WSN) is a net made of a large amount of wireless battery-operated sensors used to accomplish a set of monitoring and compu-

¹ Email: fabian.castano@univ-ubs.fr

tation tasks. These devices work collaboratively or individually to perform their assigned tasks and deliver or spread the collected data to a remote base station through a multihop path of active sensors.

When the sensors and the base station are randomly placed, lifetime is limited by the sensors that are one hop away from the base station [2]. These sensors are bottleneck because they are required to pass all the information to the base station and their energy is drained faster than the energy of the distant sensors. A possible solution consists in deploying multiple base stations. Thus, sensors are used more efficiently by avoiding to transfer all sensing data in a single area.

In dense networks, lifetime can be maximized by creating covers, *i.e.*, groups of sensors that are active at the same time. This strategy has been proven to be efficient in several applications of WSN [3,4]. Following this idea, decomposition approaches as column generation (CG) have been largely used to identify and create schedules for the covers. As well as in the classical implementation, CG decomposes the problem into a restricted master problem (RMP) and an auxiliary problem (AP). The former optimizes the lifetime using an incomplete set of columns, and the latter is used to identify profitable columns.

In this paper, an hybrid CG-GRASP heuristic is proposed to maximize network lifetime. The GRASP [1] heuristic is used to solve AP identifying interesting covers. If GRASP fails to find an interesting column, CG-GRASP could be turned into an exact method by using exact approaches to solve AP. The paper is structured as follows. Section 2 introduces the notation and the problem. In section 3 the CG approach is presented and the GRASP heuristic used to solve AP is described. Finally, in Sections 4 and 5 the experimental results and conclusions are presented.

2 Problem description

Consider a set $\mathcal{K} = \{k_1, k_2, \dots, k_m\}$ of targets with known locations and a set $\mathcal{S} = \{s_1, s_2, s_3, \dots, s_n\}$ of sensors deployed to cover the targets. A target is considered covered by a sensor if it lies within its sensing range R_s (*observation link*). The sensor nodes are able to collect and (re)transmit the information to other sensors or a base station if they are within their communication range R_c (*communication link*). All the information retrieved from the targets must be collected by a set $\mathcal{B} = \{b_1, b_2, b_3, \dots, b_r\}$ of base stations.

Let E be the set of arcs, where $e(u, v) \in E$ exists if there exists a communication link between the elements $u, v \in \mathcal{S}$, $u \in \mathcal{S}$ and $v \in \mathcal{B}$ or if

there is an observation link between the elements $u \in \mathcal{K}$ and $v \in \mathcal{S}$. An additional, and artificial, super node Υ is defined to query the information collected at the sink nodes. In other words, a communication link exists between the base stations and the new super node ($\exists e(u, \Upsilon), \forall u \in \mathcal{B}$).

Let $G = (\mathcal{V}, E)$ be a directed graph with $\mathcal{V} = \mathcal{K} \cup \mathcal{S} \cup \mathcal{B} \cup \Upsilon$. A feasible cover $C_j \subseteq \mathcal{S}$ is a subset of sensors such that for all pairs (k_i, Υ) there exists a path through the elements of $C_j \cup \mathcal{B}$. The set of all the feasible covers of \mathcal{S} is denoted by $\Omega = \{C_1, C_2, \dots, C_l\}$. A decision variable t_j is introduced to identify the time interval allocated to a feasible cover C_j . The connected maximum network lifetime problem with multiple sink connectivity is to find a collection of pairs (C_j, t_j) in such a way that network lifetime, $\sum_{j \in \Omega' \subseteq \Omega} t_j$, is maximized.

3 Decomposition approach

CG based approaches are widely used to tackle the design of energy-efficient wireless sensor networks [2,3,4]. In this approach the problem is divided into two subproblems. First, RMP, containing a reduced set of the feasible columns (covers) $\Omega' \subseteq \Omega$, is used to identify the optimal timings for the current covers. RMP is solved using linear programming, and the optimal dual variables values found are used as an input to solve AP. This one, is used iteratively to identify new profitable covers to be included in RMP. For each new cover the reduced cost is evaluated. If it is greater than zero, which means that the cover is interesting, it is added to RMP and a new CG iteration is performed. The CG process stops when no more profitable columns are found.

3.1 Master problem

Let $y_{s_{ij}}$ be a binary parameter that takes the value of 1 if sensor s_i is active in cover C_j and 0 otherwise. The problem is formulated by:

$$\text{Maximize: } \sum_{C_j \in \Omega'} t_j \quad (1)$$

Subject to:

$$\sum_{C_j \in \Omega'} y_{s_{ij}} t_j \leq b_{s_i} \quad \forall s_i \in S \quad (2)$$

$$t_j \geq 0 \quad \forall C_j \in \Omega' \quad (3)$$

The objective function (1) maximizes the network lifetime. The set of equations (2) enforces that all energy constraints are satisfied. Constraints (3) are the non-negativity constraints.

3.2 Auxiliary Problem

AP is used to identify a connected structure in G that maximizes the reduced cost. We propose to model AP as a network flow problem. It is assumed that source nodes (targets) offer some information that is transferred to Υ through the transshipment nodes (sensors or the base stations). This means that similar approaches can be used to solve the problem with a single or multiple base stations. AP decides, based on the reduced cost criterion, the values y_{vj} that indicate if a sensor $v \in \mathcal{S}$ belongs to the new cover C_j . At each iteration of CG, the cost of using the sensor v is denoted by π_v , the dual variable value associated with constraints (2), which is used to compute the reduced cost of a feasible solution. For each $e(u, v) \in E$, variable x_{uv} indicates the flow of information transferred through this arc. It should be noted that, even although a feasible network structure must be defined, only the information concerning the optimal cover is required to be returned to MP.

By using the notation above, the auxiliary problem is represented by equations (4-10) as follows:

$$\text{Maximize: } 1 - \sum_{v \in \mathcal{S}} y_{vj} \pi_v \quad (4)$$

Subject to:

$$\sum_{u \in \mathcal{S} | \exists e(v, u)} x_{vu} = 1 \quad \forall v \in \mathcal{K} \quad (5)$$

$$\sum_{u \in \mathcal{V} | \exists e(v, u)} x_{vu} - \sum_{u \in \mathcal{V} | \exists e(u, v)} x_{uv} = 0 \quad \forall v \in \mathcal{S} \cup \mathcal{B} \quad (6)$$

$$\sum_{u \in \mathcal{V} | \exists e(u, \Upsilon)} x_{u\Upsilon} = |\mathcal{K}| \quad (7)$$

$$\sum_{u \in \mathcal{V} | \exists e(v, u)} x_{vu} + \sum_{u \in \mathcal{V} | \exists e(u, v)} x_{uv} \leq 2|\mathcal{K}|y_{vj} \quad \forall v \in \mathcal{S} \quad (8)$$

$$x_{uv} \in \mathbb{Z}^+ \cup \{0\} \quad \forall u, v \in \mathcal{V} \quad (9)$$

$$y_{vj} \in \{0, 1\} \quad \forall v \in \mathcal{V} \quad (10)$$

By solving model (4-10) a network structure with optimal reduced cost can be found (4). Equations (5-7) are information flow conservation constraints. These constraints are used to guarantee that the flow arising at the nodes is transferred to Υ through the sensors and the information is not stored in sensor nodes. Constraints (8) guarantee that sensors transferring information are labeled as active in a solution

3.3 A GRASP heuristic for the auxiliary problem

In this paper, the use of a GRASP [1] heuristic is proposed to solve AP. A general description of our approach is outlined in Algorithm 1. The method iteratively defines connected structures in G by using a greedy randomized construction procedure (line 4) and a local search to improve these ones (line 5). A parameter α is used to manage the level of randomization of the constructive phase. To stop the heuristic two criteria are used, the number of iterations without improvement (*Max_iters*) and the maximum running time (*Max_time*). Note that the objective of the algorithm is to minimize $\sum_{v \in S} y_{vj} \pi_v$ (lines 6-8), that maximizes the objective function for AP (3).

Algorithm 1 Greedy randomized adaptative search procedure

GRASP(π_v , G , α)

```

1:  $C_j \leftarrow \emptyset$ ;  $\hat{C}_j \leftarrow \emptyset$ ;  $Max\_time \leftarrow 0.5s$ ;  $time \leftarrow 0$ ;  $iter \leftarrow 0$ ;  $f(C_j) \leftarrow \infty$ 
2: while  $time \leq Max\_time$  &  $iter \leq MaxIters$  do
3:    $S_{av} \leftarrow \mathcal{S} \cup \mathcal{B}$ ,  $S_{act} \leftarrow \emptyset$ ,  $\mathcal{K}_{cov} \leftarrow \emptyset$ ,  $s_0 \leftarrow \Upsilon$ 
4:    $\hat{C}_j \leftarrow GR\_DFS(S_{av}, S_{act}, \mathcal{K}_{cov}, \pi_v, G, s_0, \alpha)$ 
5:    $\hat{C}_j \leftarrow Local\_Search(\hat{C}_j)$ 
6:   if  $f(\hat{C}_j) \leq f(C_j)$  then
7:      $C_j \leftarrow \hat{C}_j$ ;  $iter \leftarrow 0$ 
8:   end if
9:    $time \leftarrow Update\_time()$ ;  $iter \leftarrow iter + 1$ 
10: end while
11: return  $C_j$ 
```

3.3.1 Greedy randomized construction

The constructive phase uses a randomized greedy depth first search strategy (GR_DFS) to create connected structures in G . A general overview of the algorithm is presented in Algorithm 2. First, an initial tree containing only Υ is generated. Next, among the elements sharing a communication link with this, a restricted candidate list (RCL) of nodes is created (lines 2-8). A new

node belonging to this list is randomly selected (line 9), added to the tree and used as a reference to elaborate a new RCL. The targets with an observation link with it are marked as covered (lines 12-14). If RCL is an empty list, the previously visited nodes are used again to create the RCL and to continue the construct phase. GR_DFS stops when all the targets are marked as covered.

Algorithm 2 Greedy randomized depth first search

GR_DFS($S_{av}, S_{act}, \mathcal{K}_{cov}, \pi_v, G, s_0, \alpha$)

```

1: while  $|\mathcal{K}_{cov}| < |\mathcal{K}|$  do
2:    $RCL = \emptyset$ 
3:   for  $u \in B$  do
4:      $\pi_u \leftarrow \min_{v: \{\exists e(v,u) \wedge v \in S_{av}\}} \pi_v$ 
5:   end for
6:    $c_{min} = \min_{v: \{v \in S_{av} \wedge \exists e(v,s_0)\}} \pi_v$ 
7:    $c_{max} = \min_{v: \{v \in S_{av} \wedge \exists e(v,s_0)\}} \pi_v$ 
8:    $RCL \leftarrow \{v \in S_{av} \mid \exists e(v, s_0) \wedge \pi_v \leq c_{min} + \alpha(c_{max} - c_{min})\}$ 
9:    $s_0 \leftarrow \text{Random\_Selection}(RCL)$ 
10:   $S_{act} = S_{act} \cup s_0$ 
11:   $S_{av} = S_{av} \setminus s_0$ 
12:  for  $u \in \mathcal{K} \mid \exists e(u, s_0)$  do
13:     $\mathcal{K}_{cov} = \mathcal{K}_{cov} \cup u$ 
14:  end for
15:  GR_DFS( $S_{av}, S_{act}, \mathcal{K}_{cov}, \pi_v, G, s_0, \alpha$ )
16: end while
17: return  $S_{act}$ 

```

3.3.2 Local search

The local search procedure is based on the sequential exploration of two neighborhood structures following a first improvement strategy. First, a *remove neighborhood* is proposed to check for useless sensors that are not required to meet the constraints. Then, a *remove-insert neighborhood* is considered to exchange one active sensor by an inactive sensor with lower cost that keeps the feasibility of the cover.

4 Results and discussion

In order to evaluate the proposed method, four sets of instances with $|S|=50, 100, 150$, and 200 sensors are used. Two sets of targets $|\mathcal{K}| = 15, 30$ complete the instances. The effect of using multiple sinks is studied using up to 3

base stations. The sensors are assumed homogeneous and $b_{s_v} = 1 \forall v \in \mathcal{S}$. The approach is coded in C++ and the Gurobi optimization engine is used to solve RMP. Computational experiments are performed on an Intel Core i-5 processor @ 1.6 GHz with 2 GB of RAM running under OS-X Lion. When the GRASP heuristic is not able to find an interesting cover, AP is solved using an integer programming solver, so it is possible to check optimality; however, the additional time consumed is not reported.

Table 1 presents a summary of the results. The columns labeled **Avg. time(s)** present the average computational time required for CG-GRASP. The columns **Opt/#Ins** indicate the fraction of optimal solutions found with CG-GRASP at each instance group. The results show that, as expected, computational time increases with instance size. Furthermore, it is observed that the increase on the number of sinks increases as well as the average computational time to reach an optimal solution.

Instance		Sinks = 1		Sinks = 2		Sinks = 3	
$ \mathcal{S} $	$ \mathcal{K} $	Avg. time (s)	Opt/#Ins	Avg. time (s)	Opt/#Ins	Avg. time (s)	Opt/#Ins
50	15	0.19	1.00	0.36	1.00	0.41	1.00
100	15	20.02	0.80	77.24	0.80	79.57	0.60
150	15	50.54	0.80	87.06	0.80	69.60	0.80
200	15	20.56	1.00	96.38	0.60	113.14	0.60
50	30	0.24	1.00	0.47	1.00	0.40	1.00
100	30	30.61	0.80	85.48	0.80	65.28	0.80
150	30	47.21	0.80	114.18	0.60	127.49	0.60
200	30	29.69	1.00	60.20	0.80	41.88	0.60
Average		24.88	0.90	65.17	0.80	62.22	0.75

Table 1
Computational results for CG-GRASP.

Solving the problem by using a pure CG method requires extensive computational effort. The detailed results of the pure CG approach can be found at the authors website (<http://or-labsticc.univ-ubs.fr/>). These results are used as a basis for counting the number of problem instances for which CG-GRASP finds the optimal solution. The ratio of optimal solutions found is shown on columns 4, 6 and 8 of Table 1.

An additional set of experiments has been performed by using the set of instances proposed by Raiconi and Gentili [3]. CG-GRASP is compared with the CMLP-GRASP heuristic proposed by the authors. As presented in our previous work, the results indicate that CG-GRASP outperforms their results in terms of solution quality and computational time [5].

5 Conclusions

This paper addresses the problem of maximizing lifetime of connected wireless sensor networks with single and multiple sinks. The proposed solution approach combines column generation with GRASP to identify profitable covers and extend network lifetime. Furthermore, the GRASP heuristic is shown to be efficient to tackle the auxiliary problem arising at each iteration of CG.

The experimental results confirm that the method is able to obtain near optimal solutions in low computational times. The experiments show that the average time required for CG-GRASP was below 2.5 minutes for the larger instances. Moreover, it is observed that the optimal solution was found in most experiments.

Future research will consider extensions to problems with partial coverage. Additionally, similar approaches will be applied to problems including the effect of distance sensor-sensor and sensor-target on the energy consumed by transmission and detection respectively.

References

- [1] T. Feo and M. Resende. Greedy randomized adaptive search procedures. *Journal of Global Optimization*, 6:109–133, 1995.
- [2] Y. Gu, B. Zhao, Y. Ji, and J. Li. Theoretical treatment of target coverage in wireless sensor networks. *Journal of Computer Science and Technology*, 26:117–129, 2011.
- [3] A. Raiconi and M. Gentili. Exact and metaheuristic approaches to extend lifetime and maintain connectivity in wireless sensors networks. In *Network Optimization*, volume 6701 of *Lecture Notes in Computer Science*, pages 607–619. Hamburg, Germany, 2011.
- [4] A. Singh, A. Rossi, and M. Sevaux. Matheuristic approaches for Q-coverage problem versions in wireless sensor networks. *Engineering Optimization*, To appear in 2013.
- [5] F. Castaño, A. Rossi, M. Sevaux and N. Velasco. A hybrid column generation-GRASP approach to extend the lifetime in wireless sensor networks with coverage and connectivity constraints. *14 conférence ROADEF de la société Française de Recherche Opérationnelle et Aide à la Décision*, <http://www-labsticc.univ-ubs.fr/sevaux/Publications/p-castano-13-slides.pdf>

A Flow Formulation for the Optimum Communication Spanning Tree

Elena Fernández, Carlos Luna-Mota^{1,2}

*Department of Statistics and Operations Research
Universitat Politècnica de Catalunya - Barcelona Tech
Barcelona, Spain*

Achim Hildenbrandt, Gerhard Reinelt, Stefan Wiesberg^{3,4,5}

*Institut für Informatik
Universität Heidelberg
Heidelberg, Germany*

Abstract

In this paper we address the Optimum Communication Spanning Tree Problem. We present a formulation that uses three index variables and we propose several families of inequalities, which can be used to reinforce the formulation. Preliminary computational experiments are very promising.

Keywords: Spanning tree, Optimum Communication Spanning Tree.

¹ Email: e.fernandez@upc.edu

² Email: carlos.luna-mota@upc.edu

³ Email: achim.hildenbrandt@informatik.uni-heidelberg.de

⁴ Email: gerhard.reinelt@informatik.uni-heidelberg.de

⁵ Email: stefan.wiesberg@informatik.uni-heidelberg.de

1 Introduction

In the *Optimum Communication Spanning Tree Problem* (OCSTP) a given set of n origin/demand points with communication requirements between them must be connected by a tree. The communication cost between an origin/destination pair, depends both on the communication requirement and on the distance on the tree between the nodes of the pair. The OCSTP is to find a spanning tree of minimum total communication cost.

The OCSTP was originally introduced by Hu [9]. Despite of its apparent simplicity, the OCSTP turns out to be a very challenging combinatorial optimization problem. Johnson, Lenstra, and Rinnooy Kan [10] proved that finding an OCST is \mathcal{NP} -hard, even if all origin/destination pairs have the same communication requirement. Moreover, unless $\mathcal{NP} = \mathcal{P}$, no polynomial time approximation scheme exists since the problem is $\mathcal{MAX SNP}$ -hard [12]. Yet, [9] presented two particular cases of the OCSTP defined on a complete graph that can be solved in polynomial time. The first one is the Optimum Requirement Spanning Tree Problem where, in the original graph, the distance between all pairs of nodes is the same. In this case, an optimal solution is given by a Gomory-Hu tree of an associated network. The second one is the Optimum Distance Spanning Tree Problem, where all pairs of nodes have the same communication requirements. Now, under some additional assumptions, the optimal solution has a star topology.

In addition to a broad range of applications within telecommunication, transportation, and computer networks, the OCST appears as a subproblem in some Network Hub Location Problems. For instance, in the Tree-of-Hubs Location Problem (THLP) (see [5,6]), when both the set of hubs and the allocation pattern of nodes to hubs are known, the problem can be polynomially transformed into an OCSTP. Thus, efficient solution procedures for the OCSTP may also serve as subroutines for solution procedures for the THLP.

To the best of our knowledge, there is only one algorithm, due to Ahuja and Murty [1], for optimally solving the general case of the OCSTP. It is a combinatorial branch & bound algorithm, which uses lower planes to obtain a valid lower bound at every node of the branch & bound tree. With this algorithm the authors were able to prove the optimality of the obtained solutions for sparse instances with up to 40 nodes. Rothlauf [13] modeled the OCSTP as an integer linear program, which was able to obtain optimal solutions for instances with up to 12 nodes in reasonable computational times. Contreras [2] (see also [3]) proposed another integer programming formulation which, enforced with additional valid inequalities, was able to produce optimal so-

lutions for instances with up to 25 nodes in reasonable computational times. Contreras, Fernández and Marín [4] presented a Lagrangean relaxation which produced good lower and upper bounds for instances with up to 50 nodes. To the best of our knowledge no exact algorithm exists able to solve medium sized instances for the general case.

One of the main issues in the search for a successful formulation for the OCSTP is to find a trade off between the number of variables in the formulation and the tightness of the Linear Programming (LP) bound. It is known that formulations with path-based 4 index variables produce very tight lower bounds for the OCSTP [4]. However, the main drawback of such formulations is the number of variables they require, since each origin/demand pair requires as many variables as arcs the network has. In practice, this makes it impossible to solve instances with more than 30 nodes with a general purpose solver. On the other hand, it is possible to formulate the OCSTP using only 2 index variables [7] but, for the moment, such formulations are not tight enough so as to solve instances with more than 25 nodes in reasonable computing times. In this paper we study formulations for the OCSTP with 3 index variables, which can be seen as a compromise between the above two types of formulations. For each fixed origin, the 3 index variables are obtained by aggregating over all possible destinations the path-based 4 index variables. The resulting formulation models the intersection of a series of spanning trees, each of them rooted at one different origin, in the spirit of recent formulations for other problems [8,11]. Basic versions of 3 index formulations are still incapable of solving moderate size instances. Nevertheless, preliminary results indicate that they can be reinforced with different types of valid inequalities so as to yield promising results.

The structure of the paper is the following. In Section 2 we first recall the formal definition of the Optimum Communication Spanning Tree Problem. Then, we introduce the basic 3 index formulation for the OCSTP with which we will work. Section 3 presents different types of valid inequalities, which reinforce the basic formulation. In Section 4 we conclude the paper with some comments and guidelines for further research.

2 Optimum Communication Spanning Trees

The Optimum Communication Spanning Tree Problem (**OCSTP**) is formally defined as follows. Let $G = (V, E)$ be a complete undirected graph with $n = |V|$ nodes and $m = |E|$ edges, $c : E \rightarrow \mathbb{R}^+$ a cost function over the edges, and $r : V \times V \rightarrow \mathbb{R}^+ \cup \{0\}$ a communication requirement function between pairs of

nodes. Without loss of generality we assume that $r_{ji} = 0$ for $j > i$. If needed we redefine r_{ij} as $r_{ij} + r_{ji}$ for all $i, j \in V, i < j$.

The solution space of the OCSTP is the set of all spanning trees of G . For a given spanning tree T of G ,

the communication cost over T of a pair of nodes $i, j \in V, i \neq j$, is defined as the cost (length) with respect to c of the (unique) path in T that connects i and j , c_{ij}^T , multiplied by the communication requirement between the pair: $c_{ij}^T r_{ij}$.

the total communication cost of T is the sum of the communication costs over T of all pairs of nodes: $\sum_{i \neq j} c_{ij}^T r_{ij}$. Note that the total communication cost of T can be evaluated in $O(n^2)$ time.

The OCSTP is to find a spanning tree of G of minimum total communication cost.

Remark 2.1 There is an alternative way of computing the communication cost of a tree T , which takes into account that T contains a unique *communication path* connecting each origin/destination pair. Let f_{ij}^T denote the total amount of flow that circulates through any of the two directions of (i, j) . Note that f_{ij}^T is the sum of the communication requirements of all the origin/destination pairs that use either (i, j) or (j, i) in their communication path. Then, the communication cost of T can also be computed as $\sum_{i,j \in V: j_1} c_{ij} f_{ij}^T$.

To formulate the OCST we use a set of binary variables which indicate the edges that are used in the tree. For all $i, j \in V, i < j$, let $x_{ij} = 1$ if edge (i, j) is in in the tree, and 0 otherwise. In addition, for each $u \in V$ we define a continuous set of variables, denoted $f_{u ij}$, to indicate how the flow $O_u = \sum_{v > u} r_{uv}$ originated at u circulates through the tree defined by x . For $u, i, j \in V, i \neq j$, $f_{u ij}$ indicates the flow with origin in u through arc (i, j) . To represent the arcs that are used for sending the flow originated at u , we define for all $i, j \in V, i \neq j$, a binary decision variable $y_{u ij}$ which takes the value 1 if arc (i, j) is used for sending the flow originated at u . Then, a valid formulation for the OCST is:

$$(1) \quad \min \sum_{i,j \in V; i < j} c_{ij} \sum_{u \in V} (f_{u ij} + f_{u ji})$$

$$(2) \quad \sum_{j \in V \setminus \{u\}} f_{uuj} = O_u \quad u \in V$$

$$(3) \quad \sum_{j \in V \setminus \{i, u\}} f_{u ij} - \sum_{j \in V; j \neq i, j \neq u} f_{u ji} = -r_{ui} \quad u, i \in V$$

- $$(4) \quad f_{uij} \leq M y_{uij} \quad u, i, j \in V, j > i$$
- $$(5) \quad f_{uji} \leq M y_{uji} \quad u, i, j \in V, j > i$$
- $$(6) \quad \sum_{i,j \in V; j \neq i, j \neq u} y_{uij} = n - 1 \quad u \in V$$
- $$(7) \quad y_{uij} + y_{uji} \leq x_{ij} \quad u, i, j \in V, j > i$$
- $$(8) \quad \sum_{i,j \in V; j > i} x_{ij} = n - 1$$
- $$(9) \quad x_{ij}, y_{uij} \in \{0, 1\}, f_{uij} \geq 0 \quad i, j, u \in V, j > i$$

For each $u \in V$, constraints (2)–(3) model a flow of value O_u with origin at u and destination r_{uv} for $v \in V$. Constraints (4) and (5) relate the f and the y variables by activating the arcs that are used for routing the flows. Together with cardinality constraints (6), constraints (2)–(5) guarantee that, for each $u \in V$, the arcs used for sending the flow O_u define a spanning tree. Constraints (7) relate the x and the y variables by imposing that if an arc is used for sending some flow, then associated edge is in the tree. These constraints will hold as equality for instances where communication exists between all pairs of nodes, i.e. $r_{ij} + r_{ji} > 0$ for all $i, j \in V, j > i$. Otherwise, some of these constraints will hold as strict inequality. Finally, constraint (8) guarantees that all the flows define the same spanning tree, by limiting the total number of edges to $n - 1$.

In constraints (4) and (5) M is a sufficiently large constant, whose value may affect significantly the effectiveness of the constraints. We use $M = O_u - rm_u$, where $rm_u = \min_{v>u} r_{uv}$.

Indeed, there are several alternatives for modeling the relation between the f and x variables. In our formulation, this is done via the intermediate y variables and constraints (4)–(5) and (7). An alternative would be to omit the y variables and to substitute the above three sets of constraints by

$$(10) \quad f_{uij} + f_{uji} \leq M x_{ij} \quad u, i, j \in V, j > i.$$

However, for instances where communication requirements do not exist between all pairs of nodes, our formulation may give is tighter LP bounds because, as mentioned, some inequalities (7) will hold as strict inequality.

3 Valid inequalities

In this section we present several families of valid inequalities that can be used to reinforce formulation (1)–(9).

(a) **Vertex cutset inequalities.** For all $u \in V$ at least one arc must leave

vertex u , i.e.

$$\sum_{j \in V \setminus \{u\}} y_{uuj} \geq 1.$$

On the other hand, for all $u, i \in V$, $u \neq i$ exactly one arc associated with the flow emanating from u must enter i , i.e.

$$\sum_{j \in V \setminus \{i\}} y_{uji} = 1.$$

(b) **Set cutset inequalities.** For $S \subset V$

$$x(\delta(S)) \geq 1.$$

(c) **Min-cut values inequalities.** Let m_{ij} denote the value of the min-cut separating i and j in the original graph, relative to a capacity vector given by r . For all $i, j \in V$, $i < j$, the following inequalities are valid:

$$\begin{aligned} m_{ij}x_{ij} &\leq \sum_{u \in V \setminus \{j\}} f_{uoj} + \sum_{u \in V \setminus \{i\}} f_{uji}. \\ \sum_{u \in V \setminus \{j\}} r_{uj}y_{uoj} + \sum_{u \in V \setminus \{i\}} r_{ui}y_{uji} &\leq \sum_{u \in V \setminus \{j\}} f_{uoj} + \sum_{u \in V \setminus \{i\}} f_{uji}. \end{aligned}$$

(d) **Minimum flows through arcs.** For $u, i, j \in V$, $i < j$, the following inequality must hold:

$$r_{uj}y_{uoj} + r_{ui}y_{uji} \leq f_{uoj} + f_{uji}.$$

Let $k_1 = \max\{m_{uj}, m_{ij}\}$ and $k_2 = \max\{m_{iu}, m_{ij}\}$. The following lower bound on the flows must hold:

$$k_1y_{uoj} + k_2y_{uji} \leq \sum_{u \in V \setminus \{j\}} f_{uoj} + \sum_{u \in V \setminus \{i\}} f_{uji}.$$

4 Comments and guidelines for further research

We have run a series of preliminary experiments with benchmark instances from the literature using a general purpose commercial solver.

Broadly speaking the obtained numerical results indicate the following:

- The bound of the LP relaxation of formulation (1)–(9) lies within 60–70% of the optimal value. As could be expected, the quality of the bound seems to deteriorate as the size of the instances increase.
- The effect of the cutset inequalities presented in items (a) and (b) seem to be very limited. For the moment we have no indication that they can be useful for the proposed formulation.
- The inequalities which seem to have more impact are the ones of item (c), derived from min-cut values. Depending on the instance, these inequalities reinforce the lower bound to 80–95% of the optimal value.
- The inequalities imposing lower bounds on the flows through used arcs, presented in item (d) also improve the quality of the lower bound. This additional improvement is larger in the instances for which the min-cut values inequalities were not so effective. When both families are added the obtained lower bounds are beyond 90% of the optimal values.
- The computing times can be improved by removing constraints (b) and (c) from the original formulation and by dynamically separating them. That is, they are incorporated to the formulation only if they are violated by the current LP solution.

From the results obtained so far it is clear that further research is needed to assess the interest of the proposed formulation. The obtained results suggest exploring further types of inequalities derived from min-cut values. Another avenue of research focuses on obtaining tight bounds for the costs of the flows through the used origin/destination paths.

Acknowledgement

This research has been partially funded through joint grants AIB2010DE-00137 by the Spanish Ministry of Science and Education, and DDAD50749416 by the German Academic Exchange Service in the Project-based Personnel Exchange Program. The research of E. Fernández and C. Luna-Mota has been partially funded through grant MTM2009–14039–C06–05 of the Spanish Ministry of Science and Education and ERDF funds.

References

- [1] Ahuja, R.K., and V.V.S. Murty, *Exact and Heuristic Algorithms for the Optimum Communication Spanning Tree Problem*, *Transportation Science* **21**

(1987), 163–170.

- [2] Contreras, I., “Network Hub Location – Models, Algorithms, and Related Problems”, Ph.D. thesis, Universitat Politècnica de Catalunya, 2009.
- [3] Contreras, I., and E. Fernández, *General Network Design: A Unified View of Combined Location and Network Design Problems*, European Journal of Operational Research **219** (2012), 680–697.
- [4] Contreras, I. and E. Fernández, and A. Marín, *Lagrangian bounds for the optimum communication spanning tree problem*, TOP **18** (2010), 140–157.
- [5] Contreras, I. and E. Fernández, and A. Marín, *Tight bounds from a path based formulation for the Tree-of-Hubs Location Problem*, Computers & Operations Research **36** (2010), 3117–3127.
- [6] Contreras, I. and E. Fernández, and A. Marín, *The tree-of-hubs location problem: A comparison of formulations*, European Journal of Operational Research (202), 390–400.
- [7] Fernández, E., and C. Luna-Mota, and G. Reinelt, *A compact formulation for the optimum communication spanning tree problem*, International Symposium on Mathematical Programming (ISMP2012), Berlin, 2012.
- [8] Gouveia, L. and J. Telhada, *Reformulation by Intersection Method on the MST Problem with Lower Bound on the Number of Leaves*, Lecture Notes in Computer Science, Proceedings of INOC 2011 (2011) **6701** 83–91.
- [9] Hu, T.C., *Optimum Communication Spanning Trees*, SIAM J. Comput. **3** 1974,(188–195).
- [10] Johnson, D.S., and J.K. Lenstra, and A.H.G. Rinnooy Kan, *The Complexity of the Network Design Problem*, Networks **8** (1978), 279–285.
- [11] Martinez, L., and A. Salles da Cunha, *The min-degree constrained minimum spanning tree problem: Formulations and Branch-and-cut algorithm*, Discrete Applied Mathematics (special issue dedicated to ISCO 2010), to appear.
- [12] Papadimitriou, C., and M. Yannakakis, *Optimization, approximation, and complexity classes*, STOC 88: Proceedings of the twentieth annual ACM symposium on theory of computing (1988), 229–234.
- [13] Rothlauf, F., *On Optimal Solutions for the Optimal Communication Spanning Tree Problem*, Operations Research **57** (2009), 413–425.

Dynamic trees in exterior-point Simplex-type algorithms for network flow problems

George Geranis¹

*Department of Applied Informatics
University of Macedonia
156 Egnatia Str., 54006 Thessaloniki, Greece*

Angelo Sifaleras^{2,3}

*Department of Technology Management
University of Macedonia
Loggou-Tourpali, 59200 Naoussa, Greece*

Abstract

Recently, a new Dual Network Exterior-Point Simplex Algorithm (DNEPSA) for the Minimum Cost Network Flow Problem (MCNFP) has been developed. In extensive computational studies, DNEPSA performed better than the classical Dual Network Simplex Algorithm (DNSA). In this paper, we present for the first time how to utilize the dynamic trees data structure in the DNEPSA algorithm, in order to achieve an improvement of the amortized complexity per pivot. Our work constitutes a first step towards the development of an efficient implementation of DNEPSA.

Keywords: Network Optimization, Computational Complexity, Data Structures.

¹ Email: geranis@uom.gr

² Email: sifalera@uom.gr

³ This research has been partly funded by the Research Committee of the University of

1 Introduction

Network optimization [1] is a core area of combinatorial optimization, consisting of optimization problems that can be modeled using networks. Important theoretical improvements in network optimization algorithms have been achieved using well-known efficient data structures like dynamic trees [9], or Fibonacci heaps [2]. The MCNFP [8] is the problem of finding a minimum cost flow of product units, through a number of source nodes, sinks and transshipment nodes.

Let $G = (N, A)$ be a directed network, where two finite sets N and A consist of n nodes and m directed arcs, respectively. For each node $i \in N$, there is an associated numeric value b_i . A node i is a source node if it is $b_i > 0$, a sink node if it is $b_i < 0$ and a transshipment node otherwise. The problem will be called balanced, if the total supply is equal to the total demand, i.e., $\sum_{i \in N} b_i = 0$. For each arc $(i, j) \in A$, there is an associated flow x_{ij} and an associated cost c_{ij} , showing the amount of product units transferred from node i to node j and the flow cost per product unit, respectively. An optimal solution of the problem is a flow consisting of arc flows $x_{ij} \geq 0$ that minimize the total cost $\sum_{(i,j) \in A} c_{ij}x_{ij}$, subject to the flow conservation equations $\sum_{(i,j) \in A} x_{ij} - \sum_{(j,i) \in A} x_{ji} = b_i, \forall i \in N$. In this paper, DNEPSA algorithm with dynamic trees, is applied to uncapacitated MCNFPs (i.e., $0 \leq x_{ij} \leq +\infty$). The dual problem can be formulated using a set of dual variables w_i and a set of reduced cost variables s_{ij} , one for each node $i \in N$ and for each arc $(i, j) \in A$, respectively.

A number of different algorithmic approaches have been developed for the MCNFP, including the well-known Primal and Dual Network Simplex Methods. A primal exterior-point Simplex-type algorithm for the MCNFP was presented in [7] extending other approaches, e.g., to the assignment problem [6]. The DNEPSA [3] for the MCNFP is an exterior Simplex-type algorithm that starts from an initial dual feasible tree-solution and, after a number of iterations, it reaches an optimal solution by producing a sequence of tree-solutions that can be both dual and primal infeasible. This paper describes the representation of the tree-solutions produced by DNEPSA by using dynamic trees. Section 2 presents a short description of DNEPSA. The tree operations that improve the algorithm's theoretical performance, are described in detail for each step of the algorithm in Section 3. Finally, Section 4 presents conclusions and ongoing research efforts.

Macedonia, Economic and Social Sciences, Greece, under grant 80749 for the advance of Basic Research.

2 Description of DNEPSA

Computational results [3] on randomly generated network problems, have already shown the superiority of DNEPSA to DNSA (both on the number of the iterations and on the time needed in order to find an optimal solution), even using the Augmented Thread Index method (ATI method) [4] rather than the more efficient data structure of dynamic trees. However, it is well known that the use of special data structures for storing and updating the necessary variables can improve the algorithm's amortized complexity and also its computational performance. The steps of DNEPSA are as follows:

Step 0 (Initialization). Compute a starting dual feasible tree solution, i.e., the flows x_{ij} , the node potentials (dual variables) w_i , and the reduced costs s_{ij} , as described in [5]. Start with a partition of the basic arcs into two subsets as follows $I_- = \{(i, j) \in T : x_{ij} < 0\}$ and $I_+ = \{(i, j) \in T : x_{ij} \geq 0\}$. Compute a direction flow vector d , towards the feasible region of the dual problem, using the following relations: $d(I_-) = 1, d(I_+) = 0, d_{ij} = \sum_{(u,v) \in I_-} h_{uv}, \forall (i,j) \notin T$.

Step 1 (Termination test). If $I_- = \emptyset$, this means that the current tree-solution is optimal. Otherwise, create a set $J_- = \{(i, j) \notin T : s_{ij} > 0 \wedge d_{ij} < 0\}$. If $(J_- = \emptyset) \wedge (I_- \neq \emptyset)$, then the problem is infeasible.

Step 2 (Choice of entering arc). Choose the entering arc (g, h) using the following minimum ratio test: $\alpha = \frac{s_{gh}}{-d_{gh}} = \min \left\{ \frac{s_{ij}}{-d_{ij}} : (i, j) \in J_- \right\}$.

Step 3 (Choice of leaving arc). Find the minimum ratios: $\theta_1 = -x_{k_1 l_1} = \min \{-x_{ij} : (i, j) \in I_- \wedge (i, j) \uparrow\uparrow (g, h)\}$ and $\theta_2 = x_{k_2 l_2} = \min \{x_{ij} : (i, j) \in I_+ \wedge (i, j) \uparrow\downarrow (g, h)\}$. Choose the leaving arc $(g, h) = (k_1, l_1)$ or (k_2, l_2) , in case that $\theta_1 \leq \theta_2$ or not respectively.

Step 4 (Pivoting). If $\theta_1 \leq \theta_2$, the pivot will be called “Type A iteration”. In this case, set $I_- = I_- \setminus (k, l)$ and $I_+ = I_+ \cup (g, h)$. An arc of negative flow $x_{kl} = -\theta_1$ is leaving the basic tree-solution T and an arc of positive flow $x_{gh} = \theta_1$ is entering T . Otherwise, if it is $\theta_1 > \theta_2$, then $(k, l) = (k_2, l_2)$ is the leaving arc and we have a *type B iteration*. In this case, set $I_+ = I_+ \cup (g, h) \setminus (k, l)$. An arc of positive flow $x_{kl} = \theta_2$ is leaving T and an arc with positive flow $x_{gh} = \theta_2$ is entering T . Let T^+ be the subtree containing node k and T^- be the subtree containing node l . Values x_{ij} , s_{ij} , d_{ij} or sets I_- , and I_+ can be efficiently updated from iteration to iteration as described in [3], rather than computed from scratch in each iteration.

An entering and a leaving arc can be found in $O(m)$ and $O(n)$ time, respectively. Therefore, since updating the basic tree solution using ATI method requires $O(n)$ time [1], a single pivot of DNEPSA using the ATI method requires $O(m + n)$ time.

3 Using Dynamic Trees in DNEPSA

Dynamic trees [9], are data structures that maintain a set of vertex disjoint rooted trees and allow us to easily change their structure by using mainly three kinds of operations:

- $link(u, v)$: links together two dynamic trees, where u is the root of the first tree and v a vertex of the second tree, by adding the edge (u, v) .
- $cut(v)$: divides a dynamic tree into two subtrees by deleting the edge that connects v to its parent.
- $evert(v)$: makes v the root of the tree by turning the tree “inside out”.

Beyond these three basic operations, there are other useful operations that can be implemented, such as:

- $parent(v)$: returns the parent of a vertex.
- $root(v)$: returns the root of the dynamic tree containing v .
- $cost(v)$: returns the cost of the edge $(v, parent(v))$.
- $mincost(v)$: finds the edge of minimum cost on the path joining v to $root(v)$.
- $update(v, x)$: adds value x to all edges on the path from v to $root(v)$.

A dynamic tree can be partitioned into a collection of vertex-disjoint dynamic paths by dividing its edges into two types: solid and dashed edges. At most one solid path can enter any vertex of the tree. Thus, a dynamic path is determined by a sequence of solid edges and a dynamic tree is partitioned into a number of dynamic paths connected together by dashed edges. Each tree-solution is a collection of dynamic paths connected to each other by dashed lines where, each dynamic path is implemented by a biased binary tree. A basic tree solution in DNEPSA can be represented by a directed dynamic tree consisting of a set of vertex-disjoint dynamic paths. Figure 1 shows a possible initial dual feasible tree-solution used by DNEPSA in order to solve a MC-NFP problem. The value next to a vertex v represents its supply or demand and the value next to an arc (i, j) represent the flow x_{ij} for that arc. Such a tree-solution can be built by a special algorithm, like the algorithm described in [5], that starts from a collection of $|N|$ single-vertex dynamic trees and uses link operations to construct the final tree. The type of the arcs of the tree (solid or dashed) depends on the order that linking operations take place.

The dynamic tree in Figure 1 consists of three dynamic paths: $(8, 2, 4, 6)$, $(7, 3)$, and $(5, 1)$. The head of a path is its bottommost vertex and the tail of the path is its topmost vertex. A set of primitive dynamic path operations have to be implemented:

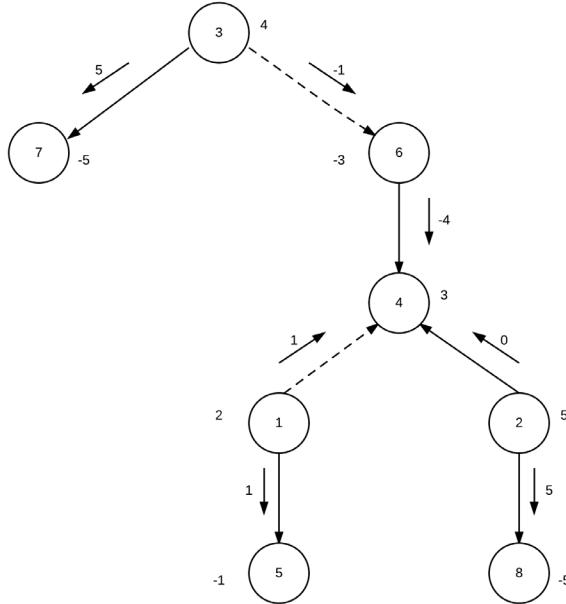


Fig. 1. Representation of the initial dual feasible tree as a dynamic tree.

- $path(v)$: returns the path containing vertex v .
- $head(p)$ and $tail(p)$: return the first and the last node of path p .
- $before(v)$ and $after(v)$: return the previous and next node of v .
- $pcost(v)$: returns the cost of edge $(v, after(v))$.
- $pmincost(p)$: finds a node v on path p so that the edge $(v, after(v))$ has the minimum cost.
- $pupdate(p, x)$: adds x to the cost of every edge of path p .
- $reverse(p)$: reverses the direction of the path.
- $concatenate(p, q, x)$: concatenates two paths p and q by adding a new edge of cost x .
- $split(v)$: deletes edges $(before(v), v)$ and $(v, after(v))$ producing two subpaths.

In addition, two composite path operations are needed:

- $splice(p)$: transforms the dashed edge leaving $tail(p)$ into a solid edge; thus the path is being extended this way.
- $expose(v)$: transforms every dashed edge from v to $root(v)$ into a solid edge.

These composite operations are implemented by using the primitive operations described before. The dynamic tree operations we need, are implemented by using some of the above primitive and composite path operations. Dynamic

paths can be implemented as biased binary trees. Figure 2 shows the representation of the dynamic tree in Figure 1 as three different binary trees, one for each dynamic path. The external nodes of a dynamic path in left-to-right order correspond to the vertices of the path from head to tail. An internal node w of the tree corresponds to the edge of the path that connects nodes u and v , where u and v are the previous and next node of the tree when it is traversed in symmetric order (inorder traversal). For all vertices of the dynamic tree with an outgoing dashed line, there is a field named *dparent* that connects a vertex to its parent in the original dynamic tree and a field named *dcost* that stores the corresponding cost.

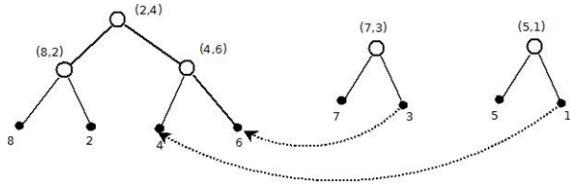


Fig. 2. Representation of dynamic paths $(8, 2, 4, 6)$, $(7, 3)$ and $(5,1)$ as binary trees.

For each node of the binary trees used to represent dynamic paths, a number of fields can be used in the data structure to store the required information:

- *external* field: indicates if the node is internal or external (values 0 or 1).
- *direction* field: stores the direction of the arc that corresponds to an internal node.
- *reversed* field: shows if an arc was reversed.
- *bparent* field: points to the parent of a node (it is null for the root).
- *bhead* and *btail* fields: point to the head and tail of the path.
- *bleft* and *bright* fields: point to the left and right child of an internal node of the tree.

When used in DNEPSA, we need more fields in order to store information about the flow of the arc that corresponds to an internal node and a flag indicating whether it belongs to I_- or I_+ . We can also store supply values b_i and dual variable values w_i in every external node of a binary tree.

Sleator and Tarjan [9] proved that $O(m \log n)$ path operations (by analyzing each compound operation to the respective component operations) exist in a series of m dynamic tree operations. Furthermore, they proved that a sequence of m dynamic tree operations requires $O(m \log n)$ time, by representing solid paths as locally biased binary trees. Thus, by using dynamic trees and utilizing these results it is possible to improve the amortized time complexity of DNEPSA per iteration, as it is described by the following Theorem 3.1.

Theorem 3.1 *DNEPSA has an $O(m + \log n)$ amortized time complexity bound per iteration.*

Proof.

In step 1, if $I_- \neq \emptyset$ then we need to compute the J_- set, that requires $O(m)$ time. In step 2, both the computation of J_- set as also finding the entering arc (g, h) using a minimum ratio α , also require $O(m)$ time.

In step 3, DNEPSA has to choose the leaving arc. The flows in cycle C existing in $T \cup (g, h)$ have to be checked so that θ_1 and θ_2 are computed. By using dynamic trees, in order to compute θ_1 and θ_2 , we can make h to be the root of the dynamic tree (by using operation $evert(h)$) and then we just traverse the path from node g up to the root. For every arc (i, j) on that path, it is enough to check if it belongs to I_- or I_+ and compare its orientation to the orientation of (g, h) . Again the time complexity is the same as the *evert* tree operation, i.e., $O(\log n)$.

In step 4, after finding the entering and the leaving arc, the new tree-solution $T \setminus (k, l) \cup (g, h)$ has to be built. This can be easily done by using the *cut* and *link* dynamic tree operations. The entering arc (g, h) is added into I_+ by setting the proper bit in the node's structure and its flow x_{gh} becomes equal to θ_1 or θ_2 , depending on the type of iteration (*type A* or *type B*). The flows x_{ij} for the basic arcs $(i, j) \notin C$ do not change. On the other hand, for a basic arc $(i, j) \in C$ the flow x_{ij} changes and its new value $x_{ij}^{(t+1)}$ is either $x_{ij}^{(t)} - x_{kl}^{(t)}$ or $x_{ij}^{(t)} + x_{kl}^{(t)}$ depending on the orientation of (i, j) compared to the orientation of (g, h) . Dynamic tree operations, in the same way as previously described, can help in determining the orientation of an arc $(i, j) \in C$ in $O(\log n)$ time. Vectors s and d can also be updated for all non basic arcs $(i, j) \notin T$. If the leaving arc (k, l) is deleted from the basic tree T , then subtrees T^+ and T^- are created. Values s_{ij} and d_{ij} do not change when both i and j belong to T^+ or both belong to T^- . If $i \in T^+ \cup j \in T^-$ or $i \in T^- \cup j \in T^+$, then it is either $s_{ij}^{(t+1)} = s_{ij}^{(t)} - s_{gh}^{(t)}$ or $s_{ij}^{(t+1)} = s_{ij}^{(t)} + s_{gh}^{(t)}$ depending on the type of iteration. The same happens for the new value of d_{ij} for every non basic arc (i, j) . By using dynamic trees, it is easy to determine for nodes i and j whether they belong to subtree T^+ or T^- . This can be done by using operation $cut(l)$ to produce subtrees T^+ and T^- and then by checking if $root(i)$ is the same as $root(j)$. Again the time complexity for these tree operations is $O(\log n)$.

Thus, DNEPSA overall has an $O(m + \log n)$ amortized time complexity bound per iteration, by using dynamic trees. \square

4 Conclusions and Future Work

By using dynamic trees in DNEPSA as described in the previous section, the amortized time complexity per pivot is reduced from $O(m+n)$ to $O(m+\log n)$, giving an improvement over the previous algorithm implementation using the ATI method.

Apart from the theoretical findings, regarding the amortized time complexity bound per iteration, experimental results demonstrating the efficiency of the proposed approach are also of equal importance. Therefore, our ongoing research efforts focus on computational testing of the DNEPSA performance using dynamic trees in order to provide empirical support for the reduction in time per iteration.

References

- [1] Ahuja, R. K., T. L. Magnanti and J. B. Orlin, “Network Flows: Theory, Algorithms and Applications” Prentice Hall, Englewood Cliffs, NJ, 1993.
- [2] Fredman, M. L. and R. E. Tarjan, *Fibonacci heaps and their uses in improved network optimization algorithms*, Journal of the ACM **34** (1987), pp. 596–615.
- [3] Geranis, G., K. Paparrizos and A. Sifaleras, *On a dual network exterior point simplex type algorithm and its computational behavior*, RAIRO - Operations Research **46** (2012), pp. 211–234.
- [4] Glover, F., D. Karney and D. Klingman, *The augmented predecessor index method for locating stepping stone paths and assigning dual prices in distribution problems*, Transportation Science **6** (1972), pp. 171–180.
- [5] Hultz, J., D. Klingman and R. Russell, *An advanced dual basic feasible solution for a class of capacitated generalized networks*, Operations research **24** (1976), pp. 301–313.
- [6] Papamanthou, C., K. Paparrizos, N. Samaras and A. Sifaleras, *On the initialization methods of an exterior point algorithm for the assignment problem*, International Journal of Computer Mathematics **87** (2010), pp. 1831–1846.
- [7] Paparrizos, K., N. Samaras and A. Sifaleras, *A new exterior simplex type algorithm for the minimum cost network flow problem*, Computers & Operations Research **36** (2009), pp. 1176–1190.
- [8] Sifaleras, A., *Minimum cost network flows: Problems, algorithms, and software*, Yugoslav Journal of Operations Research (2013), (to be published).
- [9] Sleator, D. D. and R. E. Tarjan, *A data structure for dynamic trees*, Journal of Computer and System Sciences **26** (1983), pp. 362–391.

A Study of the Connectivity Over Time Behavior of On-Demand Ad Hoc Path Selection Schemes

David Raz¹ Shira Levy²

*Management of Technology Faculty
HIT - Holon Institute of Technology
Holon, Israel*

Abstract

Selecting the most appropriate path between each pair of nodes in an ad hoc network is an issue with major impact on network performance. Many schemes were proposed and compared in the literature, using various criteria. However, the connectivity over time behavior of these schemes, which is important to some practical applications, was not well studied, especially with regards to the terrain type. In this work we use simulation to study this aspect of network performance. We demonstrate that a different connectivity requirement and a different time horizon may dictate a different scheme to use. We also demonstrate that path selection schemes are not equally sensitive to the terrain.

Keywords: Ad hoc networks, On-demand routing, Routing protocols

¹ Email: davidra@hit.ac.il

² Email: shiralevy1303@gmail.com

1 Introduction

In mobile ad hoc networks all aspects of routing, including route discovery and message delivery, must generally be performed by the nodes themselves, which usually operate as both hosts and routers. On-demand routing protocols are usually preferred, as they do not require maintaining updated routing tables ([7]). A major part of any routing protocol is the path selection scheme used to select the most appropriate path between any two peers. To this end many schemes were proposed, starting with, perhaps the simplest path selection scheme, Min Hop Count (MHC), where the route with the minimal number of hops is selected. A categorization into stability-based routing, power-aware routing and load-balanced routing was proposed by [7]. In that work Flow-Oriented Routing protocol (FORP), Min-Max Battery Cost Routing (MMBCR) and Load Balancing Routing (LBR) were used as representatives of stability-based routing, power-aware routing and load-balancing routing respectively. Other examples include Minimum Total Transmission Power Routing (MTPR, see [9]) Link Signal Strength Routing (LSSR, [4]) and more. See the respective articles for exact descriptions of the schemes.

Many works compare the performance of different schemes under various criteria. For example, [7] compares schemes for their number of route transitions, hop count, end-to-end delay per data packet, energy consumed per data packet, fairness in terms of node usage and time for first node failure. Other criteria used are delays, communication rates and overhead in route set-up and maintenance (see e.g. [6]). Similar criteria are used by other works as well. However, a criteria not often used is the connectivity over time behavior. In an application of interest, connectivity is not required to be absolute. Instead, some level of connectivity loss is acceptable. In addition, the application has a well-defined time horizon for operation. A study of the behavior of the connectivity over time of path selection schemes is therefore required.

Another aspect often overlooked is the effect of the type of terrain. The type of terrain affects path loss, and therefore the distance at which terminals can communicate. Many heuristic models for evaluating pass loss in mobile systems were proposed over the years, and the subject is still actively researched, see e.g. [8], [1] and [2]. It is important to select a model appropriate for the network and terminals used by the application. In this work we use the measurement based model proposed by [5], based on measurements of UMTS cellular phones.

In this work we use simulation to study the connectivity over time behavior of several path selection schemes. We also examine how the terrain type affects

this behavior. We demonstrate that a different connectivity requirement and a different time horizon may dictate a different scheme to use.

The structure of the work is as follows. In section 2 we describe the model used for simulation. We also describe the connectivity measure, the path loss formulas, the schemes simulated and the simulations performed. In section 3 we present the simulation results. Section 4 includes some concluding remarks and suggestions for further research.

2 Model

In each simulation, n agents are roaming in a square cell. Agents start at uniformly random locations with a full battery which is capable of transmitting four hours at 100mW. Each agent can route five simultaneous calls, in addition to taking part in one call. Transmissions are made at 100mW per connection. For example, if an agent itself communicates and also routes three additional calls, four seconds of battery time are consumed every second.

Agents travel within a 0.5 meter grid. Every second each agent can travel -0.5/0/0.5 meter in each orthogonal direction. Agents on the border of the cell cannot travel outside the cell.

Agents initiate five calls per hour in average. A call is initiated to another random agent. The duration of the call is triangularly distributed between four seconds and ten minutes, with the peak at the average (five minutes).

Power requirement is presumed to be negligible when idle. Power consumption by the routing protocol is also assumed to be negligible in comparison to the calls themselves. An additional assumption that communication routes include at most eight agents (seven hops). This is done from both a practical reason, to simplify the simulation, but also because of higher latency when the path is longer, and due to [3] showing that 95% of calls use six hops or less.

2.1 Path Loss Formulas

To evaluate whether two terminals can communicate directly we use the link budget formula:

$$(1) \quad P_{RX} = P_{TX} + G_{TX} - L_{TX} - L_{FS} - L_P + G_{RX} - L_{RX},$$

where P_{RX} is the received power (dBm), P_{TX} is transmitter output power (dBm), G_{TX} is transmitter antenna gain (dBi), L_{TX} is transmitter losses (dB), L_{FS} is free space loss, L_P is loss due to other channel effects (dB), G_{RX} is

receiver antenna gain (dBi) and L_{RX} is receiver losses (dB). In order to communicate the received power needs to be higher than the receiver sensitivity. We use $L_{FS} = 32.45 + 20 \log_{10}[\text{frequency}(MHz)] + 20 \log_{10}[\text{distance}(km)]$. As mentioned above, we used the measurement based model proposed by [5], which proposes for plain terrain

$$(2) \quad L_P = 4.62 + 20 \log_{10} \frac{4f\pi}{c} - 2.24h_t - 4.9h_r + 29.3 \log_{10} d,$$

for sub-urban terrain (buildings 12 meter or less)

$$(3) \quad L_P = 20 \log_{10} \frac{4f\pi}{c} - 2h_r + 40 \log_{10} d - 4,$$

and for urban terrain (buildings 18 meter or higher)

$$(4) \quad L_P = 20 \log_{10} \frac{4f\pi}{c} - 2h_r + 40 \log_{10} d,$$

where f is the transmission frequency, c is the speed of light, h_t is the transmitter height, h_r is the receiver height and d is the distance between agents. For heights we took 1.5 meter.

2.2 Connectivity

The connectivity of the system is defined as follows. A pair of agents are considered to be connectable if their batteries are not fully depleted and there is a path connecting them through agents whose batteries are not fully depleted. Every second the actual number of connectable pairs is calculated. The connectivity of the system is the fraction of the number of connectable pairs and $\frac{n(n-1)}{2}$, the maximum number of connectable pairs. Connectivity is therefore a unit-less number between zero and one. Note that even when all batteries are full, connectivity can drop below one because an agent or a group of agents can find themselves isolated without capability to connect to the rest of the network. Also note that the first agent to have a fully depleted battery may decrease the connectivity by as little as $\frac{n-1}{n(n-1)} = \frac{2}{n}$. This is opposed to other works where the first agent to be fully depleted may be interpreted as "loss of connectivity".

2.3 Path Selection Schemes

In order to compare the behavior of different path selection schemes we chose to compare three schemes. "Max Signal" maximizes the minimum signal received in the path, similar to LSSR. However, we do not expect to get the benefits of LSSR, which result from lower error rates, as these error rates were

not modeled by the simulation. This scheme was mainly used for comparison to the other schemes. "Min Hop" is similar to MHC, selecting the path with the minimal number of hops. "Max battery" maximizes the minimum battery level on the path, in a manner similar to MMBCR.

2.4 Simulated Environments

Each simulation environment comprises a terrain type, a cell size and a number of agents. The three path selection schemes were run for each of the environments. The terrain type is either Plain, which is denoted by "LOS" for "Line of Sight", Suburban or Urban. The cell sizes chosen were either small (200×200 meter), medium (500×500 meter) or large (1000×1000 or 1500×1500 meter). The number of agents was selected carefully so that on one hand it allows full connectivity when idle, while on the other hand it shows connectivity loss after a practical time for the relevant application, a few simulated hours at most.

Table 1
Simulated Environments

Terrain type	Cell size (m)	Number of agents
Small cell		
LOS	200×200	35
Suburban	200×200	30
Urban	200×200	20
Medium cell		
LOS	500×500	45
Suburban	500×500	50
Urban	500×500	60
Large cell		
LOS	1000×1000	50
LOS	1500×1500	60

3 Results

3.1 Comparing Schemes

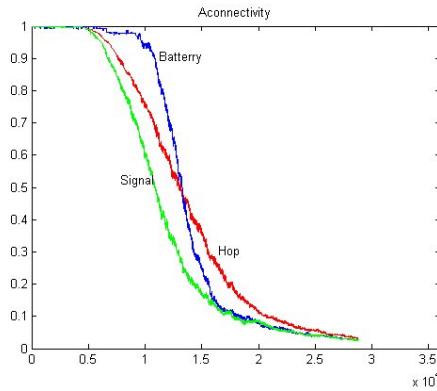


Fig. 1. Suburban, medium cell

Figure 1 depicts a typical simulation result. The simulation is of a suburban medium cell with 50 agents. Time, measured in seconds, is on the X axis, while the average connectivity is on the Y axis. The features demonstrated were common for most of the simulated environments. First, we can see that, as expected, the Signal scheme does not perform as well as the other two schemes. More interesting though is the behavior of Hop vs. Battery. As expected, maximizing battery usage has the benefit of starting to lose connectivity later. However, when connectivity starts decaying, it does so very steeply. At some point in time (after 13347 seconds in this case) Hop starts performing better than Battery. The existence of such a threshold was observed in most of the simulated environments.

There is an important practical outcome to this phenomena. To demonstrate, assume a certain application has a time horizon of 12,000 seconds, and a connectivity requirement of 70%. Clearly, only the Battery scheme can be used. However, if the time horizon is higher, say 16,000 seconds, and the connectivity requirement is lower, say 30%, only the Hop scheme can be used.

3.2 Comparing Terrains

Figure 2 depicts a comparison of the performance of the Signal scheme (2(a)) and Battery scheme (2(b)) on a medium cell with 50 agents, with different terrains. As expected, the more obstructions, the worse the connectivity. More interesting though, not all schemes are equally sensitive to the terrain.

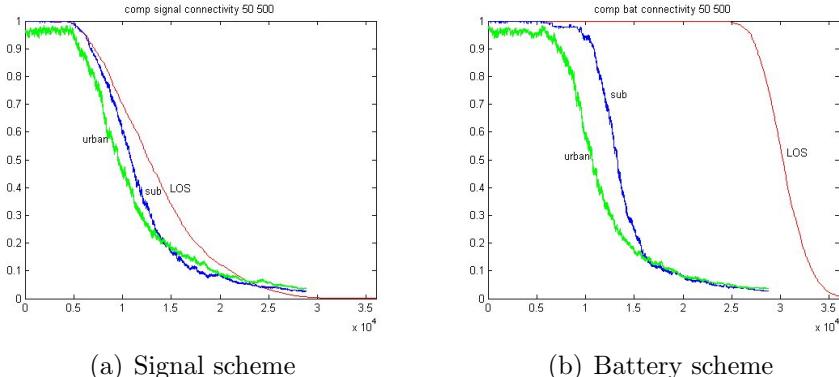


Fig. 2. Comparing terrains, medium cell

As can be seen, the Signal scheme is not very sensitive. The Battery scheme, however, is very sensitive to the terrain. The Hop scheme was found to be equally sensitive.

Another interesting phenomena is that the rate of decay, or the slope of the graph, is dependent on the terrain. We observed that the slope is steeper for suburban terrain. However, the difference may not be significant, and further study of this phenomena is required.

4 Conclusions

A practical application is considered, where the connectivity over time behavior is important. We studied three path selection schemes. Our study demonstrates that path selection schemes which are considered inferior, might actually be preferable in some situations. Specifically, in most cases there is a threshold in time after which minimizing the hop count of the path may be a better than maximizing battery life. The actual connectivity requirement and time horizon of the application may dictate the preference of one scheme over the other. Further study is underway trying to formulate this phenomena better.

We also demonstrate that path selection schemes are not equally sensitive to the terrain type. Specifically we show that a scheme that maximizes the signal strength is less sensitive to the terrain type than other schemes. There also seems to be a connection between the terrain type and the rate at which connectivity declines. Further research of this phenomena is required.

Our study focused on a small number of path selection schemes. Further research should be carried on other schemes. Specifically, there may be a

benefit to a scheme which uses one selection criteria initially, changing to another scheme when connectivity goes below some threshold. Determining this threshold may be difficult.

References

- [1] Alshami, M., T. Arslan, J. Thompson and A. Erdogan, *Frequency analysis of path loss models on wimax*, in: *3rd Computer Science and Electronic Engineering Conference (CEEC)*, Colchester, UK, 2011, pp. 1–6.
- [2] Chandra, A., A. Patra and C. Bose, *Effect of imperfect phase synchronization on the error rate performance of mpsk in rayleigh, rician and nakagami fading channels*, in: *India Conference (INDICON), 2010 Annual IEEE*, IEEE, 2010, pp. 1–4.
- [3] Chung, W., *Probabilistic analysis of routes on mobile ad hoc networks*, Communications Letters, IEEE **8** (2004), pp. 506–508.
- [4] Jayahkudy, U. and C. Tan, *Link signal strength based path selection scheme for ad hoc networks*, in: *Internet Technology and Secured Transactions (ICITST), 2010 International Conference for*, IEEE, 2010, pp. 1–6.
- [5] Konstantinou, K., S. Kang and C. Tzaras, *A measurement-based model for mobile-to-mobile umts links*, in: *Vehicular Technology Conference, 2007. VTC2007-Spring. IEEE 65th*, IEEE, 2007, pp. 529–533.
- [6] Lin, X., Y. Kwok and V. Lau, *A quantitative comparison of ad hoc routing protocols with and without channel adaptation*, Mobile Computing, IEEE Transactions on **4** (2005), pp. 111–128.
- [7] Meghanathan, N. and L. Milton, *A simulation based performance comparison study of stability-based routing, power-aware routing and load-balancing on-demand routing protocols for mobile ad hoc networks*, in: *Wireless On-Demand Network Systems and Services, 2009. WONS 2009. Sixth International Conference on*, IEEE, 2009, pp. 3–10.
- [8] Rahimian, A. and F. Mehran, *Rf link budget analysis in urban propagation microcell environment for mobile radio communication systems link planning*, in: *Wireless Communications and Signal Processing (WCSP), 2011 International Conference on*, IEEE, 2011, pp. 1–5.
- [9] Varaprasad, G., *Power aware with the survivable routing algorithm for mobile ad hoc networks*, Mobile Wireless Middleware, Operating Systems, and Applications (2010), pp. 265–275.

An Extension of a Balanced Flow in a Network

Wataru Kishimoto¹

*Dep. of information & computer sciences
Chiba University
Chiba, Japan*

Abstract

We say x is an element of a network N , if x is an edge or a vertex of N . We consider a flow network N in which capacities and distribution parameters are given on the elements in N . A d -balanced flow is defined as a flow such that the flow value at each element x is not more than $d(x) \cdot f$, where f is the total amount of the flow and $d(x)$ is the given distribution parameter of x . We give a max-flow min-cut theorem for this d -balanced flow. Then we show a method for evaluating the maximum value of d -balanced flow between a pair of vertices. The algorithm gives the maximum value of d -balanced flow s to t in N with at most $2^{|\mathcal{X}|}$ evaluations of maximum flow in a network, where \mathcal{X} is the element set of N .

Keywords: Balanced flow, Reliable flow, Max-flow min-cut, Graph theory.

1 Introduction

We consider a balanced flow problem of single commodity case. We say x is an element of a network N , if x is an edge or a vertex of N . We consider a directed flow network N in which capacities and distribution parameters are given on the elements in N . Let d be a function, which assign a positive real

¹ Email: wkishi@faculty.chiba-u.jp

number not greater than 1 to the elements in a network N . A d -balanced flow is defined as a flow such that the flow value at each element x is not more than $d(x) \cdot f$, where f is the total amount of the flow. Based on d -balanced flow, the d -capacity is defined for a mixed cut in a network. A mixed cut in a network is a set of elements removal of those separates the network. Then we give a max-flow min-cut theorem for this d -balanced flow for the single commodity case in a directed network. Then we show a method for evaluating the maximum d -balanced flow between a pair of vertices. The algorithm gives the maximum value of d -balanced flow s to t in N , with at most $2^{|X|}$ evaluations of maximum flow in a network, where X is the element set of N . We can apply all the results to undirected networks.

The problem to construct a balanced flow corresponds to, for example, the design of a reliable data communication channel for reliable data flow in a network. A multiroute flow[1,7,8] corresponds to a multiroute communication channel. A δ -reliable flow is an extension of a multiroute flow[9,10] and a (δ, η) -balanced flow is a further extension of a δ -reliable flow[11].

This paper is organized as follows. Section 2 is for preliminaries. In section 3, we define the d -balanced flow and gives the max-flow min-cut theorem. Section 4 shows a method for obtaining the maximum (δ, η) -balanced flow between a pair of vertices.

2 Preliminaries

2.1 Flow Network

Let $G = (V, E)$ be a graph with the vertex set V and the edge set E . Then $N = (G, c)$ is a network, where we say c is a capacity function such that $c(x)$ is non-negative real value and indicates the capacity of element x (i.e. $x \in V \cup E$) in G . We set $c_i(x) = 0$ if $x \notin V \cup E$. If G is directed, N is directed, otherwise N is undirected. Unless stated otherwise, networks we consider are directed.

Definition 2.1 A network is λ *uniform* or uniform with value λ if each edge and each vertex has capacity λ .

Let V_1 be a nonempty subset of vertex-set V of network N , and V_2 be the complement of V_1 . In a directed network N , a cut (V_1, V_2) of N is the set of edges, each of which is directed away from a vertex of V_1 and directed toward a vertex of V_2 . The set of edges directed away from a vertex v is called the out-incidence cut of v , and the set of edges directed into v is the

in-incidence cut of v . Then, $I_o(v)$ stands for the out-incidence cut of v and $I_i(v)$ the in-incidence cut of v .

We denote by \bar{U} the complement of a subset U of V . When $s \in V_1$ and $t \in \bar{V}_1$, we say (V_1, \bar{V}_1) is a $s-t$ cut. The sum of edge capacities of an edge set S is called the capacity of the edge set, and is indicated by $c(S)$. The capacity of a cut (V_1, \bar{V}_1) is simply represented by $c(V_1, \bar{V}_1)$ instead of $c((V_1, \bar{V}_1))$.

The capacity of a vertex also restricts the amount of flows passing through the vertex. Clearly, no vertex, even s nor t can convey a flow with the value greater than $c(I_i(v))$ and $c(I_o(v))$. If the capacity of each vertex v denoted as $c(v)$ satisfies

$$c(v) \geq \max\{c(I_i(v)), c(I_o(v))\},$$

we say c is affluent at v . Moreover, a network $N = (G, c(\cdot))$ is smooth if c is affluent at each vertex in G . In this paper, if c is affluent at v we set

$$(1) \quad c(v) = \max\{c(I_i(v)), c(I_o(v))\}.$$

If N is smooth, we set equation (1) holds at each vertex of N . In a smooth network we can define the value of an ordinary flow between any pair of vertices without care of capacities of vertices in the network.

Definition 2.2 In a directed network $N = (G, c(\cdot))$, let V_1 and V_2 be two nonempty proper subsets of the vertex set V of G such that $V_1 \cap V_2 = \emptyset$. The mixed cut is the set of elements that consists of the edges each of which is directed away from a vertex in V_1 toward a vertex in V_2 and the vertices in $V - V_1 - V_2$. The symbol (V_1, V_2) is used for this mixed cut. The capacity $c(V_1, V_2)$ is defined as

$$c(V_1, V_2) = \sum_{x \in (V_1, V_2)} c(x).$$

When $s \in V_1$ and $t \in V_2$, we say (V_1, V_2) is an $s-t$ mixed cut. If $V_1 \cup V_2 = V$, i.e. $V_2 = \bar{V}_1$, the mixed cut (V_1, V_2) coincides with the ordinary cut (V_1, \bar{V}_1) .

2.2 Definition of Flow

A flow from a vertex s to a vertex t is usually defined as a function f from the edge set of a network to nonnegative real numbers[2];however, the following definitions are used here. Henceforth, N is assumed to be a directed network.

Definition 2.3 A path-flow from a vertex s to a vertex t with the value λ is

a uniform network $P = (G_p, c_p)$ of value λ such that G_p is a directed path from s to t . We also say that P is an s - t path-flow.

Definition 2.4 A flow from a vertex s to a vertex t is defined as a network $F = (G_f, c_f)$ which is the sum of s - t path-flows P_1, P_2, \dots, P_n . The value of F is defined as the sum of the values of these path-flows. We also say that F is an s - t flow.

From the max-flow min-cut theorem, the maximum flow value from s to t is equal to the minimum capacity of s - t cuts in N [2].

2.3 Cut-Flow

Definition 2.5 [8] Let (V_1, \bar{V}_1) be a cut of a network N . A *cut-flow* of (V_1, \bar{V}_1) is a subnetwork $F_c = (G_f, c_f)$ of N such that G_f consists of the edges in (V_1, \bar{V}_1) and the end vertices of these edges. The value of F_c is defined by $c_f(V_1, \bar{V}_1)$.

Since $F_c = (G_f, c_f)$ is a subnetwork of N , $c_f(e) \leq c(e)$ for each edge e in G_f . The maximum value of cut-flow of (V_1, \bar{V}_1) is equal to $c(V_1, \bar{V}_1)$. A cut-flow of (V_1, \bar{V}_1) can be considered to be a flow between a pair of vertex sets, instead of between a single pair of vertices.

Definition 2.6 [8] Let (V_1, V_2) be a mixed cut of a network N , which consist of a vertex set V_c and an edge set E_c . Let V'_c be the set of the end vertices of the edges in E_c . A *cut-flow* of (V_1, V_2) in N is a subnetwork $F_c = (G_c, c_c)$ of N where G_c is the graph with the edges set E_c and the vertex set $V_c \cup V'_c$. The value of F_c is defined by $c_c(V_1, V_2)$.

It can be easily seen that the maximum value of a cut-flow of a mixed cut (V_1, V_2) is equal to $c(V_1, V_2)$.

3 Balanced Flow with Distribution Parameter

3.1 Balanced Flow with Distribution Parameter

Let $N = (G, c)$ is a network in which G is a graph with vertex set $V(G)$ and edge set $E(G)$. If $x \in (V(G) \cup E(G))$, we say x is an element in N or an element in G . Let $\mathcal{X} = V \cup E$ be the set of elements of G . An element x is a vertex or an edge in N and also a vertex or an edge in G . We define a distribution parameter as a function d from an element of a network to a

positive real not greater than 1, that is,

$$0 < d(x) \leq 1 \quad \text{for } x \in N.$$

In this paper we consider a network $N = (G, c)$ with a given distribution function d , where G is a graph with an edge set $E(G)$ and a vertex set $V(G)$, c is a capacity function.

Definition 3.1 A d -balanced flow is defined as a flow $F = (G_F, c_F)$ from a vertex s to a vertex t , such that

$$c_F(x) \leq d(x)f,$$

for each element x in F except s and t where f is the value of F .

A d -balanced flow is an extension of a (δ, η) -balanced flow [11] by using a distribution parameter. A single failure of an element x (that is an edge-failure or a vertex-failure) causes, at most, a fraction $d(x)$ of a d -balanced flow to fail. Therefore, at least $(1 - d(x)) \cdot$ (value of the flow) survives a failure of an element x . Thus d -balanced flow has a distinct reliability against a failure of each element.

3.2 d -Capacity of a Mixed Cut

Definition 3.2 A d -balanced cut-flow of a mixed cut (V_1, V_2) in $N = (G, c)$ is a cut-flow $F_c = (G_c, c_c)$ of (V_1, V_2) in N such that

$$c_c(x) \leq d(x)f_c,$$

for each element x in F_c where f_c is the value of F_c .

Definition 3.3 The d -capacity of a mixed cut is the maximum value of d -balanced cut-flow of the mixed cut.

Since the d -capacity is the value of a cut-flow, then

$$c_{(d)}(V_1, V_2) \leq c(V_1, V_2)$$

where $c_{(d)}(V_1, V_2)$ stands for the d -capacity of (V_1, V_2) .

Let F be a d -balanced flow from s to t in N with the value f . Suppose that a subnetwork F_1 consists of the edges and the vertices of an s - t mixed cut (V_1, V_2) in F and that its capacities of edges and vertices are the same values as in F . Then F_1 is a cut-flow of (V_1, V_2) with the value f_1 ($\geq f$). Since the capacity of each element x is less than $d(x)f$ ($\leq d(x)f_1$), F_1 is a d -balanced cut-flow of (V_1, V_2) . Consequently, the value of a d -balanced flow from s to t is not greater than the d -capacity of (V_1, V_2) .

3.3 Evaluation of d -Capacity of a Mixed Cut

Let the elements of a mixed cut (V_1, V_2) in $N = (G, c)$ be x_1, \dots, x_p such that

$$c(V_1, V_2) = \sum_{k=1}^p c(x_k);$$

$$c(x_i)/d(x_i) \geq c(x_{i+1})/d(x_{i+1}), \quad \text{for } 1 \leq i \leq p-1.$$

We give Algorithm 1, which evaluates the d -capacity of a mixed cut. In the following, the sequence $\theta_i^{(d)}$, for $i \in \mathcal{I}$, is the θ sequence of (V_1, V_2) defined as follows:

$$\begin{aligned} \theta_i^{(d)} &= \frac{1}{1 - \sum_{k=1}^{i-1} d(x_k)} \sum_{k=i}^p c(x_k) \\ &= \frac{1}{1 - \sum_{k=1}^{i-1} d(x_k)} \left\{ c(V_1, V_2) - \sum_{k=1}^{i-1} c(x_k) \right\}. \end{aligned}$$

Algorithm 1 Step 1: Calculate $\theta_i^{(d)}$ for $1 \leq i \leq \alpha$ until

$$(2) \quad c(x_i) \leq d(x_i)\theta_i^{(d)} \quad (= \frac{d(x_i)}{1 - \sum_{k=1}^{i-1} d(x_k)} \sum_{k=i}^q c(x_k)).$$

Note that if $s > t$, we define $\sum_{k=s}^t d(x_k) = 0$.

Step 2: Set d -index τ as the minimum of i ($1 \leq i \leq \alpha$) that satisfies the inequality (2).

Step 3: We say $\omega = \sum_{k=1}^{\tau-1} d(x_k)$ is the d -value of (V_1, V_2) .

Step 4: Output $\theta_\tau^{(d)}$ as $c_d(V_1, \bar{V}_1)$, the d -balanced capacity of (V_1, V_2) .

(end of algorithm)

Note that the algorithm calculate $\theta_i^{(d)}$ at most $\alpha + 1$ ($\leq p + 1$) times in step 1.

Theorem 3.4 The output $\theta_\tau^{(d)}$ of algorithm 1 is the d -capacity of (V_1, V_2) .

Theorem 3.5 In a flow network N the maximum value of d -balanced flow from s to t is the minimum value of d -capacities of s - t mixed cuts.

4 A Method for Obtaining the Maximum d -balanced Flow

On the basis of the algorithm for the maximum δ -reliable flow [10] and that of the maximum (δ, η) -reliable flow [11], we show a method for evaluating

the maximum value of d -balanced flow between a specified pair of vertices. The method consists of calculations of the maximum value of ordinary flows between the pair of vertices. The method iterates at most $2^{|\mathcal{X}|}$ times of the calculations of the maximum value of ordinary flows.

Before we show Algorithm 2, which evaluates the maximum value of d -balanced flow, we give some notations needed in the algorithm. Let $N = (G, c)$ be a network with a given distribution function d . Let s and t be a pair of distinct vertices in N . The algorithm will evaluate the maximum value of d -balanced flow from s to t . Let \mathcal{X} be the set of the elements of N . Let

$$\mathcal{D} = \left\{ \sum_{x \in X} d(x) < 1 \mid X \subset \mathcal{X} \right\}.$$

Then we denote

$$\mathcal{D} = \{\omega_0 (= 0), \omega_1, \omega_2, \dots, \omega_\beta\}$$

such that $\omega_0 (= 0) < \omega_1 < \omega_2 < \dots < \omega_\beta$. Note that $|\mathcal{D}| = \beta + 1 \leq 2^{|V|+|E|}$. For any $k \geq 1$, and a given network $N = (G, c)$, we denote $N_k = (G, c_k)$ as the network in which capacity function c_k is defined for each element x in G except source s and sink t as follows.

$$c_k(x) = \min\{c(e), d(x)\xi_k\} \quad \text{for each vertex } x \text{ in } G \text{ except } s \text{ and } t,$$

where ξ_k is defined as equation (3) in the following algorithm 2.

Algorithm 2 *Step 1: Let $N_0 = N$.*

Step 2: Evaluate the maximum value μ_0 of the ordinary flows from s to t in N_0 .

Step 3: Set $\xi_1 = \mu_0$, and $k = 1$. Then, repeat the following procedure D until the procedure stops.

Procedure D:

Step D-1: Calculate the maximum value μ_k of the ordinary flows from s to t in N_k .

Step D-2: If $\mu_k = \xi_k$, output ξ_k as the maximum value of d -balanced flows from s to t and stop.

Otherwise, set

$$(3) \quad \xi_{k+1} = \frac{\mu_k - \omega_k \xi_k}{1 - \omega_k},$$

Step D-3: Set $k = k + 1$ and go to step D-1. (end of procedure D) (end of algorithm)

Since the correctness of the algorithm is complicated, we omit it. The algorithm is not efficient. We need remarkable improvement of the algorithm

for future work.

References

- [1] Aggarwal, Charu C., and J. B. Orlin, “On multiroute maximum flows in networks.”, Networks, 39 (2002), pp.43–52.
- [2] Ahuja,R. K., T. L. Magnanti, and J. B. Orlin, “Network Flows: Theory, Algorithms, and Applications”, Prentice Hall, 1993.
- [3] Bagchi,A., A. Chaudhary, P. Kolman, and J. Sgall, “A simple combinatorial proof for the duality of multiroute flows and cuts”, Technical Report 2004-662, Charles University, Prague, 2004.
- [4] Du, D., “Multiroute Flow Problem”, Ph.D. thesis, The University of Texas at Dallas, 2003.
- [5] Du, D., and R. Chandrasekaran, “The multiroute maximum flow problem revisited”, Networks, 47(2006), pp.81–92.
- [6] Egawa, Y., A. Kaneko and M. Matsumoto, “A mixed version of Menger’s theorem” Combinatorica, 11 (1991), pp.71–74.
- [7] Kishimoto, W., and M. Takeuchi, “On M-route flows in a network ”, Proceedings of Singapore ICCS/ISITA ’92. vol.3 (1992), pp.1386 – 1390.
- [8] Kishimoto, W., and M. Takeuchi, “On m -Route Flows in a Network”, IEICE Trans., J76-A (1993), pp.1185–1200 (in Japanese).
- [9] Kishimoto, W., “Reliable Flow with Failures in a Network”, IEEE Trans. Reliability, 46 (1997), pp.308–315.
- [10] Kishimoto, W., and M. Takeuchi, “A Method for Obtaining the Maximum δ -Reliable Flow in a Network”, IEICE Trans. Fundamentals, E81-A, 5, pp.776–783(1998-05).
- [11] Kishimoto, W., “A Method for Obtaining the Maximum (δ, η) -balanced Flow in a Network”, Network Optimization - 5th International Conference, INOC 2011, Hamburg, Germany, June 13-16, 2011, Lecture Notes in Computer Science, Vol. 6701, Springer- Verlag, pp. 230-242, June 2011.

Dantzig-Wolfe Reformulation for the Network Pricing Problem with Connected Toll Arcs

Bernard Fortz² Martine Labb  ³ Alessia Violin⁴

*D  partement d'Informatique
Universit   Libre de Bruxelles
Brussels, Belgium*

Abstract

This paper considers a pricing problem on a network with connected toll arcs and proposes a Dantzig-Wolfe reformulation for it. The model is solved with column generation and the gap between the optimal integer value and the linear relaxation optimal value is shown to be at least as good as the one from the mixed-integer formulation proposed in the literature. Numerical results on different sets of instances are reported, showing that in many cases the proposed model performs strictly better.

Keywords: Combinatorial Optimization, Bilevel Programming, Pricing, Networks, Dantzig-Wolfe.

¹ This research has been funded by the Interuniversity Attraction Poles Programme initiated by the Belgian Science Policy Office. The second author acknowledges support from the “Ministerio de Ciencia e Innovaci  n” of Spain through the research project MTM2009-14039-C06. The third author acknowledges support from the Belgian national scientific funding agency “Fonds pour la Formation   la Recherche dans l’Industrie et dans l’Agriculture” (FRIA), of which she is a research fellow.

² Email: bfortz@ulb.ac.be

³ Email: m.labbe@ulb.ac.be

⁴ Email: a.violin@ulb.ac.be

1 Introduction

Consider a highway network owned by a company, which imposes tolls on its arcs in order to maximize its revenue. Users travel on the network and seek their minimum cost path. A bilevel programming framework can be used to model this problem, where the leader or first level is the company and followers or second level are network users, also called commodities. The bilevel pricing problem for a multicommodity network was first introduced by [7] and proved to be strongly \mathcal{NP} -hard [9]. Bouhtou et al [2] proposed a reformulation that reduces the practical size of the original network. Further studies have been developed in the literature and references can be found in [11] and [8].

In a highway network toll arcs are connected such that they constitute a single path. If we make the assumption that users who have left the highway do not re-enter it, tolls are imposed on paths uniquely determined by their entry and exit nodes. In consequence, for n nodes in the highway, the number of total paths will be n^2 . This Highway Problem (HP) has been proved to be strongly \mathcal{NP} -hard [5]. Furthermore, Heilporn et al [6] proposed a mixed-integer formulation, and developed an efficient branch-and-cut framework.

Moreover, as shown by [4], the product pricing problem of a company can be efficiently modeled as an (HP). This problem considers a company producing and pricing a set of products (singularly or in bundles) and aiming to maximize its revenue, and customers aiming to maximize their total utility or minimize their costs when buying (see [10]).

The aim of our paper is to present a Dantzig-Wolfe reformulation for (HP). We solve it with column generation and show that the gap between the optimal integer value and the linear relaxation optimal value is at least as good as the one provided by [5] and [6], and strictly better in many cases.

In Section 2 we present three mathematical formulations that were proposed in the literature for this problem. Section 3 is devoted to a Dantzig-Wolfe reformulation for this problem, starting from the non linear single level formulation. Section 4 theoretically compares the linear relaxation of our Dantzig-Wolfe reformulation with a mixed-integer linear formulation. Numerical results are presented in Section 5 and conclusions in Section 6.

2 Mathematical formulation

Let \mathcal{A} be the set of paths a of the highway, corresponding to pairs of entry and exit nodes, and \mathcal{K} the set of commodities k , corresponding to pairs of origin and destination nodes. We denote as $\mathcal{A}_k \subseteq \mathcal{A}$ the set of paths that can

be taken by commodity k . Each highway path has a fixed cost of c_a^k , which includes all the fixed costs commodity k incurs when using the path a : the cost for reaching the entry node from its origin, the cost for using the path and the cost for reaching the destination from the exit node. A toll free path between the origin and destination nodes of each commodity exists and its cost is denoted by c_{od}^k . Each commodity k has a demand of η^k . Variable t_a , $a \in \mathcal{A}$, represents the toll imposed by the leader on path a . Further, x_a^k , $a \in \mathcal{A}$ and $k \in \mathcal{K}$, is equal to 1 if commodity k uses toll path a and 0 otherwise; and x_{od}^k , $k \in \mathcal{K}$, is equal to 1 if commodity k uses its toll free path and 0 otherwise. Using this notation, (HP) can be formulated as a bilevel program as follows:

(HPBP)

$$\max_{t \geq 0} \sum_{k \in \mathcal{K}} \eta^k \sum_{a \in \mathcal{A}_k} t_a x_a^k, \quad (1a)$$

$$s.t. \quad (x, y) \in \arg \min_{x, y} \sum_{k \in \mathcal{K}} \left(\sum_{a \in \mathcal{A}_k} (c_a + t_a) x_a^k + c_{od}^k x_{od}^k \right), \quad (1b)$$

$$s.t. \quad \sum_{a \in \mathcal{A}_k} x_a^k + x_{od}^k = 1 \quad \forall k \in \mathcal{K}, \quad (1c)$$

$$x_a^k, x_{od}^k \in \{0, 1\} \quad \forall k \in \mathcal{K}, \forall a \in \mathcal{A}_k, \quad (1d)$$

where constraints (1c) allow one path choice for each commodity.

This problem can be reformulated as a single level non-linear optimization problem. First, the follower optimization problem can be separated for each commodity, and it can be replaced by constraints stating explicitly that the used path is the shortest one (2b). Moreover, the variables x_{od}^k , associated to the toll free paths, can be eliminated using constraint (1c). We introduce a new set of variables $p_a^k = x_a^k t_a$, having only one non linear constraint (2e) in the model. The non linear (HP) can be written as follows [5]:

(HPNL)

$$\max_{t, x, p} \sum_{a \in \mathcal{A}_k} \sum_{k \in \mathcal{K}} \eta^k p_a^k, \quad (2a)$$

$$s.t. \quad \sum_{a \in \mathcal{A}_k} [(c_a^k - c_{od}^k)x_a^k + p_a^k] - t_b \leq c_b^k - c_{od}^k \quad \forall k \in \mathcal{K}, \forall b \in \mathcal{A}_k, \quad (2b)$$

$$\sum_{a \in \mathcal{A}_k} [(c_a^k - c_{od}^k)x_a^k + p_a^k] \leq 0 \quad \forall k \in \mathcal{K}, \quad (2c)$$

$$\sum_{a \in \mathcal{A}_k} x_a^k \leq 1 \quad \forall k \in \mathcal{K}, \quad (2d)$$

$$p_a^k = t_a x_a^k \quad \forall k \in \mathcal{K}, \forall a \in \mathcal{A}_k, \quad (2e)$$

$$x_a^k \in \{0, 1\} \quad \forall k \in \mathcal{K}, \forall a \in \mathcal{A}_k, \quad (2f)$$

$$t_a \geq 0 \quad \forall a \in \mathcal{A}. \quad (2g)$$

In [5], Heilporn et al proposed a linearization of this model eliminating the non linear constraint (2e) and introducing the following set of constraints, $\forall k \in \mathcal{K}$ and $\forall a \in \mathcal{A}_k$:

$$p_a^k \leq M_a^k x_a^k \quad (3a)$$

$$t_a - p_a^k \leq N_a(1 - x_a^k) \quad (3b)$$

$$p_a^k \leq t_a \quad (3c)$$

We denote this linear mixed-integer formulation as (HPL). Constants M_a^k represent upper bounds on p_a^k variables, i.e. the highest value that commodity k is willing to pay for path a . Therefore, a valid value is $M_a^k = \max\{0, c_{od}^k - c_a^k\}$, $\forall k \in \mathcal{K}$ and $\forall a \in \mathcal{A}_k$. Constants N_a represent upper bounds on the cost of a path for all commodities, and a valid value is $N_a = \max_{k \in \mathcal{K}: a \in \mathcal{A}^k} M_a^k$, $\forall a \in \mathcal{A}$.

3 Dantzig-Wolfe reformulation

We propose a Dantzig-Wolfe reformulation for (HPNL). The master problem we choose is composed by Equations (2a) to (2d).

For each path $a \in \mathcal{A}$, a feasible solution is represented by the convex combination of $(x_a^k, t_a, p_a^k)^j$ values. We associate to the j^{th} solution a variable $\lambda_a^j \geq 0$. We also impose $\sum_{j \in \mathcal{J}} \lambda_a^j = 1$ and $x_a^k = \sum_{j \in \mathcal{J}} \lambda_a^j x_a^{k,j} \in \{0, 1\}$. Therefore, we define our master problem as follows:

(MP)

$$\max_{\lambda, x} \sum_{a \in \mathcal{A}} \sum_{k \in \mathcal{K}} \sum_{j \in \mathcal{J}} \lambda_a^j \eta^k p_a^{k,j}, \quad (4a)$$

$$\text{s.t } \sum_{a \in \mathcal{A}} \sum_{j \in \mathcal{J}} [(c_a^k - c_{od}^k) \lambda_a^j x_a^{k,j} + \lambda_a^j p_a^{k,j}] - \sum_{j \in \mathcal{J}} \lambda_b^j t_b^j \leq c_b^k - c_{od}^k \quad \forall k \in \mathcal{K}, \forall b \in \mathcal{A}, \quad (4b)$$

$$\sum_{a \in \mathcal{A}} \sum_{j \in \mathcal{J}} [(c_a^k - c_{od}^k) \lambda_a^j x_a^{k,j} + \lambda_a^j p_a^{k,j}] \leq 0 \quad \forall k \in \mathcal{K}, \quad (4c)$$

$$\sum_{a \in \mathcal{A}} \sum_{j \in \mathcal{J}} \lambda_a^j x_a^{k,j} \leq 1 \quad \forall k \in \mathcal{K}, \quad (4d)$$

$$\sum_{j \in \mathcal{J}} \lambda_a^j x_a^{k,j} - x_a^k = 0 \quad \forall a \in \mathcal{A}, \forall k \in \mathcal{K}, \quad (4e)$$

$$\sum_{j \in \mathcal{J}} \lambda_a^j = 1 \quad \forall a \in \mathcal{A}, \quad (4f)$$

$$0 \leq \lambda_a^j \leq 1 \quad \forall j \in \mathcal{J}, \forall a \in \mathcal{A}, \quad (4g)$$

$$x_a^k \in \{0, 1\} \quad \forall a \in \mathcal{A}, \forall k \in \mathcal{K}, \quad (4h)$$

Since the number of variables λ_a^j is exponential we use column generation to solve (MP) and the pricing subproblem (SP) is now described. We define

the dual variables associated to constraints of (MP) as follows: $\delta_b^k \geq 0$ for constraints (4b), $\delta_{od}^k \geq 0$ for constraints (4c), $\gamma^k \geq 0$ for constraints (4d), β_a^k for constraints (4e) and μ_a for constraints (4f). Problem (SP) includes the constraints of the original problem (HPNL) that were not included in (MP): (2e), (2f) and (2g). To reduce the domain of variables t_a we also include the following redundant set of constraints:

$$p_a^k - M_a^k x_a^k \leq 0 \quad \forall k \in \mathcal{K}, \forall a \in \mathcal{A}_k. \quad (5)$$

Problem (SP) is separable for each path a , meaning that we can solve a smaller subproblem for each path $a \in \mathcal{A}$, and it can be formulated as follows:

(SP_{*a*})

$$\begin{aligned} \max_{t,x,p} \sum_{k \in \mathcal{K}} \eta^k p_a^k - & \left\{ \sum_{k \in \mathcal{K}} \sum_{b \in \mathcal{A}} \delta_b^k [p_a^k - M_a^k x_a^k] - \sum_{k \in \mathcal{K}} \delta_a^k t_a + \right. \\ & \left. + \sum_{k \in \mathcal{K}} \delta_{od}^k [p_a^k - M_a^k x_a^k] + \sum_{k \in \mathcal{K}} x_a^k (\gamma^k + \beta_a^k) + \mu_a \right\}, \end{aligned} \quad (6a)$$

$$\text{s.t } p_a^k - M_a^k x_a^k \leq 0 \quad \forall k \in \mathcal{K}, \quad (6b)$$

$$p_a^k = t_a x_a^k \quad \forall k \in \mathcal{K}, \quad (6c)$$

$$x_a^k \in \{0, 1\} \quad \forall k \in \mathcal{K}, \quad (6d)$$

$$0 \leq t_a \leq N_a. \quad (6e)$$

This problem is non-linear due to (6c), but it is easily solvable. If the value of x_a^k is fixed for all $k \in \mathcal{K}$, then the objective function becomes linear in t_a , and achieves its optimal value in $\{M_a^k : k \in \mathcal{K}\} \cup \{0\}$, so only $|\mathcal{K}| + 1$ values of t_a must be evaluated for solving (SP_{*a*}).

4 Comparison between the formulations

Consider the non linear model (HPNL), the linear model (HPL) and the Dantzig-Wolfe reformulation (HPDW). We define as $F(*)$ the set of feasible solutions for a problem $*$. It is evident that $F(\text{HPL}) = F(\text{HPNL})$. Furthermore we can describe $F(\text{HPL}) = \{x \in X : Ax \leq b\}$, where X is the set of points satisfying constraints (3a), (3b), (3c), (2g) and (2f), and $Ax \leq b$ corresponds to constraints (2b), (2c) and (2d). It is easy to see that $F(\text{HPDW}) = \{x \in \text{conv}(X) : Ax \leq b\}$.

Consider now the linear relaxation of (HPL) and (HPDW) and note them as (HPL-LR) and (HPDW-LR) respectively. We conclude that $F(\text{HPDW-LR}) \subseteq F(\text{HPL-LR})$ (see [3]), and therefore the linear relaxation of the Dantzig-Wolfe reformulation provides an upper bound for the optimal value of the

integer problem as least as good as (HPL-LR). The following example shows that in some cases it is strictly better.

Consider a network with one toll path and three commodities (called k_1 , k_2 , k_3) with unit demand and toll free path cost of 10. The fixed cost for using the toll path is equal to 1 for k_1 , 2 for k_2 and 5 for k_3 . When solving the subproblem of (HPDW), we analyze all candidates values of the toll, i.e. 0, $M^{k_1} = 10 - 1 = 9$, $M^{k_2} = 8$ and $M^{k_3} = 5$. Moreover, for each of them, we derive the optimal assignment of x values, and we impose $p = x \cdot t$. We obtain $x^{k_1} = x^{k_2} = x^{k_3} = 1$ and a revenue of 0 for 0, $x^{k_1} = 1$, $x^{k_2} = x^{k_3} = 0$ and a revenue of 9 for M^{k_1} , $x^{k_1} = x^{k_2} = 1$, $x^{k_3} = 0$ and a revenue of 16 for M^{k_2} , and $x^{k_1} = x^{k_2} = x^{k_3} = 1$ and a revenue of 15 for M^{k_3} . The optimal toll value is thus $t = M^{k_2} = 8$.

Consider now the (HPL-LR) model, with the toll value $t = 8$. Constraints (3a) and (3b) for k_3 are:

$$\begin{aligned} p^{k_3} &\leq M^{k_3} x^{k_3} \Rightarrow p^{k_3} \leq 5x^{k_3} \\ t - p^{k_3} &\leq N(1 - x^{k_3}) \Rightarrow 8 - p^{k_3} \leq 9(1 - x^{k_3}) \end{aligned}$$

Therefore we have:

$$9x^{k_3} - 1 \leq p^{k_3} \leq 5x^{k_3} \Rightarrow 9x^{k_3} - 1 \leq 5x^{k_3} \Rightarrow x^{k_3} \leq 0.25$$

As we are maximizing p (so also x), we take $x^{k_3} = 0.25$ and consequently $p^{k_3} \leq 5x^{k_3} = 1.25$. Without the binary constraints we loose the condition $p = x \cdot t$, in fact: $p^{k_3} = 1.25 \neq x^{k_3} \cdot t = 4$. The solution associated to $t = 8$ is therefore $x^{k_1} = x^{k_2} = 1$, $x^{k_3} = 0.25$, $p^{k_1} = p^{k_2} = 8$ and $p^{k_3} = 1.25$, for a revenue of 17.25, which is a lower bound on the optimal solution of (HPL-LR), and in fact it is the optimal solution.

Therefore for this example $F(\text{HPDW-LR}) \subset F(\text{HPL-LR})$.

5 Numerical tests

In Table 1 we report numerical results concerning the linear relaxation of (HPL) and (HPDW) formulations. We report the gap between the optimal integer value and the linear relaxation optimal value, the time to solve the linear relaxation in seconds, and the number of columns generated by (HPDW-LR). The instances have been randomly generated and we report average results on 10 instances for each type and size. Fixed costs on toll paths are randomly set between 1 and 20, and demand for commodities is randomly set between 1 and 10. Complete network means that toll free path costs are randomly set between 20 and 30 (and so are always bigger than fixed costs of toll paths), such that all commodities could use all toll paths, whilst

for a non complete network we randomly set toll free path costs between 10 and 30, meaning that some toll paths will never be interesting for some commodities. We were not able to find the optimal integer solution for some of bigger instances (solving HPL): in this case we report the gap from the best integer solution found in 5h of computational time (* represents the number of instances not solved to optimality). The tests were run on SCIP 3.0 framework [1], on an Intel XEON with 16 CPUs at 2.27 GHz and with 16.00 Gb of Ram running Linux, using Cplex 12.0 as the solver for linear programming.

Complete graph		20 commodities			56 commodities			90 commodities		
		20 a	56 a	90 a	20 a	56 a * ⁷	90 a * ¹⁰	20 a	56 a * ¹⁰	90 a * ¹⁰
(HPL-LR)	Gap	6.19%	3.71%	3.15%	12.82%	11.18%	11.51%	15.33%	16.88%	17.47%
	Time	0.048	0.282	0.767	0.184	1.514	4.592	0.448	4.849	6.084
(HPDW-LR)	Gap	4.75%	3.52%	2.98%	7.72%	9.57%	10.68%	7.94%	13.94%	15.98%
	Time	0.117	0.484	0.912	4.428	18.6	31.26	40.5	129	125
	Cols	335	655	924	1141	1764	2227	2103	3034	3749
Non complete graph		20 commodities			56 commodities			90 commodities		
		20 a	56 a	90 a	20 a	56 a * ¹⁰	90 a * ¹⁰	20 a * ⁹	56 a * ¹⁰	90 a * ¹⁰
(HPL-LR)	Gap	22.78%	20.49%	19.58%	36.37%	41.91%	40.69%	42.87%	48.67%	52.06%
	Time	0.042	0.28	0.858	0.207	1.871	6.244	0.519	3.767	10.6
(HPDW-LR)	Gap	17.64%	19.30%	19.03%	22.19%	36.33%	37.67%	23.95%	39.84%	46.52%
	Time	0.111	0.465	1.216	3.738	19.6	41.1	37	116	214
	Cols	301	625	891	974	1699	2199	1853	2926	3747

Table 1
Numerical results on linear relaxation for (HPL) and (HPDW) models

We note that the gap of (HPDW-LR) is always smaller than the gap of (HPL-LR), with a greater difference between them when the number of toll paths is smaller or the number of commodities is bigger. In particular, for instances with 90 commodities and 20 toll paths, (HPDW-LR) is able to halve the gap. Moreover, instances on non complete graph show to be more difficult.

6 Conclusion and future developments

We have developed a Dantzig-Wolfe reformulation for (HP) and shown that, for this model, the gap between the integer optimal value and the linear relaxation optimal value is equal to or smaller than the equivalent gap for the mixed-integer model present in the literature, both from a theoretical and from a practical point of view. We are currently investigating different branching strategies to develop an efficient branch-and-price framework. Moreover we

plan to extend this to a branch-and-cut-and-price framework, including the valid inequalities developed for this problem by [6], as they have been shown to be very effective (they divide the gap by three or more).

References

- [1] Tobias Achterberg. Scip: Solving constraint integer programs. *Mathematical Programming Computation*, 1(1):1–41, 2009. <http://mpc.zib.de/index.php/MPC/article/view/4>.
- [2] M. Bouhtou, S. Van Hoesel, A.F. Van der Kraaij, and J.L. Lutton. Tariff optimization in networks. *INFORMS Journal on Computing*, 19(3):458–469, 2007.
- [3] A.M. Geoffrion. Lagrangean relaxation for integer programming. *Approaches to Integer Programming*, pages 82–114, 1974.
- [4] G. Heilporn, M. Labb  , P. Marcotte, and G. Savard. A parallel between two classes of pricing problems in transportation and marketing. *Journal of Revenue and Pricing Management*, 9(1/2):110–125, 2010.
- [5] G. Heilporn, M. Labb  , P. Marcotte, and G. Savard. A polyhedral study of the network pricing problem with connected toll arcs. *Networks*, 3(55):234–246, 2010.
- [6] G. Heilporn, M. Labb  , P. Marcotte, and G. Savard. Valid inequalities and branch-and-cut for the clique pricing problem. *Discrete Optimization*, 8(3):393–410, 2011.
- [7] M. Labb  , P. Marcotte, and G. Savard. A bilevel model of taxation and its application to optimal highway pricing. *Management Science*, 44(12):1608–1622, 1998.
- [8] M. Labb   and A. Violin. Bilevel programming and price setting problems. *4OR*, pages 1–30, 2012. To appear, <http://dx.doi.org/10.1007/s10288-012-0213-0>.
- [9] S. Roch, P. Marcotte, and G. Savard. Design and analysis of an approximation algorithm for Stackelberg network pricing. *Networks*, 46(1):57–67, 2005.
- [10] R. Shioda, L. Tun  el, and T. Myklebust. Maximum utility product pricing models and algorithms based on reservation price. *Computational Optimization and Applications*, 48:157–198, 2011.
- [11] S. Van Hoesel. An overview of Stackelberg pricing in networks. *European Journal of Operational Research*, (189):1393–1402, 2008.

A planning and routing model for patient transportation in health care

Alberto Coppi ^{a,1}, Paolo Detti ^{a,2}, Jessica Raffaelli ^{a,3}

^a *Dipartimento di Ingegneria dell'Informazione e Scienze Matematiche, University of Siena, Via Roma, 56, 53100 Siena, Italy*

Abstract

In this paper, a problem concerning both the planning of health care services and the routing of vehicles, for patients transportation is addressed. An integrated approach, based on the column generation technique, is proposed to solve the planning and routing problem. Preliminary results on real data show the effectiveness of the proposed approach.

Keywords: Planning and routing, health care services, heuristic.

1 Introduction and problem description

In this paper, a problem concerning the planning of health care services and the transportation of patients, in not urgent conditions, is addressed. The addressed problem arises from a real world context, namely the health care system of Tuscany, an Italian region [1]. The transportation services are performed by non-profit organizations, by means of heterogeneous vehicles (e.g.,

¹ Email: coppi@dii.unisi.it

² Email: detti@dii.unisi.it

³ Corresponding author, Email: raffaelli@dii.unisi.it

ambulance, bus, car, etc.), located in geographically distributed depots. In this context, the transportation services include the transportation of patients for discharges, transfers and pickup from their home addresses to health care facilities for health care services, such as medical examinations, consultations, therapeutic treatments. The problem has a tactical and an operational dimension [1]. The operational dimension is to determine, on the basis of a given set of transportation requests and a timetable of the visits, a transportation plan for a set of patients from a set of origins to a set of destinations (and vice versa). The aim is of minimizing the total transportation costs, satisfying a set of constraints on the quality and the timing of the service offered. The operational plan of transportation must take into account of agreements, constraints and roles established between healthcare management and non-profit organizations, which perform the transportation services and have a certain degree of autonomy in decision-making. The tactical dimension of the problem aims to integrate the transportation planning, i.e., the routing of the vehicles, with the definition of the visits timetable, i.e., the time in which the health care service of each patient will start. In this case, the problem consists in determining a timetable of the health care services of the patients, in such a way that the transportation costs are minimized, always respecting service quality constraints. Each transportation request corresponds to a patient that must be carried on by a vehicle, from a pickup location to a delivery location, satisfying capacity, feasibility and time windows constraints. The capacity constraints are related to the number of available seats in the vehicle and to the number of seats occupied by each patient (e.g., patients may need a stretcher or a wheelchair to be transported). The feasibility constraints set the vehicle typology that can be used to serve a given patient typology, depending on the specific set up of the vehicle (for example, a patient may require to be transported by an ambulance). Time windows constraints are related to the patients, requiring that each pickup and delivery location need to be reached within a time interval, and depend on the timetable of the visits. In order to guarantee quality of service requirements, constraints on the lengths of the routes and on the travel and waiting times of each patient must be also considered.

The operational dimension of the problem we address in the paper is strictly related to a problem known in the literature as Dial-A-Ride Problem (DARP), [3,6,9,11], a generalization of the Pickup and Delivery Problem with Time Windows (PDPTW)[2,4,10]. In fact, the operational dimension of the problem can be defined as a Heterogeneous Multi-Depot Dial-A-Ride-Problem. Both the DARP and PDPTW are *NP – hard*, therefore, the de-

velopment of solution methods for these problems have mainly focused on heuristics [5]. In some cases, the column generation technique has been successfully applied to develop exact and heuristic methods for these types of problems [2,4,7,8,10,12]. The above approaches from the literature are focused in solving the operational problem (i.e., considering only the routing aspects), while in this paper, we present an integrated approach for solving the tactical and operational dimension of the problem. More precisely, we propose an efficient heuristic algorithms based on a set covering formulation of the problem for finding both the visits timetable and the routing of the vehicles. Column generation is employed both to generate a set of possible routes and to plan the timetable of the health care services. Simulation results based on real world data arising from the healthcare Italian system show the effectiveness of the proposed approach.

The paper is organized as follows. In Section 2, a mathematical formulation and a solution approach for the tactical dimension of the problem are presented. Preliminary results are reported in Section 3.

2 Mathematical model and formulation

The addressed problem is characterized by two main sets of elements: the Vehicles Types and the Transportation Requests. In the following, the main characteristics of these two elements are listed.

Vehicle Type characteristics: depot location, capacity (i.e., the number of seats in the vehicle), type (ambulance, equipped vehicle, car or bus), number of available vehicles.

Transportation Request characteristics: pickup and delivery location, number of seats occupied on a vehicle, sets of possible pickup time and delivery time windows (depending on the health care service type), health care service type (e.g., transfer, treatment, etc.).

2.1 Problem formulation and solution approach

The addressed problem can be modeled as a set covering problem, as described in the following. Let S be the set of health care service types. The service type defines both the service typology (e.g., discharge, treatment etc.) and the health care structure where the service is performed. Let n_s^t be the maximum number of health care services of type s that can be allocated to time slot t (in the health care structure associated to service s). Let Q be the set of transportation requests and let Q_s be the set of transportation requests

of service type s , with $Q_s \subseteq Q$. Let T_q be the set of time slots in which the health care service associated to the transportation request q can be performed. The transportation requests in Q are partitioned into: 1) *outbound* requests, 2) *inbound* requests and 3) *one-way* requests. We denote by Q_A the set of the outbound requests, and given $q \in Q_A$, by q_R the inbound request corresponding to q .

Let V be the set of vehicle types, where the vehicle type defines both the vehicle typology and the depot in which the vehicle is located. Let n_p be the maximum number of vehicles of type p and let Ω_p be the set of routes that can be feasibly assigned to a vehicle of type p . Moreover, let c^{kp} be the cost of the route k when assigned to vehicle type p . Given a route k , let a_{qk}^t be a coefficient equal to 1 if: (i) the route k serves the transportation request q , and (ii) the health care service (associated to the transportation request q) is assigned to time slot t . In other words if $a_{qk}^t = 1$, the patient corresponding to request q is served by route k and is delivered at her destination to perform the health care service at slot t . Let z^{kp} be a binary variable equal to 1 if the route k is assigned to a vehicle of type p , then the problem can be formulated as follows.

$$\min \sum_{p \in V} \sum_{k \in \Omega_p} c^{kp} \cdot z^{kp} \quad (1)$$

$$\sum_{p \in V} \sum_{k \in \Omega_p} \sum_{t \in T_q} a_{qk}^t \cdot z^{kp} \geq 1 \quad \forall q \in Q \quad (2)$$

$$\sum_{k \in \Omega_p} z^{kp} \leq n_p \quad \forall p \in V_p \quad (3)$$

$$\sum_{p \in V} \sum_{k \in \Omega_p} \sum_{q \in Q_s} a_{qk}^t \cdot z^{kp} \leq n_s^t \quad \forall s \in S, \quad \forall t \in \{T_q : q \in Q_s\} \quad (4)$$

$$\sum_{p \in V} \sum_{k \in \Omega_p} a_{qk}^t \cdot z^{kp} - \sum_{u \geq t} \sum_{p \in V} \sum_{k \in \Omega_p} a_{qRk}^u \cdot z^{kp} \leq 0 \quad \forall q \in Q_A, \forall t \in \{T_q : q \in Q_s\} \quad (5)$$

$$z^{kp} \in \{0, 1\} \quad \forall p \in V, \forall k \in \Omega_p \quad (6)$$

The objective function (1) accounts for the minimization of the overall selected routes. Constraints (2) impose that a request must be served by at most a route. Constraints (3) set the maximum number of routes that can be assigned to a vehicle type. Constraints (4) state that the number of health care services of type s that can be assigned to time slot t cannot be greater than n_s^t . Constraints (5) state that if an outbound request $q \in Q_A$ is assigned to time slot t then its related return request q_R must be assigned to a time slot $u \geq t$. The solution approach that we propose for problem (1)–(6) is a two-phase heuristic procedure, based on the column generation technique.

In the first phase of the heuristic, the linear relaxation of (1)–(6) is solved by a column generation approach, where the pricing problems are solved by heuristic algorithms. In the second phase, a branch&cut algorithm is employed for solving the problem obtained in the first phase. Cplex is used for solving the problem in the second phase. Observe that, since the solution (possibly, fractional) obtained in the first phase could not coincide with the optimal solution of the linear relaxation, the overall solution approach could not be able to find the optimal solution of the addressed problem.

In the first phase, a restricted master problem (RMP) is first determined, by including a set of routes, for which a feasible solution exists for the problem. Then, the new columns to add to the RMP are detected by heuristically solving a set of pricing problems, one for each vehicle type. The pricing problems consist in finding variables with negative reduced cost corresponding to the violated dual constraints, reported in (7), or prove they don't exist.

$$\sum_{q \in Q} \sum_{t \in T_q} a_{qk}^t \mu_q - \delta_p - \sum_{s \in S} \sum_{q \in Q_s} \sum_{t \in T_q} a_{qk}^t \gamma_{st} + \sum_{q \in Q_A} \sum_{t \in T_q} (-a_{qk}^t + \sum_{u \geq t} a_{qRk}^u) \tau_{qt} \leq c^{k,p} \quad (7)$$

In (7), μ_q , δ_p , γ_{st} and τ_{qt} are the dual variables related to constraints (2), (3), (4) and (5), respectively. The resulting pricing problems, one for each type of vehicle $p \in V$, can be modeled as Resourced Constrained Elementary Shortest Path Problems on a graph G , where the nodes of G correspond to patients locations, assistance structures locations and vehicle depots.

Given a vehicle of type p , let y_q^t be a binary variable equal to 1 if: (i) the transportation request q is visited by the path served by a vehicle of type p , and (ii) the health care service associated to request q is allocated at time slot t ; and 0 otherwise. Let $c^p(y)$ be cost required by a vehicle of type p in order to satisfy the transportation requests q for which $t \in T_q$ exists such that $y_q^t = 1$. Let μ_q^* , γ_{st}^* , δ_p^* and τ_{qt}^* be the optimal dual variables associated to RMP. Then the pricing problem related to a vehicle of type p can be formulated as the problem of finding a set of requests, defined by y , that can be served by the vehicle and that minimize the following function:

$$c^p(y) - \sum_{q \in Q} \sum_{t \in T_q} \mu_q^* y_q^t + \sum_{q \in Q_A} \sum_{t \in T_q} (\tau_{qt}^* y_q^t - \sum_{u \geq t} \tau_{qt}^* y_{qR}^u) + \delta_p^* + \sum_{s \in S} \sum_{q \in Q_s} \sum_{t \in T_q} \gamma_{st}^* y_q^t \quad (8)$$

In order to obtain a fast column generation procedure, we adopted a heuristic algorithm for solving the pricing problem that basically consists in a local search procedure based on best-insertion rules. The scheme of the heuristic for a vehicle of type p is roughly described in Algorithm 1. In the algorithm, let *Route* be a route iteratively constructed, starting at depot D_p and ending

at depot D_p (where vehicles of type $p \in V_p$ are located), let $|Route|$ be its cardinality, and let α be an upper bound on the length of $Route$.

Algorithm 1. Heuristic algorithm for the Pricing Problem.

Algorithm 1

Input: $p \in V$;

Output: a set Φ of routes with negative (reduced) costs;

Initialize $Route = \{D_p, D_p\}$, and $\Phi = \emptyset$;

while($|Route| < \alpha$)

If any, select the request $q \in \{Q \setminus Route\}$, such that the route $\{Route \cup q\}$ is feasible and q can be inserted in $Route$ producing a maximum decreasing of the objective function (8);

if (q exists) and (the cost of $\{Route \cup q\}$ is negative)

Set $Route = \{Route \cup q\}$, save $Route$ in Φ ;

else return Φ , STOP;

end while

3 Computational results

In this section, the algorithm proposed in Section 2, in the following denoted as *ColGen*, is tested on 80 real-world instances arising from the Health Care System of Tuscany, Ausl 11 Empoli. The performance of *ColGen* are compared with both the real cost derived from the historical data and the solution obtained by a simple two-phase heuristic, working as follows. In the first phase of the heuristic, for each vehicle of type p , a set of routes that can be feasibly assigned to a vehicle of type p are generated by a greedy procedure, based on best insertion rules. In the second phase, Cplex is used to select the best routes among those generated in the first phase. Two sets of 40 instances each, denoted as Set 1 and Set 2, have been considered that differ for the number of transportation requests that must be served. Set 1 contains instances with 40 requests and 40 vehicle types, and Set 2 contains instances with 50 requests and 40 vehicle types. Each set of instances is composed by 4 subsets (each containing 10 instances) denoted as $P(0), P(5), P(10), P(15)$. In particular, $P(0)$ contains instances in which all the visits timetables are fixed and known (on these instances the problem consists in solving only the operational dimension), while $P(i)$, with $i = 5, 10, 15$, contains instances in which the visit timetable must be determined for i transportation requests (the other transportation requests in the instance have a known and fixed visit timetable).

The Tables 1 and 2 report the results for the instances in Set 1 and Set 2. Each row in the table reports the average values on the 10 instances of the related subset. In Tables 1 and 2, “Cost” denotes the total transportation cost of the solution found by *ColGen*, “Imp. *ColGen*” is the improvement (in %) of the solution found by *ColGen* with respect to the real cost derived from the historical data, “Imp. Heu” is the improvement (in %) of the solution found by the simple two-phase heuristic with respect to the real cost, “Col” denotes the total number of columns (routes) produced at the root node by *ColGen*, and t is the computation time in seconds required by the algorithm. In the tests, the upper bound α on the maximum length of a route is set to 12 in both the algorithms.

Table 1
Results on the instances of Set 1.

	Cost	Imp. <i>ColGen</i>	Imp. Heu	Col	t (sec)
P(0)	687.73	56.22%	31.72%	449.40	1834.34
P(5)	680.44	57.35%	13.60%	500.00	1654.24
P(10)	670.55	57.46%	16.20%	590.00	2107.83
P(15)	668.18	57.65%	1.99%	1130.00	9893.57

Table 2
Results on the instances of Set 2.

	Cost	Imp. <i>ColGen</i>	Imp. Heu	Col	t (sec)
P(0)	823.50	56.42%	34.49%	370.00	676.74
P(5)	812.53	57.05%	27.66%	680.00	3535.32
P(10)	805.69	57.40%	45.06%	833.33	6057.72
P(15)	794.08	62.28%	53.38%	1130.00	5683.44

In general, the results show that the solutions provided by our algorithm allow to reduce the total transportation cost of about 58% on average with respect to the real cost. Another aspect concerns the cost improvement obtained when the number of transportation requests with a not fixed visit timetable increases (in Set 2 an improvement of about 5%).

References

- [1] Agnetis, A., G. De Pascale, P. Detti, J. Raffaelli, P. Chelli, R. Colombai, G. Marconcini, E. Porfido, and A. Coppi, *Applicazione di tecniche di operations management per minimizzare il costo di trasporto di pazienti*, MECOSAN, Italian Quart. of Health Care Management Economics and Politics, **84**, (2012).
- [2] Baldacci,R., E. Bartolini, and A. Mingozzi, *An exact algorithm for the pickup and delivery problem with time windows*, Operations Research, **59**, (2011), 414-426.
- [3] Begur, S. V., D. M. Miller, and J. R. Weaver, *An Integrated Spatial DSS for Scheduling and Routing Home-Health-Care Nurses*, Interfaces, **27**, (1997), 35–48.
- [4] Bettinelli, A., A. Ceselli, and G. Righini, *A branch-and-cut-and-price algorithm for the multi-depot heterogeneous vehicle routing problem with time windows*, Transportation Research Part C, **19**, (2011), 723-740.
- [5] Cordeau, J. F., G. Laporte, J. Y. Potvin, and M .W. P. Savelsbergh, *Transportation on demand*, Handbooks in Operations Research and Management Science, Elsevier, Amsterdam, (2006). Forthcoming
- [6] Desrosiers, J., Y. Dumas, and F. Soumis, *A dynamic programming solution of the large-scale single-vehicle dial-a-ride problem with time windows*, American Journal of Mathematical and Management Sciences, **6**, (1986), 301-325.
- [7] Dumas, Y., J. Desrosiers, and F. Soumis, *The pickup and delivery problem with time windows*, European Journal of Operational Research, **54**, (1991), 7-22.
- [8] Lu, Q., M.M. Dessouky, *An Exact Algorithm for the Multiple Vehicle Pickup and Delivery Problem*, Transportation Science, **38**, (2004), 503-514.
- [9] Melachrinoudis, E., A. B. Illhana, and H. Min, *A dial-a-ride problem for client transportation in a health-care organization*, Computers & Operations Research, **34**, (2007), 742-759.
- [10] Ropke, S., and J.F. Cordeau, *Branch and Cut and Price for the Pickup and Delivery Problem with Time Windows*, Transportation Science, **43**, (2009), 267–286.
- [11] Sophie, N. P., J. F. Cordeau, K. F. Doerner, and R. F. Hartl, *Models and algorithms for the heterogeneous dial-a-ride problem with driver-related constraints*, OR Spectrum, **34**, (2012), 593–633.
- [12] Savelsbergh, M., and M.Sol, *DRIVE: Dynamic Routing of Independent Vehicles*, Operations Research, **46**, (1998), 474–490.

Efficient MIP formulation for optimal 1-for-N diversity coding

Mateusz Żotkiewicz^{1,2}

*Institute of Telecommunications
Warsaw University of Technology
Warszawa, Poland*

Abstract

In the paper we consider a communication network that uses *diversity coding* in order to achieve reliability. Having a set of demands and a network topology we face a problem of optimal routing of the demands and backup trees, and associations between the demands and the backup trees. We present a compact mixed integer programming (MIP) formulation for the optimization problem, which proves to be more efficient than other approaches that can be found in the literature.

Keywords: diversity coding, valid inequalities, mixed integer programming

1 Introduction

The idea of *diversity coding* was presented in [4,5] as a mechanism for providing self-healing and fault-tolerant communication networks. In its general form diversity coding can be difficult in operation and management. Therefore, its simplified version, i.e., *1-for-N diversity coding*, became of interest, and will be considered in this paper. The mechanism is briefly explained below.

¹ Work sponsored by National Science Centre, Poland, under grant 2011/01/B/ST7/02967.

² Email: mzotkiew@tele.pw.edu.pl

Assume that N demands of the same size are destined to a single sink node, and they all use failure-disjoint primary paths, i.e., each pair of primary paths is failure-disjoint. The idea behind 1-for-N diversity coding is that all those demands are also assigned to a backup tree rooted at the sink node, thus the sink node also receives a backup signal which is a modulo sum of all the received primary signals. In such a situation, when a single demand suffers a failure and the backup signal remains untouched, then the sink node is still able to decode the lost signal by modulo adding all the surviving signals including the backup signal.

Summarizing, assuming possible single failures of all links, in order to successfully use the mechanism of 1-for-N diversity coding in communication networks, a returned feasible solution has to meet the following conditions.

- Each demand reserves resources for its primary path.
- Each demand is protected by a single backup tree.
- If two demands are protected by the same backup tree, then their primary paths have to be link-disjoint.
- A primary path of a demand has to be link-disjoint with the whole backup tree that protects the demand.

We also make some additional assumptions that are not directly induced by the idea of diversity coding, but are made to keep the paper consistent with the recent published work on optimization of communication networks that take advantage of diversity coding. The assumptions are listed below.

- All links are directed.
- All demands are directed and of the same size.
- Parallel demands between the same pair of nodes are allowed.
- A single failure between a pair of nodes is defined as failures of links between those nodes in both directions (if both links exist).
- The problem is uncapacitated.
- The objective is to minimize the sum of capacity costs of all links.

The problem described above can be solved by heuristic methods presented in [3], suboptimally using an algorithm presented in [1] by adapting an available solution to a binary path restoration problem [7], or optimally using a MIP approach presented in [2]. In this paper we present another exact MIP approach that proves to be more efficient than the previously published methods.

The paper is organized as follows. In Section 2 we present a notation, an

optimization model, and valid inequalities that strengthen the formulation. The formulation is numerically evaluated in Section 3. The results are followed by conclusions in Section 4.

2 Model

In this section we present a novel compact MIP formulation modeling and optimizing a communication network that uses the idea of 1-for-N diversity coding for protection. First, a notation is presented. It is followed by the MIP formulation. The section ends with valid inequalities strengthening the model.

2.1 Notation

In order to formally present the optimization problem we introduce the following notation.

Sets:

\mathcal{E} set of links; a single link is denoted by e

\mathcal{V} set of nodes; a single node is denoted by v

\mathcal{D} set of demands; a single demand is denoted by d

\mathcal{T} set of backup trees; a single backup tree is denoted by t

$\delta^+(v)$ set of links originating at node v

$\delta^-(v)$ set of links ending at node v

Parameters:

w_e weight of link e ; $w_e > 0$ for all $e \in \mathcal{E}$

\bar{e} link opposite to link e (if exists)

$s(e)$ source node of link e

$d(e)$ destination node of link e

$s(d)$ source node of demand d

$d(d)$ destination node of demand d

$r(t)$ root node of backup tree t

Variables:

x_e^t 1 if a primary path of any demand protected by backup tree t uses link e ;
0 otherwise

- s_e^t 1 if backup tree t uses link e ; 0 otherwise
- y_e^d 1 if a backup path of demand d uses link e ; 0 otherwise
- n_d^t 1 if demand d is protected by backup tree t ; 0 otherwise

2.2 Optimization model

Having the notation defined, the problem can be presented as follows.

$$\min \sum_{e \in \mathcal{E}} \sum_{t \in \mathcal{T}} w_e (s_e^t + x_e^t) \quad (1a)$$

$$\sum_{e \in \delta^+(v)} x_e^t - \sum_{e \in \delta^-(v)} x_e^t = \sum_{d \in \mathcal{D}} n_d^t \quad \forall t \in \mathcal{T}, \forall v \in \mathcal{V} : v \neq r(t)$$

$$\sum_{e \in \delta^+(v)} y_e^d - \sum_{e \in \delta^-(v)} y_e^d = \begin{cases} 1 & \text{if } s(d) = v \\ -1 & \text{if } d(d) = v \\ 0 & \text{otherwise} \end{cases} \quad \forall v \in \mathcal{V}, \forall d \in \mathcal{D} \quad (1b)$$

$$\sum_{t \in \mathcal{T}} n_d^t = 1 \quad \forall d \in \mathcal{D} \quad (1c)$$

$$s_e^t \geq n_d^t + y_e^d - 1 \quad \forall e \in \mathcal{E}, \forall t \in \mathcal{T}, \forall d \in \mathcal{D} \quad (1d)$$

$$\sum_{e \in \mathcal{E}} (s_e^t + x_e^t + s_{\bar{e}}^t + x_{\bar{e}}^t) \leq 1 \quad \forall e \in \mathcal{E}, \forall t \in \mathcal{T} \quad (1e)$$

$$n_d^t = 0 \quad \forall d \in \mathcal{D}, \forall t \in \mathcal{T} : d(d) \neq r(t) \quad (1f)$$

The objective of the presented formulation is to minimize a sum of costs of capacities that have to be reserved on all links. Constraints (1a) and (1b) enforce conservation of flows with respect to variables x_e^t and y_e^d , respectively. In the first case, flows are considered per tree, and their sources are results of associating a demand with a given backup tree. In the second case, flows related to different demands are independent of each other, and for each of them there is a set of variables y_e^d . The fact that each demand has to be protected by one backup tree is enforced by (1c). Constraint (1d) assures that if a demand uses a given backup tree, then this backup tree contains all links that are used by a backup path of the demand. Disjointness of primary paths from their backup trees is assured by (1e). The fact that primary paths of any two demands that are protected by a single backup tree are disjoint is assured by an upper bound on variable x_e^t . Finally, constraint (1f) assures that a demand can be protected only by a tree that has its root in a destination node of the demand.

2.3 Valid inequalities

The presented formulation can be strengthened by the following valid inequalities. The inequalities are not crucial from the point of view of the correctness of the formulation, but can significantly improve performance of the MIP model.

$$\sum_{e' \in \delta^+(d(e)): e' \neq \bar{e}} s_{e'}^t \geq s_e^t \quad \forall t \in \mathcal{T}, \forall e \in \mathcal{E} \setminus \delta^-(r(t)) \quad (2)$$

Inequality (2) enforces that if a backup tree enters a node, which is not its root node, then it also has to leave this node.

$$\sum_{e \in \delta^+(s(d))} s_e^t \geq n_d^t \quad \forall t \in \mathcal{T}, \forall d \in \mathcal{D} \quad (3)$$

$$\sum_{e \in \delta^-(r(t))} s_e^t \geq n_d^t \quad \forall t \in \mathcal{T}, \forall d \in \mathcal{D} \quad (4)$$

Inequality (3) assures that if a demand is protected by a backup tree, then this backup tree has to leave a source node of the demand, while inequality (4) assures that the same tree has to enter its root node. Notice that not all backup tree have to enter their root nodes. If a backup tree is not used by any demand, then it does not have to use any link at all.

$$\sum_{d \in \mathcal{D}} n_d^t \leq |\delta^-(r(t))| - 1 \quad \forall t \in \mathcal{T} \quad (5)$$

Inequality (5) limits a number of demands that can be supported by a single backup tree to a number of links entering a root of the backup tree minus one. The inequality can be directly derived from the assumptions of Section 1. If N demands are protected by a single backup tree, then we need $N + 1$ different links that enter a root of the backup tree in order to assure the needed disjointness.

$$\sum_{e \in \delta^+(v)} s_e^t \begin{cases} = 0 & \text{if } v = r(t) \\ \leq 1 & \text{if } v \neq r(t) \end{cases} \quad \forall t \in \mathcal{T}, \forall v \in \mathcal{V} \quad (6)$$

The last inequality, i.e., (6), helps to enforce that backup trees are actual trees by letting them use at most one link to leave a node. Notice that backup trees in a solution provided by (1) are actual trees even when (6) is not present. However, in case when weights of links can be equal to zero, then (6) has to

be added to (1) in order to assure that in an obtained solution backup trees are actual trees.

3 Numerical results

First we notice that the problem defined in Section 2 can be decomposed into $|\mathcal{V}|$ independent subproblems. As a result of constraint (1f), variables related to a single destination node are independent of variables related to other destination nodes. For that reason we validated the methods using test cases with one destination only, as solving the problem with N different destinations could be decomposed into solving N single destination problems (each of the same complexity).

We tested three different approaches of solving the problem: the state-of-the-art MIP approach of [2] used as a benchmark in our research (refereed to as *benchmark*), our novel MIP approach (refereed to as *novel*), and the same novel approach facilitated with the presented valid inequalities (refereed to as *strong novel*). We tested the approaches on three different topologies from SNDlib [6]. The problem was modeled using Visual Studio 2010 and solved by CPLEX 12.3 on Intel 4-core 2.4 GHz CPU running Windows 7. For each network we randomly generated 140 test cases (20 for each considered number of demands). A number of backup trees used for each test case is equal to the half of a number of demands. The obtained results can be found in Table 1.

The table consists of ten columns. In the first three columns we put basic data about used topologies, i.e., their names in SNDlib, numbers of nodes, and numbers of links, while in the fourth column a number of demands is presented. The last six columns contain actual results. In columns *time* one can find average time spent by an approach on solving a problem. Notice that for all test cases we used a 10-minute time limit, so for configurations with exactly 20 timeouts the average time will be equal to 600 s. In columns *TLE* a number of timeouts is presented.

The results prove that the presented method, when facilitated by the developed valid inequalities, outperforms the best approach to this problem presented so far. The novel approach proves to be better in the wide spectrum of different configurations and sizes of the problem. The only test cases that have been solved faster by the state-of-the-art approach were the smallest instances of the problem, but even in those cases our approach not facilitated by the valid inequalities can be considered as an equally efficient alternative.

Table 1
Performance of different MIP approaches

top.	$ \mathcal{V} $	$ \mathcal{A} $	$ \mathcal{D} $	benchmark		novel		strong novel	
				time	TLE	time	TLE	time	TLE
polska	12	36	12	6	0.3	0	0.3	0	0.8
				8	5.5	0	13.7	0	1.8
				10	41.0	0	160.3	0	9.3
				12	227.1	2	443.7	11	166.1
				14	521.6	17	537.3	17	439.8
				16	420.0	14	420.5	14	420.1
				18	480.0	16	480.4	16	480.1
				18	480.0	16	480.4	16	480.1
atlanta	15	44	12	6	5.8	0	0.5	0	0.8
				8	93.1	2	11.9	0	3.9
				10	178.4	2	164.5	2	17.5
				12	382.9	12	228.6	7	158.9
				14	450.0	15	360.6	12	363.2
				16	390.0	13	360.8	12	362.2
				18	510.0	17	480.7	16	484.5
				18	480.0	16	480.4	16	480.1
france	25	90	12	6	2.2	0	0.6	0	0.6
				8	113.1	3	23.8	0	3.4
				10	205.5	5	163.9	3	49.5
				12	340.2	9	388.1	8	312.7
				14	302.7	10	319.0	10	301.2
				16	420.6	14	421.3	14	420.2
				18	450.5	15	451.3	15	450.3
				18	480.0	16	480.4	16	480.1

4 Conclusion

In the paper we considered a communication network that uses *diversity coding* in order to achieve reliability. We presented a novel compact mixed integer programming formulation for the related problem of routing optimization in those networks. We implemented the proposed model, together with the set of developed valid inequalities, and evaluated it on a number of test case. We also implemented the state-of-the-art approach presented in [2], and we compared it with our novel approach. The results show that the proposed formulation is more efficient than other formulations that can be found in the literature. Particulary, it outperforms the mentioned state-of-the-art MIP approach to this problem.

References

- [1] Avci, S. N. and E. Ayanoglu, Coded path protection: Efficient conversion of sharing to coding, *Proceedings IEEE International Conference on Communications* (Ottawa, Canada, 2012).
- [2] Avci, S. N. and E. Ayanoglu, Design algorithms for fast network restoration via diversity coding, *Proceedings UCSD Information Theory and Applications Workshop* (San Diego, CA, USA, 2012).
- [3] Avci, S. N., X. Hu, and E. Ayanoglu, Recovery from link failures in networks with arbitrary topology via diversity coding, *Proceedings IEEE Global Communications Conference* (Houston, TX, USA, 2011).
- [4] Ayanoglu, E., I. Chih-Lin, R. D. Gitlin, and J. E. Mazo, Diversity coding: using error control for self-healing in communication networks, *Proceedings IEEE Conference on Computer Communications* (San Francisco, CA, USA, 1990), 95–104.
- [5] Ayanoglu, E., I. Chih-Lin, R. D. Gitlin, and J. E. Mazo, *Diversity coding for transparent self-healing and fault-tolerant communication networks*, IEEE Transactions on Communications **41** (1993) 1677–1686.
- [6] Orlowski, S., M. Pióro, A. Tomaszewski, and R. Wessäly, *SNDlib 1.0 — Survivable Network Design library*, Networks **55** (2010) 276–286.
- [7] Żotkiewicz, M., M. Pióro, and A. Tomaszewski, *Complexity of resilient network optimisation*, European Transactions on Telecommunications, **20** (2009) 701–709.

Edge Coloring by Total Labelings of 4-regular Circulant Graphs

Hamida Seba¹

*Université de Lyon, CNRS
Université Lyon 1, LIRIS, UMR5205, F-69622, France*

Riad Khennoufa

*Département INFO-Bourg, IUT Lyon1
Université de Lyon, F-01000, Bourg en Bresse, France*

Abstract

Edge-coloring total k -labeling of a connected graph G is an assignment f of non negative integers to the vertices and edges of G such that two adjacent edges $e = uv$ and $e' = uv'$ of G have different weights. The weight of an edge uv is defined by:

$$w(e = uv) = f(u) + f(v) + f(e).$$

In this paper, we study the chromatic number of the edge coloring by total labeling of 4-regular circulant graphs $C_n(1, k)$.

Keywords: Edge coloring, total labeling, 4-regular circulant graphs.

¹ Email: hamida.seba@univ-lyon1.fr

1 Introduction

In wireless communications, the time slot assignment problem consists to assign time slots to communication links with the constraint that if the considered links are close geographically, they must have different time slots to avoid interferences. As time slots are critical resources, we need to minimize the number of time slots assigned to links while guaranteeing an interference-free schedule. This problem was shown to be NP-complete [1]. The time slot assignment problem is generally associated with the problem of proper edge coloring in the graph that represents the network. The vertices of the graph are the network devices and the edges are communication links. Several edge colorings of graphs are proposed each devoted to some particular communication environments or constraints.

In this paper, we study the *edge coloring by total labeling* of 4-regular circulant graphs. This coloring aims to generate a proper edge coloring of a graph by a total labeling of this graph.

The notion of *edge coloring by total labeling* is a mixing of vertex coloring and edge coloring that was introduced by Brandt et al. [2].

We use standard notations and definitions in graph theory. If $G = (V, E)$ is a given graph, $V(G)$ denotes the vertex set of G and $E(G)$ denotes its edge set. We denote by $\Delta(G)$ the maximum degree of the graph G .

Definition 1.1 An *edge-coloring total k -labeling* is a mapping f from the set of $V(G) \cup E(G)$ to the color set $\mathcal{C} = \{1, 2, \dots, k\}$ such that the weights of any two adjacent edges $e = uv$ and $e' = uv'$ of G are different $w(e) \neq w(e')$ where

$$w(e = uv) = f(u) + f(v) + f(uv).$$

The smallest integer k for which there exists an edge-coloring total k -labeling is the *edge coloring by total labeling chromatic number* denoted by $\chi'_t(G)$. The complexity of computing the edge coloring by total labeling chromatic number of a given graph is still unknown.

Clearly, for any graph G of maximum degree $\Delta(G)$ we have $\chi'_t(G) < \Delta(G) + 1$ because a vertex of degree $\Delta(G)$ needs a different color from those of the $\Delta(G)$ colors assigned to its incident edges.

By the theorem of Vizing [4], a proper edge coloring with a constant labeling of vertices defines an edge coloring by total labeling where the possible weights of the edges incident to a vertex u of maximum degree $\Delta(G)$ are $\{2f(u) + 1, 2f(u) + 2, \dots, 2f(u) + k, f(u) + k + 2, f(u) + k + 3, \dots, f(u) + 2k\}$. So, to color the graph G with $\Delta(G)$ colors we must have $2k - 1 \geq \Delta(G)$. This

induces the following lower bound [2]:

$$\chi'_t(G) \geq \left\lceil \frac{\Delta(G) + 1}{2} \right\rceil.$$

An upper bound of $\chi'_t(G)$ is given by Brandt et al. [2] in the following theorem.

Theorem 1.2 ([2]) *If G is a graph of maximum degree Δ , then $\chi'_t(G) \leq \Delta$.*

In [2], Brandt et al. compute the edge coloring by total labeling chromatic number $\chi'_t(G)$ for some classes of graphs. They give an exact value for complete bipartite graphs and complete graphs K_n if $n \not\equiv 2 \pmod{4}$ and propose an upper and lower bound otherwise. They show that $\chi'_t(F) = \left\lceil \frac{\Delta(F)+1}{2} \right\rceil$ for a forest of maximum degree $\Delta(F)$. They also give an exact value for the edge coloring by total labeling of cubic graphs that can be partitioned into two parts that induce perfect matchings.

In this paper, we consider this coloring parameter for 4-regular circulant graphs.

2 Circulant Graph $C_n(1, k)$

For two positive integers $1 \leq k_1, k_2 \leq \lfloor \frac{n-1}{2} \rfloor$ the 4-regular circulant graph $G = C_n(k_1, k_2)$ has a vertex set $V(G) = \{0, 1, 2, 3, \dots, n-1\}$, where two vertices x, y are adjacent if and only if:

$$\begin{cases} x \equiv (y \pm k_1) \pmod{n} & \text{for some } x, y \text{ } 0 \leq x, y \leq n-1, \\ x \equiv (y \pm k_2) \pmod{n} & \text{for some } x, y \text{ } 0 \leq x, y \leq n-1, \end{cases}$$

We only consider connected 4-regular circulants, that is, those circulants $C_n(1, k)$ with $k \geq 6$ and $k \geq 2$ [3].

Our main result is the following theorem:

Theorem 2.1 *Let $G = C_n(1, k)$ be a circulant graph with two positive integers $n \geq 6$ and $k \geq 2$ then if (k is odd) or (k is even and $n \neq 2xk + k - 1$ and $n \neq (2x + 1)k + 1$, $x > 0$) we have:*

$$\chi'_t(C_n(1, k)) = 3.$$

Proof. First we note that with 3 labels, say 1, 2 and 3, we can have 7 weights: 3, 4, 5, 6, 7, 8 and 9. By Vizing's theorem and by using Theorem 1.2, we obtain $3 \leq \chi'_t(G) \leq 4$. So, we have to prove that $\chi'_t(G) \leq 3$. We proceed by construction.

Let $C_n(1, k)$ be a circulant graph where n is a positive integer. Consider an ordering $v_0, v_1, v_2, \dots, v_{n-1}$ of the vertices of $C_n(1, k)$, we rename the elements of $E(C_n(1, k))$ as follows: for $0 \leq i \leq n - 1$

$$\begin{cases} e_i = v_i v_{(i+1) \bmod n}, \\ e'_i = v_i v_{(i+k) \bmod n}. \end{cases}$$

An edge e_i is called an external edge and an edge e'_i is called an internal edge. We consider two cases:

Case 1: n is a multiple of k , i.e., $n = pk$ for some positive integer p . In this case, $C_n(1, k)$ contains k internal cycles isomorphic to C_p . We define a labeling function f of the circulant graph $C_n(1, k)$ for all $0 \leq i \leq n - 1$ as follows:

$$f(v_i) = \begin{cases} 1 & \text{if } \lfloor \frac{i}{k} \rfloor \text{ is even,} \\ 3 & \text{if } \lfloor \frac{i}{k} \rfloor \text{ is odd.} \end{cases}$$

Note that this labeling ensures that any two vertices connected by an internal edge have different labels, namely 1 and 3 except for p odd where we have in each internal cycle an internal edge that has both endvertices with label 1. So, to label the edge set, we consider two subcases according to the parity of p .

- **Case 1.1:** p is even.

We first label the internal edges of $C_n(1, k)$ by the two labels 2 and 3 alternatively which allows us to use two weights $w_1 = 6$ and $w_2 = 7$. Next, we label the external edges as follows: $f(e) = 1$ if the endvertices of e has the two labels 1 and 3, i.e., $w(e) = w_3 = 5$. For a sequence of $k - 1$ vertices labeled with 1, we label the edges by 1 and 2 alternatively which allows us to use two weights $w_4 = 3$ and $w_5 = 4$ and for a sequence of $k - 1$ vertices labeled with label 3, we label the edges by 2 and 3 alternatively which allows us to use the weights $w_6 = 8$ and $w_7 = 9$.

Therefore, the weights of the internal edges are different from those used for external edges, consequently with three labels we can construct an edge coloring by total labeling of the circulant graph $C_n(1, k)$ for any positive integers $n \geq 6$ and $k \geq 2$ such that $n = pk$ and p is an even positive integer.

Figure 1 (a) shows an example of this coloring with the graph $C_{24}(1, 4)$.

- **Case 1.2:** p is odd.

We label first the internal edges as the previous case by two labels 2 and 3 alternatively with the exception that $f(e') = 1$ if the two labels of its endvertices are 1. This allows us to use three weights for internal edges $w_1 = 3$, $w_2 = 6$ and $w_3 = 7$. We label the external edges as follows:

- $f(e) = 1$ if the endvertices of e has the two labels 1 and 3, i.e., $w(e) = w_4 = 5$,
- For a sequence of $2k$ vertices labeled with label 1, we label the edges with the two labels 2 and 3 alternatively. This allow us to use the two weights $w_4 = 5$ and $w_5 = 4$ alternatively such that $w(e_i) = 4$ if $w(e_{i-1}) = 5$ or $w(e_{i+1}) = 5$. These weights are different from the weights used for the neighboring internal edges $w_1 = 3$, $w_2 = 6$ and $w_3 = 7$.
- For a sequence of k vertices labeled with label 1, we label the edges by the two labels 1 and 2 alternatively, this allow us to use the two weights $w_1 = 3$ and $w_5 = 4$ which are different from the two weights used for the two neighboring internal edges $w_2 = 6$ and $w_3 = 7$.
- For a sequence of k vertices labeled with label 3, we label the edges by the two labels 2 and 3 alternatively, this allow us to use the two weights $w_6 = 8$ and $w_7 = 9$ which are different from the two weights used for the two neighboring internal edges $w_2 = 6$ and $w_3 = 7$.

Consequently with three labels we can construct an edge coloring by total labeling of the circulant graph $C_n(1, k)$ for any positive integers $n \geq 6$ and $k \geq 2$ such that $n = pk$ and p is a positive odd integer.

Figure 1 (b) shows an example of this coloring with the graph $C_{15}(1, 3)$.

Case 2: n is not a multiple of k , i.e., $n = ak + b$ for some positive integers a and b . In this case, if we use the labeling function defined for the vertices in case 1, several internal edges will have both endvertices with label 1 So, we consider two subcases:

- **Case 2.1:** (a is even and $b \neq k - 1$) or (a is odd and $b \neq 1$).

In this case, we use the labeling function used in the case 1.2 with the following modifications in the labelling of the external edges (the labeling of the vertices and the internal edges is unchanged):

- $f(e) = 1$ if the endvertices of e has the two labels 1 and 3 i.e. $w(e) = w_4 = 5$,

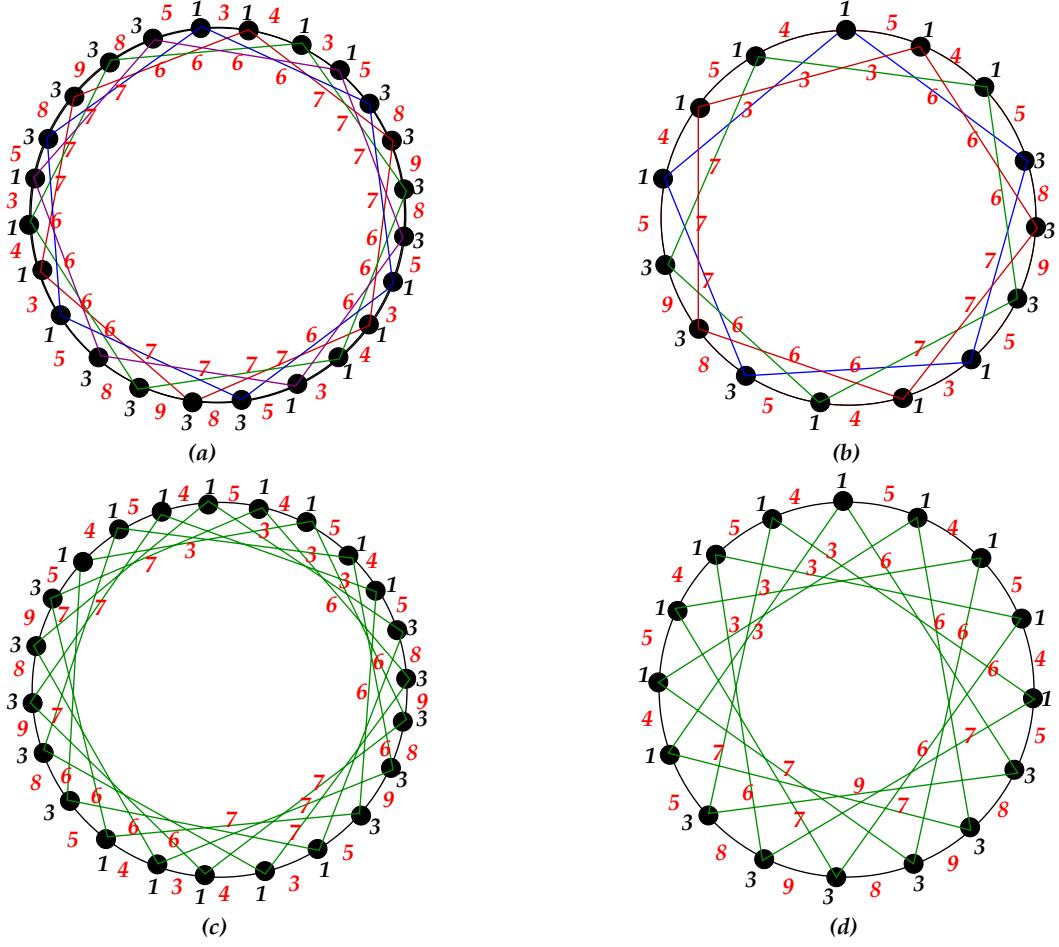


Fig. 1. (a) Coloring of $C_{24}(1, 4)$. (b) Coloring of $C_{15}(1, 3)$. (c) Coloring of $C_{23}(1, 5)$. (d) Coloring of $C_{16}(1, 5)$. The labels on the edges are the weights.

- For a sequence of $b + k$ vertices, $0 \leq b < k - 1$, such that at least one of these vertices is an endvertex of an internal edge that has the weight 3, we label their edges by the two labels 2 and 3 if the number of vertices in the sequence is even, i.e., the possible weights used are $w_4 = 5$ and $w_5 = 4$, and if the number of vertices in the sequence is odd one edge of this sequence takes the label 1 such that the weight of neighboring internal edges are different from 3, i.e., one edge has the weight $w_1 = 3$. Note that this is possible only if $b \neq k - 1$ when a is even and $b \neq 1$ when a is odd.
- For a sequence of k vertices labeled with label 1 such that none of them is an endvertex of an internal edge having the weight 3, we label the edges

by the two labels 1 and 2 alternatively, this allow us to use the two weights $w_1 = 3$ and $w_5 = 4$ which are different from the two weights used for the two neighboring internal edges $w_2 = 6$ and $w_3 = 7$.

- For a sequence of k vertices labeled with label 3, we label the edges by the two labels 2 and 3 alternatively, this allow us to use the two weights $w_6 = 8$ and $w_7 = 9$ which are different from the two weights used for the two neighboring internal edges $w_2 = 6$ and $w_3 = 7$.

Figure 1 (c) shows an example of this coloring with the graph $C_{23}(1, 5)$.

- **Case 2.2:** $((a \text{ is even and } b = k - 1) \text{ or } (a \text{ is odd and } b = 1))$ and k is odd.

We define a labeling function f of the circulant graph $C_n(1, k)$ as follows: for all $0 \leq i \leq n - 1$

$$f(v_i) = \begin{cases} 1 & \text{if } 0 \leq i \bmod (2k + 1) \leq k - 1, \\ 3 & \text{if } k \leq i \bmod (2k + 1) \leq 2k, \end{cases}$$

We first label the internal edges as follows:

- $f(e') = 1$ if the endvertices of the edge e' have the label 1, i.e., the edge has the weight $w_1 = 3$,
- $f(e') = 3$ if the endvertices of the edge e' have the label 3, i.e., the edge has the weight $w_2 = 9$
- we label the rest of the other internal edges by the two labels 2 and 3 alternatively which allows us to use the two weights $w_3 = 6$ and $w_4 = 7$.

For the external edges, we proceed as follows:

- $f(e) = 1$ if the endvertices of the edge e have the two labels 1 and 3 i.e. the edge has the weight $w_5 = 5$,
- For a sequence of vertices labeled with label 1, we label their edges by the two labels 2 and 3 if the number of vertices in the sequence is even i.e. the possible weights used are $w_6 = 4$ and $w_5 = 5$ and if the number of vertices in the sequence is odd one edge of this sequence takes the label 1 such that the weight of neighboring internal edges are different from 3, i.e., one edge has the weight $w_1 = 3$.
- For a sequence of vertices labeled with label 3, we label their edges by the two labels 2 and 3 alternatively such that an edge neighbor to the edge with weight 5 takes the weight $w_7 = 8$.

According to this labeling, we have:

- If $f(v) = 1$, then the possible weights of the internal edges are 3, 6 or 7 and the two external edges take the weights 4 or 5 and if the two internal

edges have the weights 6 or 7, then the possible weights of external edges are 4 and 5.

- If $f(v) = 3$, then the possible weights of the internal edges are 9, 6 or 7 and the two external edges take the weights 5 or 8 and if the two internal edges have the weights 6 or 7, then the possible weights of the external edges are 8 and 9.

Therefore, the weight condition of edge coloring by total labeling is verified.

Figure 1 (d) shows an example of this coloring with the graph $C_{16}(1, 5)$.

Note that:

- If a is even and $b = k - 1$ we have $n = 2xk + k - 1 = xk + x(k+1) + k - 1 - x$, $x > 0$ so the coloring is possible only if $x \neq 2k - 1$.
- If a is odd and $b = 1$ we have $n = (2x+1)k + 1 = xk + x(k+1) + k + 1 - x$, $x > 0$ so the coloring is possible only if $x \neq 2k + 1$ and $x \neq 2$.

Consequently, with three labels, we can construct an edge coloring by total labeling of the circulant graph $C_n(1, k)$ for any positive integers $n \geq 5$ and $k \geq 2$ such that k is odd or (k is even and $n \neq 2xk + k - 1$ and $k \neq (2x+1)k + 1$, $x > 0$).

□

References

- [1] Arikan, E., *Some complexity results about packet radio networks*, IEEE Trans Informat Theory **30** (1984), pp. 910–918.
- [2] Brandt, S., K. Budajová, D. Rautenbach and M. Stiebitz, *Edge coloring by total labelings*, Discrete Mathematics **310** (2010), pp. 199–205.
- [3] Hattingh, J. H., *The edge-chromatic number of circulant*, Quaestiones Math **11** (1988), pp. 371–381.
- [4] Vizing, V. G., *On an estimate of the chromatic class of a p-graph*, Diskret. Analiz **3** (1964), pp. 25–30.

Pricing techniques for self regulation in Vehicle Sharing Systems

Ariel Waserhole^{a,b,1,4} Vincent Jost^{b,2,4} Nadia Brauner^{a,3}

^a Laboratoire G-SCOP, 46, avenue Félix Viallet, 38031 Grenoble Cedex 1, France

^b LIX CNRS - École polytechnique, 91128 Palaiseau Cedex, France

Abstract

We study self regulation through pricing for Vehicle Sharing Systems (VSS). Without regulation VSS have poor performances. We want to improve the efficiency of VSS using pricing as incentive. We take as base model a Markovian formulation of a closed queuing network with finite buffer and time dependent service time. This model is unfortunately intractable for the size of instances we want to tackle. We discuss heuristics: a scenario approach, a fluid approximation, simplified stochastic models and asymptotic approximations. We compare these heuristics on toy cities.

Keywords: One Way Vehicle Sharing System, Markov Decision Process, Self Regulating System, Fluid Approximation, Product Form Queuing Networks, Linear Programming.

One way Vehicle Sharing Systems (VSS), in which users pick-up and return a vehicle in different places is a new type of transportation system that presents many advantages. However even if advertising promotes an image of flexibility and price accessibility, in reality customers might not find a vehicle at the

¹ Email: Ariel.Waserhole@g-scop.grenoble-inp.fr

² Email: vJost@lix.polytechnique.fr

³ Email: Nadia.Brauner@g-scop.grenoble-inp.fr

⁴ Chaire Microsoft-CNRS Optimisation & Sustainable Development

original station (which may be considered as an infinite price), or worse, a parking spot at destination.

Since the first bicycle VSS, problems of vehicles and parking spots availability have appeared crucial. There could be various reasons but we can highlight two important phenomena. The gravitation effect happens in bicycle VSS when users refuse to climb up hills with a vehicle even if their destination is there, *e.g.* Montmartre hill in Vélib Paris. It implies that some stations are constantly unbalanced. The tide phenomenon represents the oscillation of demand intensity along the day, *e.g.* morning and evening flows between working and residential areas.

We define the system performance as the number of trips sold (to be maximized). Our methods can be adapted to maximize the vehicle utilization time or a more general (monetary) gain. The performance of the system of bicycle VSS can be improved by vehicle relocation thus reducing the saturation of stations and hence maximizing the total time of vehicle usage. It is a specific Vehicle Routing Problem to which several research papers are devoted. Unlike bicycles, in car VSS such as Autolib' (Paris) or Car2Go (Vancouver, ULM...), vehicle relocation seems inappropriate for the resulting congestion and environmental costs. Our scope is to focus on self regulating systems through pricing incentives, avoiding physical station balancing. We want to study if an intelligent management of the incentives can increase significantly the performance of VSS.

1 Model

1.1 Protocol and pricing policies

We consider the following protocol:

- (i) A user asks to rent immediately a vehicle at a station a , to return it at a station b at a specified time;
- (ii) The system offers him a price (or refuses = infinite price);
- (iii) Either the user accepts the price, takes the vehicle in a and a parking spot is reserved in b during the rental duration (to ensure the feasibility of the return). Or the user refuses and leaves the system.

We assume that we know the stochastic demand $\lambda_{a,b}^t(p)$ of users that want to go from station a to station b at time t if the charged price is p . We study different pricing policies: Static prices depend only on the origin, the destination stations and the time slot; Dynamic prices can in addition depend

on the vehicle distribution among the stations.

1.2 Markovian model – MDP

As a base model we consider a closed queuing network that evolves as a Markovian process. Durations between two client arrivals (demand) and transportation times follow exponential distributions. The system state can be fully characterized at each instant by the vehicle distribution among the stations: number of vehicles in each station and number of vehicles in transit between each pair of stations.

This model leads straightforwardly to a Markov Decision Process (MDP) that computes the dynamic policy maximizing the average gain. In each state of the system the MDP has to define a price to take each trip. If a trip can have k different discrete prices, then deciding the prices of n trips amounts to select one action out of k^n . The action space is at first sight exponential with the number of stations. However using the framework of Action Decomposable MDP (see Waserhole *et al.* [9]), we can reduce the size of this action space to $k \times n$.

1.3 State of the art

In the literature only simple forms of this closed queuing network model have already been studied for VSS. George and Xia [3] consider a VSS with a stable demand, a fixed price and infinite station capacities. With these simplifications their model fits into a known class of queuing network (BCMP) that admits a compact product form to compute the system performance. They solve an optimal fleet sizing problem considering a cost for the maintenance of a vehicle and a gain for its rental.

Fricker and Gast [1] consider simple cities that they call homogeneous. These cities have a unique fixed station capacity \mathcal{K} , an arrival rate stable over time and uniform routing matrix: $\lambda_{a,b}^t = \frac{\lambda}{M}$ with M the number of stations; they also have a unique transportation time following an exponential distribution of mean μ^{-1} . With a mean field approximation, they obtain asymptotic results when the number of stations tends to infinity: If there is no operational regulation system, the optimal sizing is to have a fleet of $\frac{\mathcal{K}}{2} + \frac{\lambda}{\mu}$ vehicles per station which corresponds in filling half of the stations plus the average number of vehicles in transit ($\frac{\lambda}{\mu}$). Moreover they show that even with an optimal sizing each station has still a probability $\frac{2}{\mathcal{K}+1}$ to be empty or full which is pretty disappointing since these cities are perfectly balanced. Fricker

et al. [2] extend some analytic results to inhomogeneous cities modeled by clusters and verify experimentally some others.

To the best of our knowledge there is no work on time dependent demand, station capacities nor self regulation though pricing. Taking these features into account is the subject of our study.

2 Solutions techniques

The problem of a Markovian approach is the exponential number of states necessary to describe the system; it is known in the literature as the curse of dimensionality. The use of Action Decomposable MDP allows to optimize small systems, in the order of 20 vehicles and 5 stations. However for medium and big cities an exact optimization, or even an exact evaluation, seems out of reach with current methods. Nevertheless the Markovian model is easy to evaluate by simulation.

In this section, we search for exact solutions of sub-problems. These approximations help to outline the difficult aspects and to compute bounds to identify potential optimization gaps. They give heuristics for the base model that will be compared by simulation in Section 3.

2.1 Scenario based approach

When considering a stochastic problem it is classic to look at its deterministic versions. We study the scenario based approach where a finite set of trip requests is known at the beginning of the horizon. The goal is then to find the static policy that maximizes the number of trips sold.

Such optimization has two main advantages: On the one hand the a posteriori study gives an upper bound on the best static policy with on-line demands; On the second hand, solving efficiently the deterministic problem on a scenario is the first step toward robust optimization techniques. Morency *et al.* [5] show that in Montreal BSS Bixi, 68% of the trips were made by “members”, and that their frequencies of use are quite stable along the week. In this case, considering the incertitude as a set of deterministic scenarii might be a promising approach.

If all trip requests are known, the evaluation of a scenario can be computed thanks to a greedy algorithm in a time and space network with a flow constrained by a First Come First Served (FCFS) type rule. Optimizing this scenario amounts to open or close trips (in the space network) in order to maximize the underlying FCFS flow. This problem is NP-hard even for one

vehicle, infinite station capacities and a unique price for all trip requests (see Waserhole *et al.* [6]). If we relax the FCFS constraint we can use a max flow algorithm to get an upper bound on a scenario. Unfortunately this bound can be arbitrarily far from the optimal FCFS flow [6].

2.2 Fluid approximation

We study another deterministic approach, a fluid approximation of the MDP that can be seen as a plumber problem. For this model, we assume that we know the maximum demand $\Lambda_{a,b}^t$ of users that want to take a trip at time step t between stations a and b . This demand can be obtained with the minimum price, possibly negative if the system pays the user. We also assume that there exists a price to obtain any demand $\lambda_{a,b}^t \in [0, \Lambda_{a,b}^t]$; *i.e.* the demand is elastic and surjective in $[0, \Lambda_{a,b}^t]$. Thanks to this hypothesis we can avoid an explicit function linking the price and the demand when maximizing the number of trips sold or the utilization time.

The fluid model is built by replacing discrete stochastic demands by continuous deterministic ones preserving the mean values. The vehicle distribution among the station evolves then deterministically. Stations are represented by tanks connected by pipes representing the demands. Vehicles are considered as continuous fluid evolving in this network. The volume of a tank represents the capacity of a station. The length of a pipe represents the transportation time between two stations. The section of a pipe between two tanks a and b represents the demand between stations a and b ; it ranges over time from 0 to the maximum demand $\Lambda_{a,b}^t$.

In Waserhole and Jost [8], we show that maximizing the number of trips sold can be formulated as a State Constrained Separated Continuous Linear Program (SCSCLP), that can be solved efficiently; see Luo and Bertsimas [4]. For discrete prices we would obtain a nonlinear model with integer variables [8]. This fluid approximation gives a static policy (fluid heuristic) and an upper bound on the stochastic base model.

2.3 Tractable stochastic sub-problems

In Waserhole and Jost [7] a policy is said to be “conservative” if in each station and at each instant, the demand for taking and returning a vehicle are equal (in expectation). We use the property that a policy is conservative if and only if the stationary distribution on the reachable states of the induced Markov chain is uniform.

As in section 2.2 we assume that the demand is elastic and surjective. For

stable demands over time, infinite station capacities and null transportation times, we propose a linear programming approach that computes the static conservative policy maximizing the number of trips sold [7]. This formulation has a polynomial size.

To go further in the simplification, we approximate the base model with stable demands by the search of a circulation with maximum value in the demand graph. This approximation gives a static policy and an upper bound on the base model for static and also dynamic policies. We prove that it is asymptotically exact when the number of vehicles tends to infinity [7].

3 Simulation results

We compare the relative gain of the heuristics presented in Section 2 to the classic “generous” policy where the prices are set to their minimum values. We evaluate their performance by simulation on a benchmark. This benchmark consists of instances with stations spread on a regular planar grid with Manhattan distances. Instances contain $6 \times 4 = 24$ stations of capacity 10, and a fleet of $60\% \times 10 \times 24 = 144$ vehicles. One of them is an exception with the same number of vehicles and stations but infinite station capacities and null transportation times. The demand is first built homogeneous; then we introduce tide or gravitation phenomena. Figure 1 presents some simulation results.

Experimentally, the more intense the demands, the more the fluid heuristic improves the number of trips sold (compared to the generous policy) by giving priority to short trips. The conservative and maximum circulation heuristics are considering null transportation times; therefore they can’t improve homogeneous instances.

The upper bound given by the fluid model and the one given by the max flow (on a scenario) are close. However the gap between these upper bounds and the value of the fluid heuristic simulated is generally in the order of 30%. It might be due to the deterministic approximation that neglects the saturation probability of the stations close to a critical state (empty or full).

For stable demands with gravitation, null transportation times and infinite station capacities, the conservative heuristic is optimal but the fluid and the maximum circulation heuristics are giving similar results. The approximation by a fluid and even by an infinite number of vehicles seems relevant under these assumptions.

Finally, we adapt the conservative and the maximum circulation heuristics to deal with time dependent demands by optimizing independently each time

slot, in which the demand is considered stable; it doesn't provide an upper bound anymore but only a forecast value. Tests show that the fluid heuristic is the only one that optimizes correctly the tides.

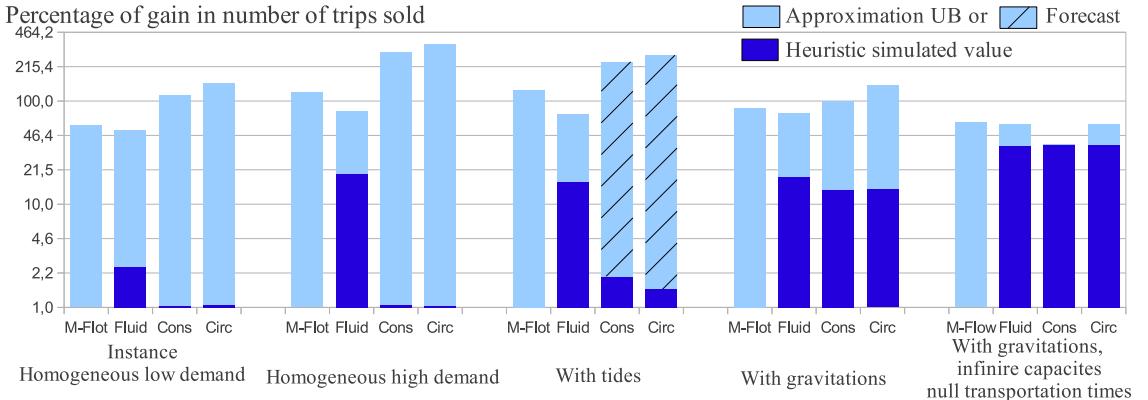


Fig. 1. Percentage of optimization gap in number of trips sold compared on a logarithmic scale to the generous policy (minimum prices) for the max flow upper bound (M-Flow), the fluid (Fluid), conservative (Cons), maximum circulation (Circ) heuristics and their upper bound or forecast value.

4 Preliminary conclusions and perspectives

We discussed some issues about regulation through pricing for VSS. We considered a Markovian model with a simple protocol. This model is intractable and there exists some approximations to tackle it in the literature giving heuristics and upper bounds. Different solution methods were compared on simple instances. The fluid heuristic gave the best results improving significantly the classic generous policy in which prices are fixed and set to their minimum. However the fluid heuristic can be complex to compute, especially for very big systems, whereas the maximum circulation heuristic provides a good trade-off between simplicity and solution value.

The upper bounds are pretty far above the simulated value of the heuristics. Static policies might not have enough leverage to reduce this gap and further investigations of dynamic policies would be interesting.

In order to fit with real VSS, several extensions should be investigated. In dense networks of stations, users are interested in several options for origin and destination stations. For regular or “long” term rental, reservations and time flexibility are issues. Locking a parking spot at destination might be too safe, hence overbooking should be considered.

Last but not least, evaluation of the demand is a hard issue. Not only is the elasticity of the demand (as a function of price) hard to estimate, but the demand itself shouldn't be assimilated to the trips that have been sold. It is a problem of censored demand: system saturation is correlated with high demands which are truncated in trips sold data.

References

- [1] C. Fricker and N. Gast. Incentives and regulations in bike-sharing systems with stations of finite capacity. *arXiv :1201.1178v1*, January 2012.
- [2] C. Fricker, N. Gast, and H. Mohamed. Mean field analysis for inhomogeneous bike sharing systems. In *23rd Intern. Meeting on Probabilistic, Combinatorial, and Asymptotic Methods for the Analysis of Algorithms (AofA'12)*, 2012.
- [3] D. K. George and C. H. Xia. Fleet-sizing and service availability for a vehicle rental system via closed queueing networks. *European Journal of Operational Research*, 211(1):198 – 207, 2011.
- [4] X. Luo and D. Bertsimas. A new algorithm for state-constrained separated continuous linear programs. *S/AM Journal on control and optimization*, pages 177–210, 1999.
- [5] C. Morency, M. Trépanier, and F. Godefroy. Insight into the Montreal bikesharing system. In *TRB-Transportation Research Board Annual Meeting, Washington, USA, Paper #11-1238, 17 pages*, January 2011.
- [6] A. Waserhole and V. Jost. Vehicle sharing system pricing regulation: Deterministic approach, complexity results. <http://hal.archives-ouvertes.fr/hal-00727040>, 2012.
- [7] A. Waserhole and V. Jost. Vehicle sharing system pricing regulation: Tractable optimisation models for queueing networks. <http://hal.archives-ouvertes.fr/hal-00751744>, 2012.
- [8] A. Waserhole, V. Jost, and N. Brauner. Vehicle sharing system pricing regulation: A fluid approximation. <http://hal.archives-ouvertes.fr/hal-00727041>, 2012.
- [9] A. Waserhole, V. Jost, and J. P. Gayon. Action decomposable MDP, a linear programming formulation for queuing network problems. <http://hal.archives-ouvertes.fr/hal-00727039>, 2012.

Approximating the Lovász θ Function with the Subgradient Method

Monia Giandomenico ¹

*Department of Information Engineering, Computer Science and Mathematics
University of L'Aquila
L'Aquila, Italy*

Adam N. Letchford ²

*Department of Management Science
Lancaster University
Lancaster, United Kingdom*

Fabrizio Rossi, Stefano Smriglio ¹

*Department of Information Engineering, Computer Science and Mathematics
University of L'Aquila
L'Aquila, Italy*

Abstract

The famous Lovász theta number $\theta(G)$ is expressed as the optimal solution of a semidefinite program. As such, it can be computed in polynomial time to an arbitrary precision. Nevertheless, computing it in practice yields some difficulties as the size of the graph gets larger and larger, despite recent significant advances of semidefinite programming (SDP) solvers. We present a way around SDP which exploits a well-known equivalence between SDP and lagrangian relaxations of non-convex quadratic programs. This allows us to design a subgradient algorithm which is shown to be competitive with SDP algorithms in terms of efficiency, while being preferable as far as memory requirements, flexibility and stability are concerned.

Keywords: Maximum Stable Set, Quadratic Programming, Lagrangian Relaxation, Subgradient Method.

1 Introduction

Given an undirected graph $G = (V, E)$, the Lovász θ number, introduced in [11], is defined as follows:

$$\begin{aligned} \theta(G) := \max \quad & \sum_{i \in V} Y_{0i} \\ \text{s.t. } & Y_{0i} = Y_{ii} \quad (i \in V) \\ & Y_{ij} = 0 \quad (\{i, j\} \in E) \\ & Y \in \mathcal{S}_{n+1}, \end{aligned} \tag{1}$$

where \mathcal{S}_{n+1} denotes the cone of real symmetric square positive semidefinite matrices of order $n+1$. $\theta(G)$ plays a central role in combinatorial optimization as it can be computed in polynomial time to an arbitrary precision (being the optimal value of a Semidefinite Program) [9] but, at the same time, provides tight bounds for the *stability number* $\alpha(G)$ and for the *chromatic number* $\chi(G)$ of a graph. In detail, $\alpha(G) \leq \theta(G) \leq \chi(\bar{G})$, where \bar{G} is the complement graph of G . These nice features led to several interesting results, such as a polynomial time algorithm to compute $\alpha(G)$ and $\chi(G)$ when G is a perfect graph [9]. However, most of the results based on the θ number have mainly theoretical significance. On the contrary, the practical potential of this strong bound has not yet been fully exploited. In fact, computing $\theta(G)$ of graphs from standard libraries has been a popular test problem for general SDP solvers since the beginning of their developments (we refer the reader to the repository [4] for a complete list of available resources on Semidefinite Programming). Among them, the boundary point method of Povh *et al.* [16] and the regularization method of Malick *et al.* [13] turned out to be efficient to compute $\theta(G)$.

Conversely, one can consider Lagrangian reformulations of structured SDPs (as in the Spectral Bundle method by Helmberg and Rendl [10]) or other non-linear approaches (such as those by Burer and Monteiro [1] and, recently, Malick and Roupin [14]). Indeed $\theta(G)$ can be computed as the optimal solution of a certain Lagrangian dual problem. In this paper we describe a subgradient algorithm for solving this dual approximately, and thereby approximating $\theta(G)$ from above. Of course, the subgradient method is a basic first-order method for non-differentiable optimisation, that is known to suffer from slow convergence in many cases. Nevertheless, and surprisingly, we found that our particular implementation gave results that have a good trade-off between tightness and computing time.

¹ Email: name.surname@univaq.it

² Email: a.n.letchford@lancaster.ac.uk

2 Lagrangian relaxation

A description of $\theta(G)$ can be derived starting from the standard quadratic formulation of the stable set problem:

$$\max \sum_{i \in V} x_i$$

$$x_i^2 - x_i = 0 \quad (i \in V) \tag{2}$$

$$x_i x_j = 0 \quad (\{i, j\} \in E) \tag{3}$$

If we associate multipliers λ_i to constraints (2) and μ_{ij} to (3), the resulting Lagrangian relaxation has the form:

$$L(\lambda, \mu) = \max_{x \in \mathbb{R}^{|V|}} f(x, \lambda, \mu) = \max_{x \in \mathbb{R}^{|V|}} \sum_{i \in V} (1 + \lambda_i)x_i - \sum_{i \in V} \lambda_i x_i^2 - \sum_{\{i, j\} \in E} \mu_{ij} x_i x_j$$

or the following more compact form:

$$L(\lambda, \mu) = \max_{x \in \mathbb{R}^{|V|}} [(\mathbf{1} + \lambda)^T x - x^T A(\lambda, \mu) x] \tag{4}$$

with $A(\lambda, \mu) = \text{Diag}(\lambda) + M(\mu)$, where $M(\mu)$ denotes the symmetric matrix with $\mu_{ij}/2$ in the i th row and j th column whenever $\{i, j\} \in E$, and zeroes elsewhere. In [8] it is shown that the optimal value of the Lagrangian dual problem yields the Lovász theta bound, that is:

$$\theta(G) = \min_{(\lambda, \mu) \in \mathbb{R}^{|V|+|E|}} L(\lambda, \mu) = L(\lambda^*, \mu^*) \tag{5}$$

where λ^* and μ^* coincide with the optimal dual solution to problem (1). Note that the function $f(x, \lambda^*, \mu^*)$ is concave and the matrix $A(\lambda^*, \mu^*)$ is positive semidefinite. We remark that Luz and Schrijver [12] showed that an optimal solution $A(\lambda, \mu)$ of problem (5) exists such that $\lambda = \mathbf{1}$ and the minimum eigenvalue λ_{\min} of $A(\mathbf{1}, \mu)$ is equal to zero. Observe however that imposing the second restriction causes the Lagrangian dual to become non-convex. We decided not to impose either restriction in our subgradient algorithm.

3 Subgradient algorithm

The subgradient algorithm is initialized with multipliers $\lambda^1 = \mathbf{1}$ and $\mu^1 = \mathbf{0}$, i.e., $A(\lambda^1, \mu^1) = I_{|V|}$. In all cases in which $A(\lambda, \mu)$ is positive definite (and obviously $I_{|V|}$ is so), the global minimizer \tilde{x} of the lagrangian relaxation (4)

Algorithm 1 Update multipliers

Input:	$g^k, h^k, L(\lambda^k, \mu^k), \epsilon$
Output:	true if succeeded, false otherwise, updated multipliers: λ^{k+1}, μ^{k+1} , lagrangian solution: \tilde{x}
Parameters:	$\gamma, \text{step_tolerance}$, an estimate of $L(\lambda^*, \mu^*) : \hat{L}$

```

/* Initialization */       $t := \gamma \frac{\hat{L} - L(\lambda^k, \mu^k)}{\|g^k\|^2 + \|h^k\|^2}$ 
/* Main loop */          while ( $t \geq \text{step\_tolerance}$ ) {
/* Multipliers update */  $\lambda^{k+1} := \lambda^k + tg^k, \mu^{k+1} := \mu^k + th^k;$ 
/* Check pd-ness */      if ( $\text{chol}(A(\lambda^{k+1}, \mu^{k+1}))$ )
/* Lambda perturbation */ then compute  $\tilde{x}$ ;
                           return true;
                           else  $\lambda^{k+1} := \lambda^{k+1} + \epsilon \cdot \mathbf{1}$ 
                                 if ( $\text{chol}(A(\lambda^{k+1}, \mu^{k+1}))$ )
                                 then compute  $\tilde{x}$ 
                           return true
/* Stepsize reduction */    $t := t/2;$ 
                           }
return false;

```

is unique and can be analytically computed, being the root of the equation $\nabla_x f(x, \lambda, \mu) = -2Ax + (\lambda + \mathbf{1}) = \mathbf{0}$.

At the generic iteration k of a subgradient algorithm, one has to evaluate the search directions $g^k \in \mathbb{R}^{|V|}$, $h^k \in \mathbb{R}^{|E|}$ and to update the multipliers. In order to improve practical performance, search directions are evaluated by using a *deflection* strategy, that is, by defining the new search directions as a combination of the previous directions and the current subgradients [5]. In our implementation we use the deflection described in [19]. In detail:

$$g^k = \nabla_\lambda f(\tilde{x}) + \delta^k g^{k-1}; \quad h^k = \nabla_\mu f(\tilde{x}) + \delta^k h^{k-1}; \quad \text{with } \delta^k = \frac{\|\nabla_{\lambda, \mu}^k f(\tilde{x})\|}{\|(g^{k-1}, h^{k-1})\|}.$$

As far as multiplier updates are concerned, we design an algorithm that maintains the matrix $A(\lambda^{k+1}, \mu^{k+1})$ positive definite through a “safeguard” strategy: Algorithm 1 receives the optimal value of the current Lagrangian relaxation $L(\lambda^k, \mu^k)$ and the search directions g^k, h^k . The procedure begins with updating the multipliers by a standard stepsize t . If the resulting matrix is positive definite, then multipliers λ^{k+1}, μ^{k+1} are returned and subgradient iteration $k+1$ starts. Otherwise, the positive definiteness is tentatively restored: first by applying a (small) perturbation ϵ to the diagonal of $A(\lambda^{k+1}, \mu^{k+1})$

(that is, to λ multipliers). Note that $\epsilon > |\lambda_{\min}(A(\lambda^{k+1}, \mu^{k+1}))|$ would guarantee the positive definiteness of the matrix. However, in practice, this can correspond to large values of ϵ that results in bound impairment since it is equivalent to adding a term $\epsilon \cdot \|x\|_2^2$ to $f(x, \lambda^{k+1}, \mu^{k+1})$. For this reason, we halve the stepsize and repeat the main loop until either the matrix is positive definite or t drops below a given tolerance `step_tolerance`.

The bottleneck of the algorithm is the pd-ness test that is carried out by the Cholesky Decomposition (see [9], Chapter 9.3 for details). Function `chol(A)` in Algorithm 1, based on the implementation developed in [2], returns `true` if the matrix is pd and `false` otherwise. If the matrix $A(\lambda^{k+1}, \mu^{k+1})$ is positive definite then `chol(A)` also returns a lower-triangular matrix L such that $A = LL^T$. This decomposition allows one to evaluate the current optimal solution of the lagrangian subproblem $-2Ax + (\lambda + \mathbf{1}) = \mathbf{0}$ being equivalent to the two triangular systems $-2Lz + (\lambda + 1) = 0$, $L^Tx = z$.

A further device of our implementation is that the main loop execution can be easily parallelized. Precisely, a number `n_threads` of different sets of new candidate multipliers associated with different values of t can be computed in parallel. If multiple sets yield a pd matrix, the one corresponding to the largest stepsize is taken.

4 Computational Results

The test bed consists of graphs from the DIMACS Second Challenge available at the web site [3]. The SDP solver used is `mprw2` by Malik et al., which is publicly available in [18], running under MATLAB (R2008b). Such an algorithm currently outperforms other approaches on medium/large size instances. The subgradient algorithm is implemented in C++ in a `linux` architecture (Ubuntu 12.04), compiler `g++ 4.7`. The computations were run on a machine equipped by 2 Intel Xeon 5150 processors (for a total of 4 cores) clocked at 2.6 GHz and having 8GB of RAM. The parameter setting of Algorithm 1 is as follows: $\gamma = 0.8$, $\epsilon \in \{10^{-6}, 10^{-4}\}$, `step_tolerance` = 10^{-12} . The subgradient algorithm stops when it reaches either an iteration limit or an optimality tolerance fixed to 10^{-6} (same as the SDP solver).

In Table 1, besides the graph name, size, density and stability number, we report: a bound on $\theta(G)$ and the CPU time required by [18] to compute it; the best value Z found by the subgradient algorithm and the corresponding CPU time; the values $\theta_{105\%} = 1.05 \cdot \theta$ and $\theta_{110\%} = 1.1 \cdot \theta$ along with the CPU times required by the subgradient algorithm to reach these values (** if $\theta_{105\%} < Z$).

The results give clear indications. In all cases Z is significantly close to $\theta(G)$, that is, the subgradient algorithm returns quite good solutions to (5). In several cases the SDP solver is faster, but also the subgradient algorithm takes reasonable CPU times. As pointed out in [13], the SDP solver performs poorly on some classes of instances (see, e.g., p_hat, san, sanr), while the subgradient algorithm still guarantees restrained CPU times, showing a more robust behaviour. This is also expressed by the times elapsed to reach $\theta_{105\%}$, $\theta_{110\%}$, which are quite competitive. This shows that the subgradient algorithm quickly computes quite good approximations of $\theta(G)$.

Overall, robustness and efficiency let the subgradient method be suitable to be embedded into a branch-and-bound algorithm to compute $\alpha(G)$ or $\chi(\bar{G})$. Recall that $\theta(G)$ turns out to be a much stronger upper bound for $\alpha(G)$ than those obtained by traditional polyhedral combinatorics techniques [6], used in previous branch-and-cut algorithms [17] [15]. In fact, several attempts have been recently made to design exact algorithms based on upper bounds close to the Lovász θ bound, either computed by SDP [20] or by LP [8] [7] [6] algorithms. The subgradient method proves to be a promising third competitor, as it allows faster reoptimization with respect to SDP algorithms and much easier implementation if compared to either LP or SDP methods.

References

- [1] Burer, S & R.D.C. Monteiro (2003) *A nonlinear programming algorithm for solving semidefinite programs via low-rank factorization*, Math. Program., Ser. B, **95**, 329–357.
- [2] Choi, J., J.J. Dongarra, L.S. Ostrouchov, A.P. Petitet, D.W. Walker and R.C. Whaley, *The Design and Implementation of the ScaLAPACK LU, QR and Cholesky Factorization Routines*, 1996.
- [3] DIMACS Repository
<ftp://dimacs.rutgers.edu/pub/challenge/graph/benchmarks/clique>.
- [4] Semidefinite Programming Repository
<http://www-user.tu-chemnitz.de/~helmburg/semdif.html>.
- [5] D'Antonio, G., and A. Frangioni, *Convergence Analysis of Deflected Conditional Approximate Subgradient Methods*, SIAM Journal on Optimization, **20/200** (1), (2009), 357–386. ISSN 1052-6234.
- [6] Giandomenico, M., Rossi, F., and S., Smriglio, *Strong lift-and-project cutting*

planes for the Stable Set Problem, Math. Program. Ser. A, online first DOI: 10.1007/s10107-012-0513-3.

- [7] Giandomenico, M., Letchford, A., Rossi, F., and S. Smriglio, *An Application of the Lovàsz-Schrijver $M(K, K)$ Operator to the Stable Set Problem*, Math. Program. Ser. A, **120/2** (2009), 381–401.
- [8] Giandomenico, M., Letchford, A.N., Rossi, F., and S., Smriglio, *A new approach to the stable set problem based on ellipsoids*, Lecture Notes in Computer Science, LNCS 6655, Proc. of the 15th IPCO conference, (2011), NY, 223–234.
- [9] Grötschel, M., Lovász, L., and A.J. Schrijver *Geometric Algorithms in Combinatorial Optimization*. New York: Wiley, 1988.
- [10] Helmberg, C., and F. Rendl, *A Spectral Bundle Method for Semidefinite Programming*, SIAM Journal on Optimization, **10**, (1997), 673–696.
- [11] Lovász, L., *On the Shannon capacity of a graph*, IEEE Trans. Inform. Th., IT-25, (1979), 1–7.
- [12] Luz, C.J., and A.J. Schrijver, *A Convex Quadratic Characterization of the Lovász Theta Number*, SIAM J. Discrete Math., **19**(2) (2005), 382–387.
- [13] Malick, J., Povh, J., Rendl, F., and A. Wiegele, *Regularization Methods for Semidefinite Programming*, SIAM J. Optimization, **20**(1) (2009), 336–356.
- [14] Malick, J., and F. Roupin, *On the bridge between combinatorial optimization and nonlinear optimization: a family of semidefinite bounds for 0-1 quadratic problems leading to quasi-Newton methods*, Math. Program. Ser. B, to appear.
- [15] Nemhauser, G.L., and G., Sigismondi, *A strong cutting plane/branch-and-bound algorithm for node packing*, J. Opl Res. Soc., **43** (1992), 443–457.
- [16] Povh, J., Rendl, F., and A. Wiegele, *A boundary point method to solve semidefinite programs*, Computing, **78** (2006), 277–286.
- [17] Rossi, F., and S., Smriglio, *A branch-and-cut algorithm for the maximum cardinality stable set problem*, Oper. Res. Lett., **28** (2001), 63–74.
- [18] SDP solver `mprw2.m`, Alpen-Adria-Universität Klagenfurt website
<http://www.math.uni-klu.ac.at/or/Software>.
- [19] Sherali, H.D., and O., Ulular, *A primal-dual conjugate subgradient algorithm for specially structured linear and convex programming problems*, Applied Mathematics and Optimization, **20** (1989), 193–221.
- [20] Wilson, A.T., *Applying the boundary point method to an SDP relaxation of the maximum independent set problem for a branch and bound algorithm*, Master thesis, New Mexico Institute of mining and technology, 2009.

Graph	$ V $	density	α	SDP mprw2		Z	Time	Subgradient			Time
				$\theta(G)$	Time			$\theta_{105\%}$	Time	$\theta_{110\%}$	
brock200_1	200	0.25	21	27.45	20.75	27.75	25.21	28.82	5.80	30.20	2.52
brock200_2	200	0.50	12	14.23	19.65	14.58	35.21	14.94	19.44	15.65	6.57
brock200_3	200	0.39	15	18.82	18.97	19.08	33.65	19.76	11.45	20.70	4.69
brock200_4	200	0.34	17	21.29	19.51	21.53	35.59	22.35	10.19	23.42	3.95
brock400_1	400	0.25	27	39.70	134.56	40.35	140.83	41.69	45.13	43.67	18.22
brock400_2	400	0.25	29	39.56	133.63	40.02	209.64	41.54	39.09	43.52	14.72
brock400_3	400	0.25	31	39.48	133.35	39.99	175.24	41.45	34.45	43.43	15.07
brock400_4	400	0.25	33	39.60	133.27	40.13	164.37	41.58	40.99	43.56	12.99
C.125.9	125	0.10	34	37.81	8.29	37.99	14.48	39.70	0.42	41.59	0.16
C.250.9	250	0.10	44	56.24	45.46	56.77	54.02	59.05	18.47	61.87	5.96
c-fat200-1	200	0.57	12	12.00	28.27	12.42	60.34	12.60	40.23	13.20	10.91
c-fat200-2	200	0.84	24	24.00	763.17	24.09	28.09	25.20	5.15	26.40	1.25
c-fat200-5	200	0.57	58	60.32	113.59	60.57	10.12	63.34	9.18	66.35	8.60
DSJC125.1	125	0.09	34	38.40	8.70	38.49	12.23	40.32	0.99	42.24	0.12
DSJC125.5	125	0.50	10	11.47	6.00	11.68	5.53	12.05	2.06	12.62	0.96
DSJC125.9	125	0.10	4	4.00	10.90	4.09	4.68	4.20	3.46	4.40	2.31
gen200_p0.9_44	200	0.10	44	44.00	71.40	44.95	70.82	46.20	21.86	48.40	7.09
gen200_p0.9_55	200	0.10	55	55.38	791.25	55.28	52.64	58.15	0.41	60.92	0.13
hamming6-2	64	0.10	32	32.05	2.65	32.00	0.01	33.60	< 0.01	35.20	< 0.01
hamming6-4	64	0.65	4	5.33	0.24	5.37	2.35	5.61	0.97	5.86	0.45
hamming8-2	256	0.03	128	128.00	228.81	128.00	0.03	134.40	< 0.01	140.80	< 0.01
hamming8-4	256	0.36	16	16.00	12.44	16.52	80.35	16.80	17.17	17.60	8.65
johnson8-2-4	28	0.44	4	4.00	0.03	4.00	1.1	4.20	0.32	4.40	0.04
johnson8-4-4	70	0.23	14	14.00	0.48	14.00	0.01	14.70	< 0.01	15.40	< 0.01
johnson16-2-4	120	0.24	8	8.00	1.37	8.18	14.22	8.40	9.31	8.80	4.91
keller4	171	0.35	11	14.01	10.77	14.17	22.34	14.71	8.53	15.41	3.00
p_hat300-1	300	0.76	8	10.07	119.36	10.57	142.32	10.57	142.32	11.08	79.79
p_hat300-2	300	0.51	25	26.97	423.82	27.25	171.6	28.32	22.07	29.66	10.89
p_hat300-3	300	0.26	36	41.17	204.77	41.51	157.7	43.23	20.63	45.29	5.72
p_hat500-1	500	0.75	9	13.07	416.25	14.01	568.74	13.73	***	14.38	441.85
p_hat500-2	500	0.50	36	38.98	1927.48	39.53	717.02	40.93	104.33	42.88	31.61
p_hat500-3	500	0.25	50	58.57	1136.02	59.14	940.9	61.50	166.82	64.43	44.32
p_hat700-1	700	0.75	11	15.12	1029.39	16.55	2797.81	15.88	***	16.63	2605.65
p_hat700-2	700	0.50	44	49.02	5319.12	49.78	1807.36	51.47	149.32	53.93	55.08
p_hat700-3	700	0.25	62	72.74	4277.59	74.09	778.5	76.37	174.57	80.01	57.97
san200_0.7-1	200	0.30	30	30.00	684.73	30.00	0.75	31.50	0.24	33.00	0.12
san200_0.7-2	200	0.30	18	18.00	764.01	18.44	34.72	18.90	13.84	19.80	3.23
san200_0.9-1	200	0.10	70	70.00	813.33	70.00	0.68	73.50	0.10	77.00	0.03
san200_0.9-2	200	0.10	60	60.00	798.70	60.00	0.85	63.00	0.04	66.00	0.01
san200_0.9-3	200	0.10	44	44.00	691.16	44.67	33.63	46.20	5.78	48.40	1.28
san400_0.5-1	400	0.50	13	13.00	3761.83	13.41	527.08	13.65	401.04	14.30	120.34
san400_0.7-1	400	0.30	40	40.00	4373.72	40.02	48.12	42.00	5.30	44.00	1.30
san400_0.7-2	400	0.30	30	30.00	4007.51	30.04	187.61	31.50	56.21	33.00	23.32
san400_0.7-3	400	0.30	22	22.00	449.68	22.77	879.23	23.10	600.32	24.20	457.32
san400_0.9-1	400	0.10	100	100.00	4963.48	100.50	5.71	105.00	2.10	110.00	0.97
sannr200_0.7	200	0.30	18	23.83	21.06	23.95	41.06	25.02	7.00	26.21	1.95
sannr200_0.9	200	0.10	42	49.27	28.88	49.64	57.61	51.73	10.02	54.20	4.10

Table 1
Computational results

An integer fixed-charge multicommodity flow (FCMF) model for train unit scheduling[★]

Zhiyuan Lin^{a,1} Raymond S. K. Kwan^{a,1}

^a *School of Computing, University of Leeds, Leeds, UK, LS2 9JT*

Abstract

An integer fixed-charge multicommodity flow (FCMF) model is used as the first part of a two-phase approach for train unit scheduling, and solved by an exact branch-and-price method. To strengthen knapsack constraints and deal with complicated scenarios arisen in the integer linear program (ILP) from the integer FCMF model, preprocessing is used by computing convex hulls of sets of points representing all possible train formations utilizing multiple unit types.

Keywords: train unit scheduling, fixed-charge multicommodity flow, convex hull

1 Introduction

A *train unit* is a set of train carriages (or *cars*) with its own built-in engine(s). Without a locomotive, it is able to move in both directions on its own. A train unit can also be coupled with other units of the same or similar types.

Given a railway operator's timetable on a particular week day, and a fleet of train units of different types, the train unit scheduling problem aims at

[★] This research is sponsored by a Dorothy Hodgkin Scholarship funded by the Engineering and Physical Sciences Research Council (EPSRC) and Tracsis Plc.

¹ Email: {tsz1,r.s.kwan}@leeds.ac.uk

determining an assignment plan such that each train trip (or shortly *train*) is appropriately covered by a single or coupled units, with certain objectives achieved and certain constraints respected. From the perspective of a train unit, the scheduling process assigns a sequence of trains to it as its daily workload. The main objectives are to minimize the number of units used and/or the operational costs. It is also a common requirement to meet the passenger capacity demands.

Besides the basic requirements like trip covering, fleet size limit, compatibility between vehicle types and routes, etc, the train unit scheduling problem is found to be difficult due to some of its particular features listed below:

- Train units may be coupled/decoupled in response to passenger capacity demands for particular trains. Coupling/decoupling may take time that is not negligible, and some locations are banned/restricted for such activities.
- There are compatibility relationships among unit types when coupled.
- Unit coupling for individual trains is limited by an upper bound on the number of coupled cars, determined by many factors such as unit types, routes, platform lengths, time horizon, etc.
- Passenger capacity demand is expressed in number of seats (rather than number of units), leading to knapsack constraints that may yield very weak linear programming (LP) relaxations.
- Units may block each other on tracks and/or at platforms.

In [6], the authors have proposed a two-phase approach for the train unit scheduling problem. This approach is consisted of Phase I, an integer FCMF model, and Phase II, a multidimensional matching model resolving unit station blockage. The Phase I model is similar to [4], but important additional abilities for real-world conditions have been added, including forbidding coupling/decoupling at banned/restricted locations, ensuring coupling/decoupling time allowances, type compatibility, etc. The model explicitly uses knapsack constraints to satisfy many requirements, which has drawbacks as mentioned above. Moreover, the model uses extra variables and constraints to achieve unit type compatibility, which significantly increases the number of ILP constraints.

In this paper, we propose an updated variant of the Phase I model in [6]. The variant employs a convex hull computation preprocessing, with less number of constraints and much stronger LP relaxation, and is more flexible for complicated scenarios. The convex hull method is similar to previous works by Schrijver [8] , Ziatati et al [9], Cacchiani et al [4][3], except we deal with

an improved new model with important additional features derived from [6], take a more straightforward enumeration to satisfy complicated validity rules, and consider type compatibility issues with an ad hoc branching method.

2 Model and formulation

2.1 Model description

Similar to [4] and [6], the integer FCMF model is based on a *directed acyclic graph* (DAG) $\mathcal{G} = (\mathcal{N}, \mathcal{A})$. We define the node set $\mathcal{N} = N \cup \{s, t\}$, where N is the set of *train nodes*, and s and t are the *source* and *sink* node; the arc set $\mathcal{A} = A \cup A_0$, where A is the *linkage arc* set and A_0 the *sign-on/off arc* set. A linkage arc $a \in A$ links two train nodes i and j ($a = (i, j)$, $i, j \in N$), representing a potential link such that after serving train i a unit can continue to serve train j as its next task. A sign-on arc $(s, j) \in A_0$ starts from s and ends at a train node $j \in N$; a sign-off arc $(j, t) \in A_0$ starts from a train node $j \in N$ and ends at t . Generally all train nodes have a sign-on arc and a sign-off arc. We use $\delta_-(j)$ and $\delta_+(j)$ to denote all arcs that terminate at / originate from node j respectively. Finally, a path $p \in P$ in \mathcal{G} represents a sequenced daily workload (the train nodes in the path) for a unit.

For the fleet, we denote K the set of all unit types. As for type-route compatibility, type-graphs \mathcal{G}^k representing routes each type $k \in K$ can serve are constructed based on \mathcal{G} , including all entities, e.g. P^k refers to the set of all paths in type-graph \mathcal{G}^k .

It is well-known that for multicommodity flow problems there are two equivalent formulations based on arcs and paths, with the same LP-relaxation bounds ([5] [4]). The model in this paper uses the latter, with *path variable* $x_p \in \mathbb{Z}_+, \forall p \in P^k, \forall k \in K$ to indicate the number of units used in path p . Note there is other *blockflow variable* $y_a \in \{0, 1\}, \forall a \in \mathcal{A}$ indicating whether an arc a is used.

2.2 Model ILP formulation

$$(P) \quad \min \quad W_1 \sum_{k \in K} \sum_{p \in P^k} c_p x_p + W_2 \sum_{a \in \mathcal{A}} y_a \quad (1)$$

$$\sum_{p \in P^k} x_p \leq b^k, \quad \forall k \in K; \quad (2)$$

$$\sum_{k \in K_j} \sum_{p \in P_j^k} D_{f,k}^j x_p \leq d_f^j, \quad \forall f \in \bar{F}_j, \forall j \in N. \quad (3)$$

$$\sum_{k \in K_a} \sum_{p \in P_a^k} x_p \leq m_a y_a, \quad \forall a \in \mathcal{A}; \quad (4)$$

$$\sum_{a \in \delta_-(j)} y_a = 1, \quad \forall j \in N_B^-; \quad \sum_{a \in \delta_+(j)} y_a = 1, \quad \forall j \in N_B^+; \quad (5)$$

$$\tau_{\text{arr}(i)}^D \left(\sum_{a \in \delta_+(i)} y_a - 1 \right) + \tau_{\text{dep}(j)}^C \left(\sum_{a \in \delta_-(j)} y_a - 1 \right) \leq e_{ij}, \quad \forall (i, j) \in A^*; \quad (6)$$

$$x_p \in \mathbb{Z}_+, \quad \forall p \in P^k, \forall k \in K; \quad y_a \in \{0, 1\}, \quad \forall a \in \mathcal{A}. \quad (7)$$

In the first term of Objective (1), c_p is the overall cost for path p . It can be a weighted linear combination of several sub-costs to achieve multiple effects and/or preferences. The second term minimizes total number of *used* arcs, leading to minimization of the total number of coupling/decoupling activities, thus eliminating unnecessary ones; it also partly drives y_a to desired binary values. $W_{1,2}$ are weights for the two terms, and generally $W_1 > W_2$.

Constraints (2) ensure the deployed number of units for each type $k \in K$ is within its fleet size upper bound b_k . Constraints (3) are used to fulfil the following requirements for individual trains: (i) passenger capacity demands; (ii) coupling upper bounds due to various factors; and (iii) type compatibility. Constraints (3) will be discussed in detail in Section 2.3. Constraints (4) calculate binary blockflow variables $y_a, \forall a \in \mathcal{A}$ such that $y_a = \begin{cases} 1, & \sum_{k \in K_a} \sum_{p \in P_a^k} x_p > 0, \\ 0, & \sum_{k \in K_a} \sum_{p \in P_a^k} x_p = 0, \end{cases}$, where P_a^k is the set of paths in \mathcal{G}^k passing through arc a , K_a is the type set allowed at arc a , and m_a is the maximum number of coupled units flowing through arc a . Constraints (5) are to forbid coupling/decoupling at banned locations, where $N_B^- \subseteq N$ is the set of trains whose departure locations are banned for coupling/decoupling, and $N_B^+ \subseteq N$ for arrival locations. Constraints (6) are to ensure time allowance validity for coupling/decoupling at some linkage arcs $A^* \subseteq A$ where a time violation may occur; $\tau_{\text{arr}(i)}^D$ is the time for a single decoupling operation at the arrival location of train i , $\tau_{\text{dep}(j)}^C$ the time for a single coupling operation at the departure location of train j , and e_{ij} is the time slack between i and j . Finally (7) gives the variable domain.

2.3 Computation for convex hulls associated with trains

Similar to [4], let K_j be the set of permitted types for train $j \in N$, and $w^j = (w_1^j, w_2^j, \dots, w_{|K_j|}^j)^T \in \mathbb{Z}_+^{K_j}$, where w_k^j is the number of units of type k used for j . We define a *unit combination set* by enumerating all valid unit formation for a train:

$$\mathcal{W}_j := \left\{ w^j \in \mathbb{Z}_+^{K_j} \mid \forall w^j : \text{a valid unit combination for train } j \right\}, \quad (8)$$

such that

- (i) $\sum_{k \in K_j} q_k w_k^j \geq r_j$, where q_k is the capacity of unit type k and r_j is the passenger capacity demand, both measured in numbers of seats.
- (ii) $n(w^j) \leq u(w^j), \forall w^j \in \mathcal{W}_j$, where $n(w^j)$ and $u(w^j)$ are number of coupled cars and coupling upper bound for combination w^j respectively.
- (iii) the used types ($k : w_k^j > 0$) are compatible;

Notice while it is easy to satisfy demand requirement as in (i) by linear constraints, it is generally very difficult for (ii) and (iii) in the same way. Moreover, enumeration generally suits any validity rules, which in our cases are more complicated than those in [4] and [6]. It also avoids knapsack constraints and excludes a large proportion of incompatible types.

For all \mathcal{W}_j in our instances, due to their small dimensions ($|K_j| \leq 4, \forall j \in N$), and numbers of points ($|\mathcal{W}_j| \leq 9, \forall j \in N$), it is possible to explicitly compute the convex hulls of all unit combination sets, which we refer to as *train convex hulls*:

$$\text{conv}(\mathcal{W}_j) = \left\{ w^j \in \mathbb{R}_+^{K_j} \mid D^j w^j \leq d^j \right\}, \quad \forall j \in N, \quad (9)$$

described by a set of facets $f \in F_j$, with $D^j \in \mathbb{R}^{F_j \times K_j}$ and $d \in \mathbb{R}^{F_j}$. Note we only need the nonzero facets $f \in \bar{F}_j \subseteq F_j$. Finally, all w -variables will be replaced by x -variables in the same way as in [4], which forms Constraints (3) in (P) .

Example We use an example from the real data of a UK rail operator to illustrate the above preprocessing approach. For a train “1C11”, train units of Type 318 (3-car, 219 seats), 320 (3-car, 230 seats) and 156 (2-car, 145 seats) are permitted, and only Type 318 and 320 are compatible for coupling. When served by Type 318 and/or 320, the coupling upper bound is 6 cars, and when

served by Type 156, this bound changes to 4 cars. In addition, “1C11” has a passenger capacity demand of 230 seats. Then we have:

$$\mathcal{W}_{1\text{C}11} = \{(w_{318}, w_{320}, w_{156}) | (2, 0, 0), (0, 1, 0), (0, 2, 0), (1, 1, 0), (0, 0, 2)\},$$

and its corresponding train convex hull:

$$\text{conv}(\mathcal{W}_{1\text{C}11}) = \left\{ w \in \mathbb{R}_+^3 \middle| \begin{array}{l} f_1 : w_{318} + 2w_{320} + w_{156} \geq 2 \\ f_2 : w_{318} + w_{320} + w_{156} \leq 2 \end{array} \right\},$$

which is a polytope with two nonzero facets $\{f_1, f_2\} = \bar{F}_{1\text{C}11}$, giving two corresponding constraints for “1C11” in Constraints (3).

3 Solution approach

3.1 A branch-and-price ILP solver

The above ILP (P) is solved by a branch-and-price [2] method based on column generation [7] where path variables x_p are generated dynamically.

Let $\alpha^k \leq 0, \beta_{f,j} \leq 0, \gamma_a \leq 0$ be the dual variables from Constraints (2), (3) and (4), N_p and \mathcal{A}_p the set of train nodes and arcs in path p respectively, the reduced cost of a path $p \in P^k$ is,

$$\bar{c}_p = c_p - \alpha^k - \sum_{j \in N_p} \sum_{f \in \bar{F}_j} D_{f,k}^j \beta_{f,j} - \sum_{a \in \mathcal{A}_p} \gamma_a, \quad (10)$$

finding the smallest value of which can be regarded as a shortest path problem with train node weight $-\sum_{f \in \bar{F}_j} D_{f,k}^j \beta_{f,j}$, $\forall j \in N^k$, arc weight $-\gamma_a$, $\forall a \in \mathcal{A}^k$, plus a source-sink weight $c_p - \alpha^k$. There are $|K|$ subproblems from Dantzig-Wolfe decomposition.

3.2 Branching rules

Although all vertices are feasible points for each $\text{conv}(\mathcal{W}_j)$, it may still contain points with incompatible types serving the same train, which have to be eliminated. This can be realized by a *train-family branching* proposed in [6]. We introduce the concept of *train unit family* such that unit types that are compatible belong to the same family. The symbol Φ_j is used to denote the set of all families that can serve train j .

The main idea of train-family branching is to check the LP relaxation solution and select a train j that is covered by more than one family, say

families $\phi_1, \phi_2, \dots, \phi_n$ (n is usually not a large number). Then we form $n + 1$ branches with respect to families $\phi_1, \phi_2, \dots, \phi_n$.

- For the first n branches $1, \dots, n$, say at a branch $i \in \{1, \dots, n\}$, only family ϕ_i is allowed to serve train j . To achieve this, in the restricted master problem (RMP), all paths indicating any families in $\Phi_j \setminus \{\phi_i\}$ serving j are deleted; in the shortest path problem of type k whose family is not ϕ_i , node j is deleted from the shortest path network.
- In the last $(n+1)$ th branch, if $|\Phi_j \setminus \{\phi_1, \dots, \phi_n\}| \geq 1$, then we forbid families ϕ_1, \dots, ϕ_n to serve train j , which can be realized by similar path/node deleting ways as described above; if $\Phi_j = \{\phi_1, \dots, \phi_n\}$, then the $(n + 1)$ th branch is no longer needed.

Notice this branching rule does not add any extra constraints to the RMP. It actually reduces the number of columns in the RMP and the network scale of the shortest path subproblems, which effectively divides the search space and forces the trains to be served by compatible types.

It should be mentioned that it is not enough to drive all variables into integers only by train-family branching. When all trains have been served by compatible types, the branching strategy will be switched to *arc variable branching*, e.g. the method proposed in [1] for integer multicommodity flow problems.

4 Preliminary experiments and conclusions

The experiments are based on the data from a UK rail operator with around 2250 train trips on a weekday. All possible unit combinations have been computed, and the `convhull()` and `convhulln()` functions provided by Matlab R2012a are used for convex hull computation. We find that the numbers of nonzero facets of train convex hulls are very small (2.11 facets per train on average), which can be proved in a similar way as in [9]. Among the 2250 trains, 51 trains (2.2%) have a number of nonzero facets of 1, 1987 trains (88.3%) of 2, 176 trains (7.8%) of 3, 13 trains (0.57%) of 5, and 23 trains (1.0%) of 7. This implies in the ILP (P) the number of constraints will be greatly reduced, compared with the original model. Further results and analysis will be presented at INOC2013 and in a post-conference full paper.

This paper has proposed an updated model of a previous version for real-world train unit scheduling. A method by explicitly computing convex hulls for each train is used thereby making it easier to solve the complex integer model within practical computational time. The preliminary experiments have

shown promising results that the numbers of facets for most train convex hulls are small.

Testing and analyzing the final scheduling results with the rail operator we are collaborating are ongoing.

References

- [1] Alvelos, F., “Branch-and-Price and Multicommodity Flows,” Ph.D. thesis, Escola de Engenharia, Universidade do Minho, Portugal (2005).
- [2] Barnhart, C., E. L. Johnson, G. L. Nemhauser, M. W. P. Savelsbergh and P. H. Vance, *Branch-and-price: Column generation for solving huge integer programs*, Operations Research **46** (1998), pp. 316–329.
- [3] Cacchiani, V., A. Caprara, G. Maroti and P. Toth, *On integer polytopes with few nonzero vertices*, Operations Research Letters **41** (2013), pp. 74–77.
- [4] Cacchiani, V., A. Caprara and P. Toth, *Solving a real-world train-unit assignment problem*, Mathematical Programming **124** (2010), pp. 207–231.
- [5] Cook, W. J., W. H. Cunningham, W. R. Pulleyblank and A. Schrijver, “Combinatorial Optimization,” Wiley, New York, 1998.
- [6] Lin, Z. and R. S. K. Kwan, *A two-phase approach for real-world train unit scheduling*, in: *Proceedings of the 12th Conference on Advanced Systems for Public Transport*, Santiago, Chile, 2012.
- [7] Lübecke, M. E. and J. Desrosiers, *Selected topics in column generation*, Operations Research **53** (2002), pp. 1007–1023.
- [8] Schrijver, A., *Minimum circulation of railway stock*, CWI Quarterly **6** (1993), pp. 205–217.
- [9] Ziarati, K., F. Soumis, J. Desrosiers and M. M. Solomon, *A branch-first, cut-second approach for locomotive assignment*, Manage Sci **45** (1999), pp. 1156–1168.

The Degree Preserving Spanning Tree Problem: Valid Inequalities and Branch-and-cut method

Bernard Gendron ¹

Département d'informatique et de recherche opérationnelle, Université de Montréal, Interuniversity Research Centre on Enterprise Networks, Logistics and Transportation (CIRRELT), Bernard.Gendron@cirrelt.ca

Abilio Lucena ²

*Departamento de Administração and Programa de Engenharia de Sistemas e Computação, Universidade Federal do Rio de Janeiro, Rio de Janeiro, Brasil
abiliolucena@globo.com*

Alexandre Salles da Cunha ³

Departamento de Ciência da Computação, Universidade Federal de Minas Gerais, Belo Horizonte, Brasil, acunha@dcc.ufmg.br

Luidi Simonetti ⁴

*Instituto de Computação, Universidade Federal Fluminense, Niteroi, Brasil,
luidi@ic.uff.br*

Abstract

Given a connected undirected graph G , the Degree Preserving Spanning Tree Problem (DPSTP) consists in finding a spanning tree T of G that maximizes the number of vertices that have the same degree in T and in G . In this paper, we introduce Integer Programming formulations, valid inequalities and a Branch-and-cut algorithm

for the DPSTP. Reinforced with new valid inequalities, the upper bounds provided by the formulation behind our Branch-and-cut method dominate previous DPSTP bounds in the literature.

Keywords: Degree preserving trees, Vertex feedback edge set, Branch-and-cut

1 Introduction

Assume we are given a connected undirected graph $G = (V, E)$ with set of vertices $V = \{1, \dots, n\}$ and edges E ($m = |E|$). For any subgraph $G' = (V, E')$ of G ($E' \subseteq E$), let $\delta_{G'}(i)$ denote the set of edges of E' that are incident to $i \in V$. Whenever $|\delta_G(i)| = |\delta_{G'}(i)|$, i is said to be of *full degree* in G' . The Degree Preserving Spanning Tree Problem (DPSTP) asks for a spanning tree $T = (V, E_T)$ of G , with the maximum number of full degree vertices.

DPSTP has also been studied in [4] under its complement problem named *The Vertex Feedback Edge Set Problem* (VFESP): find a cotree (the complement of a tree) of G that is incident to the minimum number of vertices of V . The vertices that are not incident to the edges in the cotree are exactly those with full degree in its accompany tree. Thus, a solution to one problem is readily available, once a solution to the other is given.

DPSTP models an application in water distribution networks [7], where one has to decide where to install pressure meters, in order to measure flow in the network. According to [7], it suffices to install meters only at the vertices that are incident to the edges in a cotree of the network. Therefore, minimizing the number of vertices incident to the cotree is the most economical choice.

DPSTP is NP-complete, even if G is a split or bipartite planar graph [1,2]. If it is planar, DPSTP admits a linear time approximation algorithm. DPSTP is polynomially solvable when G has a bounded treewidth [2]. The directed version of DPSTP was studied in [5]. To date, solution approaches to DPSTP were mostly restricted to approximation algorithms [1,2] and heuristics [7,4]. The only DPSTP exact approach is the Branch-and-bound algorithm in [1],

¹ Bernard Gendron is partially funded by the Natural Sciences and Engineering Council of Canada (NSERC) Discovery Grants Program

² Abilio Lucena is partially funded by CNPq grant 310561/2009-4 and FAPERJ grant E26-110.552/2010

³ Alexandre Salles da Cunha is partially funded by CNPq grants 302276/2009-2, 477863/2010-8 and FAPEMIG grant PRONEX APQ-01201-09

⁴ Luidi Simonetti is partially funded by FAPERJ and CNPq grants 483.243/2010-8, 304793/2011-6

for which almost no computational results are available.

In this paper, we discuss Integer Programming (IP) formulations and implement a Branch-and-cut algorithm for one of them. Reinforced with new valid inequalities, the Linear Programming (LP) bounds behind the Branch-and-cut algorithm dominate those implied by the multi-commodity flow model in [1].

The paper is organized as follows. In Section 2, we present the formulations and the valid inequalities. The Branch-and-cut algorithm is described next, in Section 3. In Section 4, we report preliminary computational results. The paper is closed in Section 5, suggesting directions for future research.

2 IP Formulations and Valid Inequalities

Optimization problems that ask for optimal trees with side constraints can be formulated in different ways, depending on how solution connectivity is enforced. We formulate DPSTP as an IP where directed cutset constraints are used. Connectivity could be imposed by other means, for example, by undirected subtour breaking inequalities or by multi-commodity flow balance constraints (see [6]). Our choice is motivated by two reasons. For DPSTP, it provides the same bounds implied by the other two mentioned models. However, it was observed in the literature that computing the bounds implied by directed cutset formulations is usually conducted faster.

Our formulation for DPSTP makes use of a digraph $D = (V, A)$, where $A = \{(i, j) \cup (j, i) : \{i, j\} \in E\}$ is an orientation of the edges in E . Given a root vertex $r \in V$, the topology we look for is a spanning arborescence of D , rooted out of r , with maximum number of full degree vertices. The following variables are used:

- $\{x_{ij} \in \{0, 1\} : (i, j) \in A\}$ to indicate whether ($x_{ij} = 1$) or not ($x_{ij} = 0$) an arc $(i, j) \in A$ is included in the arborescence, and,
- $\{y_i \in \{0, 1\} : i \in V\}$ to indicate whether ($y_i = 1$) or not ($y_i = 0$) $i \in V$ has full degree.

We use the following notation. \mathbb{B} denotes the set $\{0, 1\}$. Given $S \subseteq V$, define $y(S) := \sum_{i \in S} y_i$, $E(S) := \{\{i, j\} \in E : i, j \in S\}$ and $A(S, V \setminus S) := \{(i, j) \in A : i \in S, j \notin S\}$. Accordingly, given $M \subseteq A$, define $x(M) := \sum_{(i,j) \in M} x_{ij}$. We assume that \mathcal{T} denotes the set of all spanning trees of G . For a given spanning tree $T = (V, E_T)$ ($T \in \mathcal{T}$) and $e \in E \setminus E_T$, let $C_{T,e}$ denote the set of vertices in the unique cycle of subgraph $(V, E_T \cup \{e\})$. For any formulation P for DPSTP, $w(P)$ denotes the LP upper bounds P implies.

The formulation is given by:

$$w = \max \{y(V) : (x, y) \in P_1 \cap \mathbb{B}^{2m+n}\}, \quad (1)$$

where polytope P_1 is implied by:

$$x(A(V \setminus \{i\}, \{i\})) = 1, \forall i \in V \setminus \{r\} \quad (2a)$$

$$x(A(V \setminus S, S)) \geq 1, \forall S \subset V \setminus \{r\}, S \neq \emptyset \quad (2b)$$

$$y_i - x_{ij} - x_{ji} \leq 0, \forall i \in V, \{i, j\} \in \delta_G(i) \quad (2c)$$

$$x_{ij} \geq 0, \forall (i, j) \in A, \quad (2d)$$

$$y_i \geq 0, \forall i \in V. \quad (2e)$$

Constraints (2a) impose that there must be one incident arc to every vertex $i \in V \setminus \{r\}$. Directed cutset constraints (2b) guarantee that there must exist a path connecting r to every other vertex in the solution. Constraints (2c) enforce that i is of full degree if one orientation of every edge $\{i, j\} \in \delta_G(i)$, (i, j) or (j, i) , is included in the arborescence.

In addition to $x \in \mathbb{B}^{2m}$, the multi-commodity flow formulation in [1] uses real valued flow variables and flow balance constraints to ensure connectivity. The projection of that spanning tree formulation into the x space is precisely the intersection of (2a),(2b) and (2d) [6]. The formulation in [1] also uses variables $y \in \mathbb{B}^n$ and constraints (2c) to define full degree vertices. As a result, P_1 and the formulation in [1] provide the same LP bound, $w(P_1)$.

Consider now the following degree enforcing inequalities:

$$x(A(\{i\}, V \setminus \{i\})) - y_i \leq |\delta_G(i)| - 2, \forall i \in V \setminus \{r\} \quad (3a)$$

$$x(A(\{r\}, V \setminus \{r\})) - y_r \leq |\delta_G(r)| - 1, \quad (3b)$$

$$x(A(\{i\}, V \setminus \{i\})) - (|\delta_G(i)| - 1)y_i \geq 0, \quad \forall i \in V \setminus \{r\} \quad (3c)$$

Constraints (3a) impose that, if $i \in V \setminus \{r\}$, at most $|\delta_G(i)| - 2$ (resp. $|\delta_G(i)| - 1$) arcs must leave i in a solution, in case $y_i = 0$ (resp. in case $y_i = 1$). Constraints (3c) guarantee that, if $y_i = 1$, exactly $|\delta_G(i)| - 1$ arcs must leave i . Note that, summing up constraints (2c) for all $\{i, j\} \in \delta_G(i)$ and using (2a), one ends up with $x(A(\{i\}, V \setminus \{i\})) \geq |\delta_G(i)|y_i - 1$, which is weaker than (3c). Stronger LP upper bounds are thus obtained if (3) are added to P_1 . Note that the bounds of the resulting strengthened formulation are not necessarily symmetric with respect to the root choice r .

For the next family of valid inequalities, assume that $C \subseteq V$ defines a set of vertices. If $|E(C)| \geq |C|$, subgraph $(C, E(C))$ includes at least one cycle of G . Since in such cases at least one edge from $E(C)$ must be out of any

acyclic subgraph of G , at least two vertices of C cannot be of full degree in any spanning tree of G . Therefore, $y(C) \leq |C| - 2$ is valid for DPSTP. Actually, such constraint can be generalized to

$$y(C) \leq |C| - \alpha_C, \quad (4)$$

where $\alpha_C \geq 0$ denotes the minimum number of vertices spanned by any cotree of $(C, E(C))$. If $(C, E(C))$ defines an acyclic subgraph, $\alpha_C = 0$. If $(C, E(C))$ defines a cycle of G , $\alpha_C = 2$. Finally, if $(C, E(C))$ defines a clique of G , $\alpha_C = |C| - 1$. The evaluation of the optimal values α_C for other subgraphs involves solving one VFESP instance, defined for the subgraph $(C, E(C))$ of G . However, given any lower bound $\bar{\alpha}_C$ on α_C , one could replace $|C| - \alpha_C$ by $\lfloor |C| - \bar{\alpha}_C \rfloor$ in a weakened version of (4).

Finally, note that if $C = V$, $\alpha_C = n - w$ and that DPSTP can also be formulated as the following IP:

$$w = \max \{y(V) : y \in P_C \cap \mathbb{B}^n\}, \quad (5)$$

where polytope P_C is given by (2e) and

$$y(C_{T,e}) \leq |C_{T,e}| - \alpha_{C_{T,e}}, \forall T = (V, E_T) \in \mathcal{T}, \forall e \in E \setminus E_T. \quad (6a)$$

$$y_i \leq 1, i \in V. \quad (6b)$$

3 Branch-and-cut algorithm

Our Branch-and-cut algorithm (BC) first solves the relaxation $\max\{y(V) : (x, y) \in \overline{P}\}$, where \overline{P} is implied by (2a), (2c)-(2e) and (3). If its solution (\bar{x}, \bar{y}) is integer and implies an arborescence of D , it also solves DPSTP. Otherwise, we look for violated constraints (2b) and (4) to reinforce \overline{P} .

Cutset constraints (2b) are separated in $O(n^4)$ time complexity as follows. Assume that $\overline{D} = (V, \overline{A})$ ($\overline{A} := \{(i, j) \in A : \bar{x}_{ij} > 0\}$) is the support graph associated to (\bar{x}, \bar{y}) , the solution to the LP relaxation at hands. For each $i \in V \setminus \{r\}$, compute a minimum cut $A(V \setminus S, S) : r \in V \setminus S, i \in S$, that separates r and i in the network given by \overline{D} and arc capacities $\{\bar{x}_{ij} : (i, j) \in \overline{A}\}$. If $\bar{x}(A(V \setminus S, S)) < 1$, a cutset (2b) is violated and is appended in the relaxation.

For the separation of (4), we define edge weights $\{c_{ij} := \bar{x}_{ij} + \bar{x}_{ji} : \{i, j\} \in E\}$ and compute a maximum weight spanning tree $\overline{T} = (V, E_{\overline{T}})$ of G . \overline{T} is used to provide a valid lower bound on w and also to try to identify violated cuts (4), as follows. For each edge $e \in E \setminus E_{\overline{T}}$, we identify the set $C_{\overline{T},e}$. We initially set $\alpha_{C_{\overline{T},e}} = 2$ and then try to lift the cut: if $|E(C_{\overline{T},e})| \geq |C_{\overline{T},e}| + 1$, we

set $\alpha_{C_{\overline{T},e}} = 3$. If $(C_{\overline{T},e}, E(C_{\overline{T},e}))$ implies a clique of G , we set $\alpha_{C_{\overline{T},e}} = |C_{\overline{T},e}| - 1$. If $\bar{y}(C_{\overline{T},e}) > |C_{\overline{T},e}| - \alpha_{C_{\overline{T},e}}$, the violated cut is appended into the relaxation. The separation of (2b) and (4) is carried out until no violated cuts are found. At that moment, if the solution to the relaxation is not integer feasible, BC branches on variables.

4 Preliminary Computational Results

For each size $n \in \{30, 50, 100\}$, we generated DPSTP instances without cut-nodes nor bridges as follows. We first include in E edges in an Hamiltonian cycle of G : $\{\{1, 2\}, \{2, 3\}, \dots, \{n-1, n\}, \{1, n\}\}$. For the other pairs $i, j \in V$, we include the corresponding edge $\{i, j\}$ into E , according to a probability, defined by a desired graph density.

All algorithms were implemented in C and all computational results reported here were obtained with a Pentium 4 machine, running at 3.2 GHz, with 2GBytes of RAM memory. BC was implemented with calls to XPRESS Mixed Integer Programming solver (21.01.00) callback routines. The node selection policy was set to *best-first*. Apart from the heuristics, cut generation and pre-processing procedures that were turned off, all other default XPRESS settings were used.

For the computational results presented in Table 1, we have chosen $r = 1$. Assume that P_2 denotes the intersection of P_1 with inequalities (3) and those inequalities (4) found by our separation heuristic. In the table, we report on computational results for some selected instances in our test bed. In the first four columns, we provide an instance identifier (id), n , m and the implied graph density. In the next two columns, we provide two LP upper bounds: $w(P_1)$ and $w(P_2)$. In the last columns of the table, BC results are given: the number of BC nodes, the best lower (BLB) and best upper bound (BUB) found during the search and, finally, the CPU time (in seconds) needed to solve the instance. A time limit of one CPU hour was imposed.

As results in the table suggest, degree enforcing constraints (3) and cuts (4) significantly strengthened $w(P_1)$ bounds. For the instances considered in the table, our upper bounds are, on the average, 17% stronger than $w(P_1)$, the bounds behind the Branch-and-bound in [1]. Three instances could be solved at the root node, without branching, by matching the lower bounds provided by the DPSTP heuristic introduced here and the stronger LP upper bounds (rounded down). Despite the upper bound improvements reported here, some instances with $n = 100$ remained unsolved after one CPU hour.

Instance				LP Upper bounds		Branch-and-cut results			
id	n	m	dens (%)	$w(P_1)$	$w(P_2)$	nodes	BLB	BUB	$t(s)$
30_1	30	46	10.6	19.477	17.310	3	17	17	0.16
30_2	30	45	10.3	19.405	17.000	1	17	17	0.08
30_3	30	37	8.5	23.667	22.000	1	22	22	0.11
30_4	30	121	27.8	7.195	5.243	3	4	4	0.58
50_1	50	363	29.6	6.776	4.781	241	3	3	56.44
50_2	50	124	10.1	20.074	16.558	87	15	15	3.07
50_3	50	128	10.4	19.451	16.167	1	16	16	0.18
50_4	50	399	32.6	6.140	4.025	238	2	2	87.33
100_1	100	501	10.1	20.395	15.654	1954	12	14	3600
100_2	100	471	9.5	21.019	14.882	5515	11	13	3600
100_3	100	526	10.6	19.195	14.319	5133	11	12	3600
100_4	100	298	6.0	33.915	27.519	3061	25	25	957.06

Table 1

Branch-and-cut computational results for some selected instances. The same root $r = 1$ was chosen for all experiments.

5 Conclusions and Future Research

We addressed the Degree Preserving Spanning Tree Problem, for which Integer Programming formulations and valid inequalities were discussed. A Branch-and-cut algorithm was also introduced. Our computational results indicated that the bounds provided by the formulation behind our method are stronger than those in the literature.

We plan to proceed with the polyhedral investigation, attempting to characterize other valid inequalities. Another direction is to devise other strategies for strengthening cuts (4), when the sets $C_{\overline{T},e}$ found by the separation heuristic are small. One possible approach in that direction would be to use BC, in a recursive fashion, to evaluate optimal (or near optimal, through LP bounds) coefficients to strengthen the cuts.

To present another line of investigation, assume that binary decision variables $\{z_{ij} : \{i,j\} \in E\}$ are used to select the edges in the tree we are looking for. As before, given $E' \subseteq E$, define $z(E') := \sum_{\{i,j\} \in E'} z_{ij}$. DPSTP can be formulated as: $\max\{y(V) : (z, y) \in P_T \cap \mathbb{B}^{m+n}\}$, where P_T denotes the intersection of the spanning tree polytope $P_{STP} := \{z \in \mathbb{R}_+^m : z(E) = n-1, z(E(S)) \leq$

$|S| - 1, \forall S \subset V, S \neq \emptyset\}$ and

$$y_i \leq 1/|\delta_G(i)|z(\delta_G(i)), \forall i \in V. \quad (7)$$

It is not difficult to check that, at the LP relaxation of this DPSTP formulation, inequalities (7) must be tight. Replacing $y(V)$ in the objective function of such a relaxation by the sum of the right-hand-sides of (7) for all $i \in V$, DPSTP upper bounds could be obtained by solving the LP: $\max\{\sum_{\{i,j\} \in E}(1/|\delta_G(i)| + 1/|\delta_G(j)|)z_{ij} : z \in P_{STP}\}$. This bound should be weak but can be computed very quickly by Kruskal's algorithm. This observation may lead to a Branch-and-bound algorithm to DPSTP, similar to the one proposed by Fujie [3], for the Max-Leaf Spanning Tree Problem.

We also plan to investigate ways of using formulation (5) to devise other exact solution approaches for DPSTP.

References

- [1] Bhatia, R., Khuller, S. Pless, R., Sussmann, Y.S. The full-degree spanning tree problem. *Networks*, 36:203–209, 2000.
- [2] Broersma, H., Koppius, O., Tuinstra, H., Huck, A., Kloks, T., Kratsch, D., Müller, H. Degree-preserving trees. *Networks*, 35:26–39, 2000.
- [3] Fujie, T. An exact algorithm for the maximum-leaf spanning tree problem. *Computers and Operations Research*, 30:1931–1944, 2003.
- [4] Khuller, S., Bhatia, R., Pless, R. On local search and placement of meters in networks. *SIAM Journal on Computing*, 32:470–484, 2003.
- [5] Lokshtanov, D., Raman, V., Saurabh, S., Sikdar, S. On the directed full-degree spanning tree problem. *Discrete Optimization*, 8:97–109, 2011.
- [6] Magnanti, T.L., Wolsey, L. Optimal Trees. In O. Ball et al, editor, *Handbooks in OR and MS*, volume 7, pages 503–615. North-Holland, 1995.
- [7] Pothof, I.W.M., Schut, J. Graph-theoretic approach to identifiability in water distribution network. Memorandum 1283, Faculty of Applied Mathematics, University of Twente, Enschede, The Netherlands, 1995.

Solving the bi-objective prize-collecting Steiner tree problem with the ϵ -constraint method

Markus Leitner^{a,1} Ivana Ljubić^{b,2} Markus Sinnl^{b,3}

^a *Institute of Computer Graphics and Algorithms, Vienna University of Technology, Vienna, Austria*

^b *Department of Statistics and Operations Research, University of Vienna, Vienna, Austria*

Abstract

In this paper, we study the bi-objective prize-collecting Steiner tree problem, whose goal is to find a subtree that minimizes the edge costs for building that tree, and, at the same time, to maximize the collected node revenues. We propose to solve the problem using an ϵ -constraint algorithm. This is an iterative mixed-integer-programming framework that identifies one solution for every point on the Pareto front. In this framework, a branch-and-cut approach for the single-objective variant of the problem is enhanced with warm-start procedures that are used to (i) generate feasible solutions, (ii) generate violated cutting planes, and (iii) guide the branching process. Standard benchmark instances from the literature are used to assess the efficacy of our method.

Keywords: bi-objective combinatorial optimization, Steiner tree problem, ϵ -constraint method

¹ Supported by the Austrian Science Fund (FWF) under grant I892-N23. Email: leitner@ads.tuwien.ac.at

² Supported by the APART Fellowship of the Austrian Academy of Sciences. Email: ivana.ljubic@univie.ac.at

³ Email: markus.sinnl@univie.ac.at

1 Introduction. Problem definition

Many problems arising in the design of telecommunication, district heating, or water distribution networks, in forestry planning or even in image processing and system biology can be modeled as variants of the prize-collecting Steiner tree problem (PCSTP) (see e.g., [1] for a recent survey). The problem is defined as follows: Given an undirected graph $G = (V, E)$, with node revenues $r : V \rightarrow \mathbb{N}_0$, and edge costs $c : E \rightarrow \mathbb{N}$, find a subtree $G' = (V', E')$, $V' \subseteq V$, $E' \subseteq E$, such that $\sum_{v \in V'} r_v - \sum_{e \in E'} c_e$ is maximized. If node revenues and edge costs are measured using the same (e.g., monetary) units, this objective function corresponds to the net-worth maximization problem. However, difficulties arise if the “node revenues” are used to model some other aspects that cannot be so easily expressed using the monetary units. For instance, in wildlife corridor design (see [4]), the goal is to determine a subset of land parcels that connect areas of biological significance for a given species. In this case, nodes correspond to land parcels and their revenues correspond to habitats’ suitability. Typically, in such applications, the problem is modeled as a budget-constrained prize-collecting Steiner tree problem (see [4]), where the budget limits are provided by decision makers, and the collected revenues are maximized subject to the given budget. From the perspective of a decision maker it would be preferable to consider both objectives, namely, the cost minimization and the revenue maximization, as part of the optimization process. In that case, a decision maker would be able to easily perform a sensitivity analysis regarding the variations of the budget and the corresponding deviations from the total revenue. Therefore, we propose to solve the PCSTP as a bi-objective optimization problem. In this work we study an iterative mixed-integer-programming method known as the ϵ -constraint method for solving the bi-objective PCSTP (BOPCSTP).

Problem formulation

The set of all feasible subtrees will be modeled using a rooted Steiner arborescence model originally proposed in [6], see also [7]. Let T be the set of *terminal nodes*, (i.e., nodes with positive revenue), and $V \setminus T$ be the set of potential *Steiner nodes* (i.e., nodes with zero revenue). We create a directed graph $(V \cup \{0\}, A)$ with a root 0 as follows: $A := \{(i, j), (j, i) \mid \forall e = \{i, j\} \in E\} \cup \{(0, t) \mid \forall t \in T\}$, and set $c_{ij} = c_{ji} = c_e$ and $c_{0t} = 0$, for all $t \in T$. Using binary variables x_{ij} , for all $(i, j) \in A$ to indicate whether an arc is part of a solution and binary variables y_v , for all $v \in V$ to indicate if a vertex v is part of a solution, the following inequalities are used to model the set of all feasible

subarborescences of G rooted at 0:

$$\mathcal{P} = \{(x, y) \in \{0, 1\}^{|A|+|V|} \mid x(\delta^-(i)) = y_i, \forall i \in V, x(\delta^+(0)) = 1, x(\delta^-(W)) \geq y_k, \forall W \subseteq V, k \in W\},$$

where $\delta^+(W)$ and $\delta^-(W)$ denote the outgoing and incoming cutset, resp., for any $W \subseteq V$, and $x(A') := \sum_{(i,j) \in A'} x_{ij}$ for each $A' \subseteq A$. The BOPCSTP can now be defined as follows:

$$(\min \sum_{a \in A} c_a x_a, \max \sum_{v \in V} r_v y_v) \text{ subject to } (x, y) \in \mathcal{P}. \quad (1)$$

To optimize such an objective vector, we turn to the concept of Pareto optimality [5]. In bi-objective optimization, a solution is Pareto optimal, if there is no other feasible solution which is better in one objective and not worse in the other. The corresponding objective vector is called non-dominated. If all other feasible solutions are better in at most one of the objectives, the solution is called weakly Pareto optimal. The set of all Pareto optimal solutions is called efficient set and the set of all non-dominated vectors is called Pareto front. Note that the size of the Pareto front is at most as big as the size of the efficient set, since one point on the Pareto front can be achieved by more than one Pareto optimal solution. Our goal for the BOPCSTP is to find one solution for every point on the Pareto front.

2 ϵ -constraint method for the BOPCSTP

To find one solution for every point on the Pareto front, we use the ϵ -constraint method which is popular for solving bi-objective combinatorial optimization problems, see e.g., [3,8]. The method works by repeatedly solving ϵ -constraint problems, which are obtained by adding one of the objectives as constraint. In our approach, the second objective is relaxed and added as constraint. Notice that the same solution will be optimal for the maximization problem of the second objective and for the minimization problem $\min \sum_{v \in V} r_v(1 - y_v)$. This follows from the fact that the difference between the two objective values is $\sum_{v \in V} r_v$, which is a constant, and in the rest of the paper we will assume that the second objective function is given in minimization form. Hence, for a given $\epsilon \geq 0$, we obtain the following ϵ -constraint problem $P(\epsilon)$ for the BOPCSTP:

$$P(\epsilon) : \min \left\{ \sum_{a \in A} c_a x_a \mid (x, y) \in \mathcal{P}, \sum_{v \in V} r_v(1 - y_v) \leq \epsilon \right\}$$

Every optimal solution for a given value of ϵ is weakly Pareto optimal [5]. In the ϵ -constraint method, this fact, together with the integrality of the input, is used to systematically find one solution for every point on the Pareto front by decreasing ϵ by Δ , where Δ is the greatest common divisor of all p_v , $v \in T$. Since we chose the second objective as ϵ -constraint, the starting boundary point is the Pareto optimal point with the minimum possible value for the first objective. This point corresponds to a single-terminal solution without edges and with the terminal that has the maximal node revenue, i.e., $\epsilon = \sum_{v \in V} r_v - r^*$, where r^* is the maximal node revenue (ties are broken randomly). The value of ϵ is then decreased by Δ in each iteration. The algorithm terminates when the Pareto optimal point with the minimum possible value of the second objective is reached. This point corresponds to a solution which is the minimum cost Steiner tree connecting all terminals, i.e., $\epsilon = 0$. An overview of the algorithm is given in Algorithm 1. The Pareto optimal solutions are kept in the set Sol .

Algorithm 1 ϵ -constraint method for the BOPCSTP

```

 $Sol \leftarrow \emptyset; \quad r^* = \max_{v \in V} r_v; \quad \epsilon \leftarrow \sum_{v \in V} r_v - r^*$ 
while  $\epsilon \geq 0$  do
     $(x^*, y^*) \leftarrow \operatorname{argmin} P(\epsilon)$ 
     $Sol \leftarrow Sol \cup (x^*, y^*)$ 
     $\epsilon \leftarrow \sum_{v \in V} r_v(1 - y_v^*) - \Delta$ 
    Remove weakly Pareto optimal points from  $Sol$ 

```

The weakly Pareto optimal solutions found by the algorithm have the same cost but different revenue, since we are only minimizing the first objective. There are different ways to deal with this. Since preliminary tests showed that the number of discovered weakly Pareto optimal solutions is relatively small in our instances, we simply remove them in a postprocessing phase.

3 Solution framework and acceleration features

The proposed ϵ -constraint method heavily depends on the choice of the mixed-integer-programming formulation \mathcal{P} that is used to model feasible solutions, and on the solution method associated with it. Our implementation relies on the branch-and-cut approach from [7] for the single-objective PCSTP. In this section, we describe three acceleration features built in our solution framework that are enhancements of the approach from [7] with respect to the bi-objective optimization.

Our algorithm exploits information gained during the iterative ϵ -constraint framework (called “solution process” in the following): (i) to construct heuristic starting solutions, (ii) to define branching priorities, and (iii) to speed-up the generation of violated cutting planes. In the following, solving a single ILP by branch-and-cut will be denoted with “iteration”.

Constructing feasible solutions

Potential starting solutions are kept in a list L , which is indexed by the first objective and contains only non-dominated solutions. L is initialized with the non-optimal incumbent solutions of the first iteration. In the following iterations, L is populated in two ways: The first way consists of adding, for the Pareto optimal solution computed in the previous iteration, a terminal t not in this solution with minimum connection costs. If there are several terminals with the same minimum connection costs, one with the highest revenue is chosen. Moreover, non-optimal incumbent solutions are also added to L . The starting solution for the current iteration is chosen as follows: Let c^* be the cost of the Pareto optimal solution of the previous iteration and let Δ_c be the greatest common divisor of all c_e , $e \in E$. We define $c' := c^* + \Delta_c$. If the entry $L[c']$ exists and is a feasible solution, we use it, otherwise we continue increasing c' repeatedly by Δ_c until a feasible solution is found, or at most hundred entries have proceeded.

Guiding the branching process

For guiding the branching process, we consider two strategies in which different priorities are associated with variables. In the basic branching strategy (*BBS*), no information from the ϵ -constraint framework is used, i.e., branching is guided as for the single-objective PCSTP: the highest branching priority is associated with terminal-variables, followed by the priorities for variables of Steiner nodes, followed by the priorities for arc-variables. In the advanced branching strategy (*ABS*), we collect the information gained in the previous iterations to guide the branching process. In the first iteration, branching priorities are initialized as above. Every time a terminal or Steiner node occurs in an optimal solution of an iteration, the branching priority of the corresponding variable is increased by one for the following iteration.

Detecting violated cutting planes

The idea of cut pools for bi-objective MIP approaches is to store cuts from previous iterations and reuse them [9]. We implemented a cut pool for directed cutset constraints in the following way: During the first branch-and-cut iter-

ation, all violated cuts detected by the maximum-flow algorithm are stored in the cut pool. In each following iteration, violated cutset constraints are then separated as follows: (i) check, if there exist violated cuts in the cut pool created in the previous iteration (ii) add all these cuts and delete them from the current cut pool, in case no such cuts exist, call the maximum-flow separation routine. In either case, all detected violated cuts are inserted in the new cut pool that is used in the next iteration.

4 Computational results

The computational results have been obtained using a Intel Xeon X5500 with 2.67Ghz and 24GB RAM and CPLEX 12.4 as solver for the ILPs. As test instances we used the prize-collecting variant of the Steiner instance sets C and D from the OR-library [2]. Both the C and D sets contain 40 graphs divided into two groups, denoted by I and II. Instances in C have 500 nodes and between 625 and 12 500 edges, instances in D have 1 000 nodes and between 1 250 and 25 000 edges. The node revenues for terminals in the instances of type I are between 1 and 10 and in instances of type II between 1 and 100.

We tested the following four settings: “Basic” is a basic setting in which single-objective ϵ -constraint PCSTP is repeatedly solved, without taking the advantage of the bi-objective framework and learning from the previous iterations (i.e., initialization heuristics and cut pools are turned off and *BBS* is the branching strategy). Setting “ABS” is the Basic setting enhanced by the advanced branching strategy (*ABS*) and without initialization heuristics and cut pools. The setting “ABS+H” uses the initialization heuristics in addition to the *ABS* branching, and finally, in the setting “ABS+H+CP” all three bi-objective acceleration features are turned on (branching, heuristics and cut pools).

Figures 1a and 1b compare the four settings by showing boxplots of the running times for the 40 instances of the set C and D, respectively. The numbers shown in the boxes are the average running times over 40 instances, and the numbers shown above the boxes explain in how many out of 40 cases, we were not able to discover the complete Pareto front within the given time-limit. The timelimit was set to 1 800 and 7 200 seconds, for the set C and D, respectively. Regarding the set C, only with the setting “ABS+H+CP”, the complete Pareto front of all 40 instances could be discovered, moreover the results also indicate that this is the fastest among all approaches. Instance C5-II turned out to be the most difficult one and only the last setting has finished within the timelimit for this instance. This instance also has the biggest

Pareto front by far, with 1 462 points on it, the second biggest front is from instance C5-I and has size 1 056. The size of the Pareto front is clearly one of the factors that heavily influences the running time. Further factors that determine the running time are the number of terminals and the density of instances, as this is the case for the single-objective PCSTP. Not using the three acceleration features of our bi-objective framework clearly worsens the overall performance. The instances of group D turned out to be much more difficult to solve, and even with the fastest setting, for five instances (D5-II, D10-II, D15-II, D19-II and D20-II) we were not able to explore the complete Pareto front within two hours. Among all instances solved within the time-limit, instance D5-I has the biggest Pareto front with 2 160 points in it. Again, the runtime does not only depend on the size of the Pareto front, e.g., with the last setting, instance D4-II with a Pareto front of size 1 860 is solved in 3 130 seconds, while instance D19-II, with a Pareto front of size 537 needs 5 924 seconds. Instances with five or ten terminals are easy to solve, however, the runtime is up to five times higher than the one needed for instances of set C with five or ten terminals.

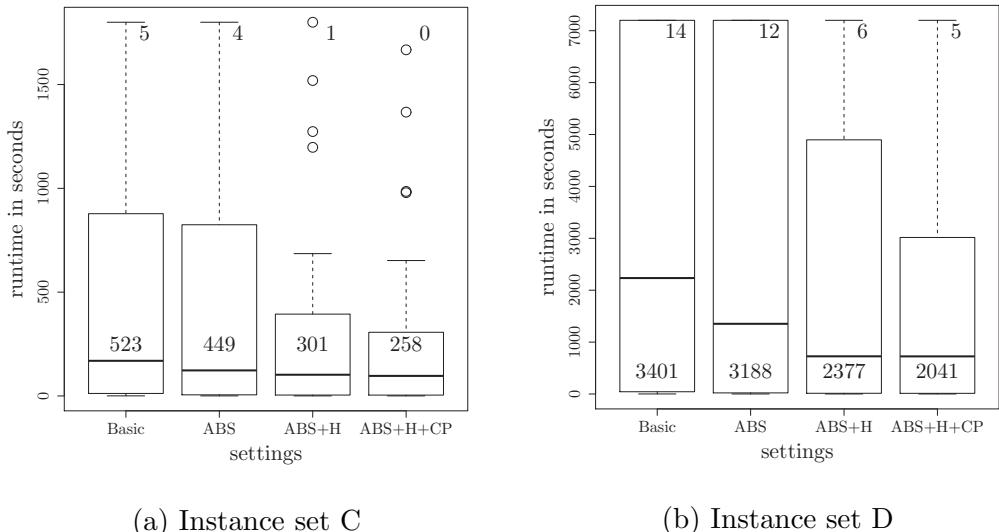


Fig. 1. Runtimes for the four settings of our bi-objective framework.

5 Conclusions and future work

In this paper, we introduced the bi-objective PCSTP and showed how to transform a single-objective branch-and-cut algorithm to an ϵ -constraint algorithm to find one solution for every point on the Pareto front. Moreover, three

acceleration techniques that exploit information obtained during the solution process are presented. Computational results show that these techniques are crucial to discover the Pareto front for larger instances. The largest Pareto front discovered has a size of 2160, the largest graph, for which the whole Pareto front has been found, has 1 000 nodes and 25 000 edges.

In a forthcoming paper, more potential acceleration techniques for the ϵ -constraint method will be explored, the method will be compared to other solution algorithms for bi-objective combinatorial optimization problems and the computational results will be analyzed in more detail.

References

- [1] Álvarez-Miranda, E., I. Ljubić and P. Toth, *Exact approaches for solving robust prize-collecting steiner tree problems*, submitted (2012).
- [2] Beasley, J. E., *OR-Library: Distributing Test Problems by Electronic Mail* (1990).
- [3] Bérubé, J.-F., M. Gendreau and J.-Y. Potvin, *An exact ϵ -constraint method for bi-objective combinatorial optimization problems: Application to the Traveling Salesman Problem with Profits*, European Journal of Operational Research **194** (2009), pp. 39–50.
- [4] Dilkina, B. and C. Gomes, *Solving connected subgraph problems in wildlife conservation*, in: *CRAIOP 2010*, Springer LNCS **6140**, 2010, pp. 102–116.
- [5] Ehrgott, M. and M. M. Wiecek, *Mutiojective Programming*, in: F. S. Hillier, editor, *Multiple Criteria Decision Analysis: State of the Art Surveys*, International Series in Operations Research & Management Science **78**, Springer New York, 2005 pp. 667–708.
- [6] Fischetti, M., *Facets of two Steiner arborescence polyhedra.*, Mathematical Programming **51** (1991), pp. 401–419.
- [7] Ljubić, I., R. Weiskircher, U. Pferschy, G. W. Klau, P. Mutzel and M. Fischetti, *An Algorithmic Framework for the Exact Solution of the Prize-Collecting Steiner Tree Problem*, Mathematical Programming **105** (2006), pp. 427–449.
- [8] Neumayer, P. and D. Schweigert, *Three algorithms for bicriteria integer linear programs*, OR Spectrum **16** (1994), pp. 267–276.
- [9] Riera-Ledesma, J. and J. Salazar-González, *The biobjective travelling purchaser problem*, European Journal of Operational Research **160** (2005), pp. 599–613.

WDM Fiber Replacement Scheduling¹

Andreas Bley, Daniel Karch²

*COGA Group, TU Berlin
Berlin, Germany*

Fabio D'Andreagiovanni³

*Department of Optimization, Zuse-Institut Berlin (ZIB)
Berlin, Germany*

Abstract

We consider the migration of a WDM telecommunication network to a new technology. In the course of the migration process, shared network resources must be temporarily shut down, affecting the network connections that use them. In this paper we describe an ILP-based approach to find a migration schedule that minimizes the total service disruption occurring in the network.

Keywords: Network Optimization, WDM, Integer Linear Programming

1 Introduction

A critical challenge for telecommunication networks is standing the pace of the rapidly evolving technological context, in which new technologies and updates

¹ This work was partially supported by the *German Federal Ministry of Education and Research* (BMBF), project *ROBUKOM* [1,2], grant 03MS616E.

² {bley,karch}@math.tu-berlin.de

³ d.andreagiovanni@zib.de

of technological standards are frequently made available. Service providers cannot neglect such advances, as these provide drastic increases in operational efficiency. However, the implementation of technological upgrades is not a trivial task: new hardware components need to be installed in the network, and during the installation network connectivity may be compromised. The Wavelength Division Multiplexing (WDM) technology, which we take as the basis for our problem, shares links among several connections, and tearing down a single link might affect several connections at once. When the upgrades involve large parts of the network, not all operations can be done in parallel, as the number of available technicians is limited. A bad scheduling of the endeavor can thus dramatically increase the disconnection time of parts of the network, causing extended service disruption. In this work, we study the problem of defining a schedule of network upgrades that minimizes the total service disruption time. To the best of our knowledge, this problem has not yet been investigated. The aim of our work is to close this gap. Our work has also been driven by real needs of the *German National Research and Education Network* (DFN). Specifically, DFN is interested in optimizing the migration of the Germany-wide fixed network, in order to contain service disruption.

We formally define the problem in Section 2, and give an ILP model in Section 3. To strengthen the model, we consider lower bounding techniques and an extended formulation in Sections 4 and 5, respectively. In Section 6 we present computational results.

2 The Network Migration Problem

A telecommunication network can be essentially described as a set of links connecting a set of nodes. We consider the problem of updating the technology installed on the links: each link is based on an old technology A and is to be updated to a new technology B , within a time horizon $[T] = \{0, 1, \dots, T\}$ of elementary time periods. The network is modeled as a graph $G(V, E)$, with V denoting the set of nodes and E the set of links, where a link $e \in E$ is a two-element subset of V , e.g. $e = \{i, j\} \subseteq V$. Updating a link $e \in E$ requires two technicians operating at the end nodes of the link, plus a number $\alpha_e \in \mathbb{N}$ of additional technicians, depending on the length of the link, the presence of signal regenerators etc. There needs to be at most one worker present at a node, regardless of the number of incident links that are scheduled at that time. The number of updates scheduled in a period of the migration horizon is limited by the number K of available technicians. Formally, a link subset $F \subseteq E$ may only be upgraded in a single period if $\sum_{e \in F} \alpha_e + |V(F)| \leq K$.

A set P of fixed paths is given that models the connections realizing the users' traffic demands. A path p is *active*, when all its links are based on the same technology, i.e., either A or B . When at some point in time, p contains links in technology A , as well as links in technology B , the path is *disrupted*.

The *WDM Fiber Replacement Scheduling Problem (WDM-FRS)* consists in finding a schedule $S : E \rightarrow [T]$, that maps each link $e \in E$ to a time $S(e) \in [T]$, so that all links are migrated at time T , the work budget is respected at all times, and the overall service disruption is minimized.

3 A Time-Indexed ILP Formulation

Our model for WDM-FRS is based on time-indexed decision variables. We assume that we can migrate at least one link per period, otherwise the problem is infeasible. Hence, $|E|$ is an upper bound for the length of any optimal schedule. We can derive another upper bound by considering the amount of work that is performed in consecutive periods. If the work in two consecutive periods sums up to at most K , the two periods can be collapsed into one, without increase in disruption. Thus the average amount of work per period must be at least $(K + 1)/2$, and if W is an upper bound for the total work that has to be performed, $2W/(K + 1)$ is an upper bound for the length of the schedule. We use $T := \min \{|E|, \lfloor 2W/(K + 1) \rfloor\}$ periods in our model, where we set $W := \sum_{e \in E} \alpha_e + 2|E|$, taking into account that we might not be able to migrate coincident links in the same period, in which case we have to pay for a node in every period an incident link is scheduled. The problem of finding a migration schedule minimizing the total disruption can then be formulated as an integer program, as given in Figure 1 on the following page.

We use four groups of binary variables in the model: $x_e^t = 1$, iff link e is based on technology B at the end of period t , $y_p^{tC} = 1$, iff path p is active on technology $C \in \{A, B\}$ at the end of period t , $z_i^t = 1$, iff $i \in (V \cup E)$ is being worked on in period t , and $d_p^t = 1$, iff path p is disrupted at the end of period t . The objective function (1) minimizes the disruption that is experienced throughout the migration. The constraints (2) express that a path is either active in one of the two technologies, or it is disrupted. Constraints (3) and (4) express the fact that in the first period, $t = 0$, all the links are based on the old technology A and must be migrated to the new technology B by the last period of the time horizon. The constraints (5) and (6) establish the relation between the link technology variables x_e^t of a path p and the activation variables $y_p^{t\sigma}$ of this path. Specifically, a path p is active if and only if all of its fibers are based on the same technology. The constraints (7) ensure that the work

$$\begin{aligned}
& \min \sum_{p \in P} \sum_{t \in [T]} d_p^t && (1) \\
\text{s.t. } & d_p^t + y_p^{tA} + y_p^{tB} = 1, & p \in P, t \in [T], & (2) \\
& x_e^0 = 0, & e \in E, & (3) \\
& x_e^T = 1, & e \in E, & (4) \\
& y_p^{tA} \leq 1 - x_e^t, & p \in P, e \in p, t \in [T], & (5) \\
& y_p^{tB} \leq x_e^t, & p \in P, e \in p, t \in [T], & (6) \\
& x_e^t - x_e^{t-1} = z_e^t, & e \in E, 0 < t \in [T], & (7) \\
& z_e^t \leq z_i^t, & e \in E, i \in e, 0 < t \in [T], & (8) \\
& \sum_{i \in V \cup E} \alpha_i z_i^t \leq K, & 0 < t \in [T]. & (9) \\
& x_e^t, y_p^{tC}, z_e^t, d_p^t \in \{0, 1\}, & e \in E, p \in P, & (10) \\
& t \in [T], C \in \{A, B\}.
\end{aligned}$$

Fig. 1. A time-indexed ILP formulation for WDM-FRS.

indicator variables z^t are set to 1 for all the links that are upgraded during period t , while the inequalities (8) require that a link can only be migrated when there are workers present at its end nodes at the same time. Finally, the budget constraints (9) ensure that the number of technicians required for the upgrades does not exceed the given budget at any time.

As is, the above formulation for WDM-FRS has an extremely weak LP relaxation that allows for trivial solutions without any service disruption: A fractional solution can migrate a fraction of $1/T$ of each link $e \in E$ in each period $t \in [T]$ by setting $x_e^t = t/T$, leading to no service disruption at all, and thus leaving us with a useless bound. This in turn makes the solution of larger problems hopeless and causes the generation of huge branch-and-bound trees. There is a simple remedy though: The LP cannot spread the work evenly throughout the time horizon once we fix the migration of a single fiber to a particular period, i.e., if we fix $z_e^t = 1$ for some $e \in E, t \in [T]$. When e has to be migrated in period t , links that have a path in common with e will likely be placed close to e in a good schedule. To see that we can perform such a fixing without loss of generality, consider the following simple observation:

Proposition 3.1 *The value and feasibility of a schedule are invariant under inversion and translation, i.e., if S is a feasible schedule, then*

$$S^R : E \rightarrow [T], \quad \text{and} \quad S^\tau : E \rightarrow [T] \\ e \mapsto T + 1 - S(e), \quad e \mapsto S(e) + \tau,$$

are feasible schedules, provided that $S(e) + \tau \in [T]$ for each link $e \in E$.

If we fix $z_e^t = 1$, we have to make sure that the time horizon is long enough to schedule the remaining links. Without loss of generality we can assume that not more than $T/2$ periods $t' < t$ are used in an optimal schedule (or else, we could reverse the schedule). Hence, we set $t = \lceil T/2 \rceil$. By extending the time horizon to $T' = \lceil 3T/2 \rceil$, we assure that there is enough time to schedule all remaining links. In fact, our experiments show that the LP bound of the altered model is quite usable as a starting point for a branch-and-bound phase. The choice of the fixed link affects the LP bound. Our experiments suggest that it is reasonable to choose a link that belongs to many paths.

4 Obtaining Lower Bounds

To speed up the branch-and-bound process, we would like to start it with a good lower bound on the overall disruption. To this end, we add valid inequalities of the form $\sum_{p \in M} \sum_{t \in [T]} d_p^t \geq l(M)$ to the model, where M is a set of paths, and $l(M)$ is a lower bound for WDM-FRS restricted to M .

The simplest case to consider is when $M = \{p\}$ contains only a single path. Clearly, to migrate p , we have to migrate all its links and have to work on each node on p at least once. Hence, $\alpha(p) := \sum_{e \in p} \alpha_e + |V(p)|$, is a lower bound on the work that has to be performed to migrate p , and

$$\sum_{t \in [T]} d_p^t \geq \lceil \alpha(p)/K \rceil - 1$$

is a valid inequality. However, we can do better. Assigning the links to periods with a constant budget of K is similar to a *bin packing* problem, with the difference that a node v has to be packed twice, if its incident links on p , $\{u, v\}$ and $\{v, w\}$, are not assigned to the same period. If, for each node u , we assign its cost arbitrarily to one of its incident links, we obtain a bin packing problem that is a relaxation of the original problem. Let α'_e be the cost of link e in the bin packing instance. Note that $\sum_{e \in p} \alpha'_e = \alpha(p)$, since the node costs are already included in the new link costs. Therefore, $L_1 := \lceil \alpha(p)/K \rceil$ is a lower bound also for the number of needed bins. The worst-case ratio of this

lower bound is $1/2$, meaning that the optimal number of bins is at most twice L_1 . There exist stronger bounds, such as the bound L_2 given in [3], which is computable in time proportional to $|p|$ and has a worst-case ratio of $2/3$.

Now suppose that L is a lower bound on the number of periods needed to migrate p . Clearly, this means that the links on the path need to be partitioned into at least L subsets. Then, at least $L - 1$ nodes have to be worked on in two different periods. The cost of those nodes therefore has to be paid twice. The actual work that has to be performed is then at least $\alpha(p) + L - 1$. We obtain a lower bound on the migration time by adding $L - 1$ new items of weight 1 to the bin packing instance. If L' is a lower bound on this altered instance, $\sum_{t \in [T]} d_e^t \geq L' - 1$ is a valid inequality for our model.

It is also possible to obtain lower bounds from path sets containing more than one path. Let $M = \{p_1, \dots, p_m\}$, and let $p^1 := p$, $p^{-1} := (\bigcup_{q \in M} q) \setminus p$. The set

$$N := \left\{ \bigcap_{i=1}^m p^{v_i} \mid v \in \{-1, 1\}^m \right\} \setminus \{\emptyset\},$$

is a partitioning of the links in M , e.g., if $M = \{p, q\}$, then

$$N = \{(p^1 \cap q^{-1}), (p^1 \cap q^1), (p^{-1} \cap q^1)\} = \{(p \setminus q), (p \cap q), (q \setminus p)\}.$$

We consider a time-continuous relaxation of WDM-FRS, that defines the disruption of a path p as $\text{disr}(p) := \max\{0, r_p - l_p - 1\}$, where l_p and r_p are the start time and completion time for path p , respectively. It is not too difficult to show that there is always an optimal time-continuous schedule that schedules the sets in N in a non-preemptive fashion, e.g., in the previous example, the schedule might first work on the links in $(p \setminus q)$, then on the intersection $(p \cap q)$, and finally on $(q \setminus p)$. In general, we can assume that an optimal time-continuous schedule is essentially a permutation of the elements of N . When there are at least three paths in M , such a schedule will most of the time work on some $I \in N$, while some path p is not functional, where $p \cap I = \emptyset$. Using this observation we can derive valid inequalities of the form

$$\sum_{p \in M} \sum_{t \in [T]} d_p^t \geq \left\lceil \sum_{p \in M} \alpha(p)/K + \kappa_p \right\rceil - |M|,$$

where κ_p is the additional disruption on path p that arises while p is inactive, and work is done on sets $I \in N$ that do not intersect with p , i.e., $p \cap I = \emptyset$. Finding correct values for κ_p has been proven quite tricky, and any formula that takes the sets in N explicitly into account is not likely to be useful. However, we have computed lower bounds for the case $|M| = 3$, and they are very useful to increase the lower bound at the root node.

5 Betweenness Variables

Motivated by the results in [4], we considered the extension of our model by a set of so-called *betweenness variables* μ_{efg} , representing the situation that link f is migrated between e and g . We use the new variables to impose a strict linear ordering \prec on the links. In our case, $\mu_{efg} = 1$, if for the migration times e, f , and g , either $e \prec f \prec g$, or $g \prec f \prec e$ holds. I.e., $\mu_{efg} = 1$ implies that $S(e) \leq S(f) \leq S(g)$ or $S(e) \geq S(f) \geq S(g)$. We model the consistency between the betweenness variables as explained in [4], and couple them to our original problem variables via the constraints

$$x_e^t - x_f^t + x_g^t \leq 2 - \mu_{efg}, \quad t \in [T], \{e, f, g\} \subset E.$$

The above constraints express that, if e and g are active in technology B at time t and f is not, f cannot lie between e and g in the linear ordering.

The new variables allow us to express stronger conditions for the model, but the number of introduced constraints and variables is rather large. We decided not to enforce integrality on the betweenness variables, and to use Benders' decomposition to handle all constraints involving them in a client LP to generate cuts for the master ILP.

6 Computational Results

We have implemented the model described in this paper using SCIP 3.0 [5] with CPLEX 12.4 [6] as LP solver. All experiments have been conducted running on a single core of a PC with an AMD Phenom II X6 1090T processor and 8GB of RAM. The model implemented is the one described in Section 3 with the valid inequalities from Section 4 added. We report both the results with and without the cuts from Benders' decomposition described above.

Table 1 lists the problem instances that we have considered in our experiments. We report the lower bound after the root node, the value of the solution, the number of nodes in the branch and bound tree, and the time needed. The BC column indicates how many Benders cuts were applied in the root node. All instances are subnetworks of the “full” instance, comprising 64 nodes, 84 links, and 105 paths. The addition of the Benders cuts seems advisable once the instances get bigger. Yet, a decrease in the number of branch-and-bound nodes can be observed in all instances. Especially the bigger instances benefit immensely from the stronger relaxation in the root node, and only half as many nodes are explored.

Instance	V	E	P	BC	root	sol	nodes	time
Han10	10	10	32	0	49.0	62	215	00:00:04.91
Han10	10	10	32	66	51.0	62	179	00:00:07.34
Fra20	20	22	51	0	45.7	59	409	00:00:41.11
Fra20	20	22	51	169	47.0	59	352	00:00:48.78
Stu25	25	28	52	0	78.2	135	15314	00:29:45.51
Stu25	25	28	52	628	84.6	135	8334	00:20:20.59
Des30	30	37	60	0	75.2	140	23976	01:04:58.17
Des30	30	37	60	555	81.8	140	11799	00:40:59.47

Table 1

7 Further Work

The integrality gap of our model is still significant, leaving a lot of room for improvement, be it through additional valid inequalities for our model or through a different model. To close that gap, specialized branching schemes need to be investigated. We have begun implementing several heuristics with encouraging results. In practice, finding tight lower bounds seems to be much more difficult than finding good (i.e. near-optimal) solutions.

References

- [1] Bley, A., D’Andreagiovanni, F., Hanemann, A.: Robustness in Communication Networks: Scenarios and Mathematical Approaches. In: Proc. of the ITG Symposium on Photonic Networks 2011, pp. 10–13. VDE Verlag, Berlin (2011)
- [2] Koster, A.M.C.A., Helmberg, C., Bley, A., Grötschel, M., Bauschert, T.: BMBF Project ROBUKOM: Robust Communication Networks. In: ITG Workshop Euro View 2012, pp. 1–2, VDE Verlag, Berlin (2012)
- [3] Martello, S., and Toth, P., *Knapsack Problems: Algorithms and Computer Implementations*, Wiley-Interscience series in discrete mathematics and optimization (1990).
- [4] Caprara, A., Oswald, M., Reinelt, G., Schwarz, R., Traversi, E.: *Optimal linear arrangements using betweenness variables*, Math. Prog. C, **3**, 261–280 (2011)
- [5] SCIP Optimization Suite 3.0.0, <http://scip.zib.de>
- [6] IBM ILOG CPLEX Optimizer 12.4, <http://www.ibm.com/software/integration/optimization/cplex-optimizer/>

Integer programming models for maximizing parallel transmissions in wireless networks

Yuan Li^{a,1}, Michał Pióro^{a,b,2}

^a *Department of Electrical and Information Technology
Lund University, Lund, Sweden*

^b *Institute of Telecommunications
Warsaw University of Technology, Warsaw, Poland*

Abstract

In radio communications, a set of links that can transmit in parallel with a tolerable interference is called a compatible set. Finding a compatible set with maximum weighted revenue of the parallel transmissions is an important subproblem in wireless network management. For the subproblem, there are two basic approaches to express the signal to interference plus noise ratio (SINR) within integer programming, differing in the number of variables and the quality of the upper bound given by linear relaxations. To our knowledge, there is no systematic study comparing the effectiveness of the two approaches. The contribution of the paper is two-fold. Firstly, we present such a comparison, and, secondly, we introduce matching inequalities improving the upper bounds as compared to the two basic approaches. The matching inequalities are generated within a branch-and-cut algorithm using a minimum odd-cut procedure based on the Gomory-Hu algorithm. The paper presents results of extensive numerical studies illustrating our statements and findings.

Keywords: branch-and-cut, matching polytope, Gomory-Hu algorithm, SINR.

¹ Email: yuan.li@eit.lth.se

² Email: michal.pioro@eit.lth.se

1 Introduction

In radio communications, a compatible set (CS) is defined as a group of radio links that can be simultaneously active (i.e., transmit in parallel) so that the signal to interference plus noise ratio (SINR) constraint is satisfied for each active link. The use of CS is a powerful approach for optimizing transmission scheduling, routing, power control and traffic throughput in wireless networks [3,5,13]. In [11], a max-min fair flow allocation integer programming problem for wireless mesh networks is considered where the CS subproblem is embedded as a pricing problem in a branch-and-price algorithm. Our previous work [10] develops a heuristic to generate a sub-optimal solution consisting of several CSs for a joint link rate assignment and transmission scheduling problem.

Finding a CS with a maximum weighted revenue from the transmitting links (we will refer to this problem as MWCSP - maximum weighted CS problem) has attracted a considerable attention. For example, [6,7] study the complexity of MWCSP and demonstrate its \mathcal{N} -hardness. Further, [2,8] present heuristic and approximate algorithms for MWCSP while [1] presents a distributed algorithm based on game theory. As pointed out in [4], there is not much work on finding exact solutions for MWCSP. The authors present such an algorithm based on knapsack lifted cover inequalities for maximizing the cardinality of a compatible set (link weights are all equal to 1). However, as noted in [4], their algorithm cannot be applied for the general case of MWCSP.

In this paper, we firstly summarize two basic integer programming models for MWCSP. Then, we strengthen the presented MWCSP models using the results of matching theory, and make numerical comparisons. The numerical study shows the linear relaxation of the enhanced model gives an improvement in terms of the upper bound. Embedding the resulting cut generation procedure to branch-and-bound results in an efficient branch-and-cut algorithm.

2 Problem description

2.1 Network model and notations

A wireless network can be represented as a directed graph $\mathcal{H} = (\mathcal{V}, \mathcal{A})$ with the set of nodes \mathcal{V} and the set of directed radio links (arcs) \mathcal{A} . The head and tail node of an arc $a \in \mathcal{A}$ is represented as $s(a)$ and $t(a)$, respectively, so for arc $a = (v, w)$ ($v, w \in \mathcal{V}$) we have $s(a) = v$ and $t(a) = w$. We assume that graph \mathcal{H} is bi-directed, i.e., if arc $a = (v, w)$ is provided ($a \in \mathcal{A}$) then so is the opposite arc $a' = (w, v)$ ($a' \in \mathcal{A}$). The set of all arcs incident to node v is denoted by $\delta(v)$. The sets $\delta^+(v)$ and $\delta^-(v)$ denote the outgoing arcs and the

incoming arcs of v , respectively, so that $\delta(v) = \delta^+(v) \cup \delta^-(v)$.

Let \mathcal{C} ($\mathcal{C} \subseteq \mathcal{A}$) denote a set of arcs that can be active simultaneously. Then the arcs in \mathcal{C} must satisfy the following SINR condition:

$$N + \sum_{v \in \mathcal{V}_\mathcal{C} \setminus \{s(a)\}} P_{vt(a)} \leq \frac{1}{\gamma} P_{s(a)t(a)}, \quad a \in \mathcal{C}. \quad (1)$$

In (1), $\mathcal{V}_\mathcal{C}$ denotes the set of nodes active in \mathcal{C} : $\mathcal{V}_\mathcal{C} = \{s(a) : a \in \mathcal{C}\}$, P_{vw} represents the power received by node w when node v is transmitting, N is the noise power, and γ is the SINR threshold.

We assume that a node v can either transmit or receive, and only one link incident to v can be active as expressed by the inequality:

$$|\mathcal{C} \cap \delta(v)| \leq 1, \quad v \in \mathcal{V}. \quad (2)$$

2.2 Problem formulations

A subset \mathcal{C} of \mathcal{A} is called a *compatible set* (CS) if it satisfies constraints (1) and (2). MWCSP consists in finding a CS \mathcal{C} that maximizes $\sum_{a \in \mathcal{C}} c_a$ for the given arc weights $c_a \geq 0, a \in \mathcal{A}$. Our model for MWCSP is based on binary variables $Y = (Y_a, a \in \mathcal{A})$ ($Y_a = 1$ if $a \in \mathcal{C}$) and $X = (X_v, v \in \mathcal{V})$ ($X_v = 1$ if v is active):

$$\max \quad \sum_{a \in \mathcal{A}} c_a Y_a \quad (3a)$$

$$\sum_{a \in \delta(v)} Y_a \leq 1 \quad v \in \mathcal{V} \quad (3b)$$

$$\sum_{a \in \delta^+(v)} Y_a = X_v \quad v \in \mathcal{V} \quad (3c)$$

$$Y_a (N + \sum_{v \in \mathcal{V} \setminus \{s(a)\}} P_{vt(a)} X_v) \leq \frac{1}{\gamma} Y_a P_{s(a)t(a)} \quad a \in \mathcal{A} \quad (3d)$$

$$Y \text{ binary, } X \text{ continuous.} \quad (3e)$$

In the model at most one arc a incident to node v is active (constraint (3b)) and v is active only when one outgoing arc is active (constraint (3c)). Constraint (3d) represents the SINR condition. It must be obeyed only when a is active, i.e., when $Y_a = 1$. Variables X are forced to be binary by (3b) and (3c). The resulting CS is $\mathcal{C} = \{a \in \mathcal{A} : Y_a^* = 1\}$ where Y^* is an optimal solution of (3).

Constraint (3d) is nonlinear. There are (at least) two ways to linearize it [4,11]. One is adding variables Z_{av} that express the product $Y_a \cdot X_v$:

$$Z_{av} \geq Y_a + X_v - 1, \quad a \in \mathcal{A}, v \in \mathcal{V} \quad (4a)$$

$$Z_{av} \leq Y_a, \quad Z_{av} \leq X_v, \quad Z_{av} \geq 0, \quad a \in \mathcal{A}, v \in \mathcal{V} \quad (4b)$$

$$N Y_a + \sum_{v \in \mathcal{V} \setminus \{s(a)\}} P_{vt(a)} Z_{av} \leq \frac{1}{\gamma} P_{s(a)t(a)} Y_a, \quad a \in \mathcal{A} \quad (4c)$$

where Y_a and X_v are binary. Constraints (4a) and (4b) imply that $Z_{av} = Y_a X_v$, i.e., $Z_{av} = 1$ if both Y_a and X_v are equal 1, and 0 otherwise. The Z -model is obtained when (3d) in model (3) is replaced by (4a), (4b) and (4c).

Another way is using a “big M” to express the SINR condition.

$$\gamma(N + \sum_{v \in V \setminus \{s(a)\}} P_{vt(a)} X_v) \leq P_{s(a)t(a)} + M_a(1 - Y_a), \quad a \in \mathcal{A} \quad (5)$$

where $M_a = \gamma(N + \sum_{v \in V \setminus \{s(a)\}} P_{vt(a)}) - P_{s(a)t(a)}$ is a constant. Note that when $Y_a = 1$ the inequality (5) expresses the SINR condition for link $a \in \mathcal{A}$. For $Y_a = 0$, the inequality holds trivially due to the use of M on the right-hand side of (5). Replacing constraint (3d) in model (3) by (5) results in the M -model.

As illustrated in Section 4, inequality (4c) is stronger than inequality (5) in terms of upper bound for the corresponding linear relaxations. That is, the linear relaxation of the Z -model gives a better (i.e., lower) upper bound than the linear relaxation of the M -model.

3 Reformulation based on matching polytope

Recall that the network graph $\mathcal{H} = (\mathcal{V}, \mathcal{A})$ in previous models is bi-directed, i.e., if $(v, w) \in \mathcal{A}$, then also $(w, v) \in \mathcal{A}$. Now consider the undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with the undirected edges (links) $e \in \mathcal{E}$ corresponding to the bi-directed arcs of \mathcal{A} in original graph \mathcal{H} . Then, the two oppositely directed arcs corresponding to edge $e \in \mathcal{E}$ will be denoted by $a'(e)$ and $a''(e)$. We observe that constraints (3b) and (3e) (arc-variables Y are binary) imply that any compatible set is a matching in \mathcal{G} (see [9]) that satisfies the SINR constraints.

Below we shall extend the Z -model by adding matching inequalities. This, as illustrated in Section 4, will improve the upper bound.

Let $x = (x_e, e \in \mathcal{E})$ be given binary variables associated with the edges of the undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. For any $F \subseteq \mathcal{E}$, let $x(F)$ denote the sum $\sum_{e \in F} x_e$. A subset of edges $\mathcal{C}(x) = \{e \in \mathcal{E} : x_e = 1\}$ is called a *matching* when $|x(\delta(v))| \leq 1$ for all $v \in \mathcal{V}$. A basic result of the matching theory states that the convex hull of all matchings x of \mathcal{G} is defined by the so called Edmond’s matching polytope as follows [9]:

$$\sum_{e \in \delta(v)} x_e \leq 1 \quad v \in \mathcal{V} \quad (6a)$$

$$x(E[U]) \leq \frac{|U| - 1}{2} \quad U \subseteq \mathcal{V}, |U| > 1 \text{ and odd} \quad (6b)$$

$$x_e \geq 0 \text{ and continuous} \quad e \in \mathcal{E} \quad (6c)$$

where $E[U]$ denotes the set of all edges in \mathcal{E} with both ends in U .

Adding inequalities (6) (inequalities (6b) are called matching inequalities) to the Z -model, we obtain the following ZC -model.

$$\max \quad \sum_{a \in A} c_a Y_a \quad (7a)$$

s.t. (6a), (6b), (3c), (4a), (4b), (4c)

$$Y_{a'(e)} + Y_{a''(e)} \leq x_e \quad e \in \mathcal{E} \quad (7b)$$

$$Y \text{ binary, } x, X, Z \text{ continuous, } x, Z \geq 0 \quad (7c)$$

Certainly, the number of matching inequalities (6b) in the C -model is exponential in $|\mathcal{V}|$ and therefore, in the process of solving the linear relaxation of (7), we need to generate them on-the-fly. For that, the minimum odd-cuts with respect to x in graph \mathcal{G} , generated from the Gomory-Hu tree (see [9]), are used. For the lack of space we are not able to present the procedure of generating the most violated matching inequality (6b) for a given x .

Finally, we note that it is easy to add the matching inequalities to the M -model by replacing (4a), (4b), (4c) in the ZC -model with (5). The resulting model is called the MC -model.

4 Numerical study

In this section we compare the effectiveness of the four introduced MWCSP models M , Z , MC and ZC using computational experiments. The nodes in example networks are located randomly in a square of 500×500 meters. The transmitting power for each node is $P_T = 0.1$ W. The received power is calculated as $P_R = P_T * d^{-4}$ W where d is the transmitter-receiver distance. The noise power is set to $N = 10^{-13}$ W and the SINR threshold is $\gamma = 3.5$. The weights c for objective (3a) are generated randomly between 0 and 1.

We used the implementation of the Gomory-Hu tree algorithm from [12], and embedded it in our matching inequalities generating procedure. The algorithms were implemented in Python. For linear relaxations, the Gurobi solver was used. All the computations were executed on a Windows XP computer equipped with a dual core Intel 2.53 GHZ CPU and 1.93G RAM.

4.1 Comparison of the quality of linear relaxations

Table 1 shows running times and optimality gaps for the linear relaxations of the MWCSP models. The optimality gap of a particular linear relaxation is defined as $\frac{\text{LRO} - \text{MIP}}{\text{MIP}} 100\%$, where MIP denotes the optimal objective of (3) (obtained by solving the M -model using Gurobi) while LRO is the opti-

Table 1
Strength of linear relaxations of the MWCSP models

network	<i>M</i> -model		<i>Z</i> -model		<i>MC</i> -model			<i>ZC</i> -model			
	$ \mathcal{V} $	$ \mathcal{A} $	gap(%)	time (s)	gap(%)	time (s)	gap(%)	time (s)	cuts	gap(%)	time (s)
30	202	42.66	0.15	39.93	1.24	41.18	1.05	14	27.71	9.87	7
	226	81.78	0.12	75.30	1.35	80.65	0.58	4	42.71	4.24	3
	246	66.26	0.13	62.06	1.51	65.41	0.64	6	38.19	8.25	5
40	382	74.24	0.27	69.65	6.95	73.73	1.69	22	40.63	29.91	7
	374	69.09	0.26	64.28	7.08	68.71	1.65	9	39.09	25.99	9
	366	85.27	0.25	79.19	3.29	84.10	2.33	11	43.93	28.33	7
50	524	74.11	0.45	69.10	20.50	73.88	4.04	14	40.76	56.03	4
	576	69.79	0.49	65.91	18.78	68.04	4.11	18	39.42	80.92	6
	568	84.28	0.53	79.16	21.67	83.87	4.04	11	44.05	60.32	3
60	780	87.14	0.80	81.84	25.41	86.73	7.72	18	44.84	303.91	12
	744	91.70	0.88	86.53	60.60	91.63	5.03	9	46.37	77.09	2
	800	88.24	0.87	82.49	65.05	87.96	10.91	17	45.18	127.29	4
average	76.21	0.44	71.29	21.53	75.49	3.65	12.75	41.07	67.32	6.5	

Table 2
Effectiveness of BBM, BBZ, BCM, BCZ

network	BBM			BBZ			BCM			BCZ		
	$ \mathcal{V} $	$ \mathcal{A} $	time(s)	B&B	time(s)	B&B	time(s)	B&B	cuts	time(s)	B&B	cuts
15	38	4	203	12	160	3	131	8	9	101	9	
	32	2	110	4	63	1	20	5	1	18	4	
	46	7	285	11	116	6	211	14	9	90	14	
20	64	18	852	20	128	22	494	16	21	107	12	
	86	435	8981	293	1337	313	5584	56	124	536	36	
	90	112	2385	105	431	84	1480	50	85	367	40	
25	130	1908	22805	831	2087	1528	12750	212	565	1247	106	
	168	2786	26051	1526	2716	6730	25950	806	1214	2218	217	
	160	2701	26503	1578	2912	3289	18819	567	1544	2314	179	
30	205	3h*	71131	7071	8754	3h*	49687	678	3850	4094	207	
	226	3h*	68154	3h*	14117	3h*	47856	554	7673	9612	310	
	248	3h*	58461	3h*	11238	3h*	34725	487	3h*	10770	541	

imum objective of the linear relaxation. Column “cuts” shows the number of generated matching inequalities needed to obtain the optimum. The *Z*-model is better (has smaller gaps) than the *M*-model. Adding matching inequalities slightly improves the *M*-model, while this improvement is significant for the *Z*-model. However, the *MC*-model is still worse than the *Z*-model. The *ZC*-model provides the smallest gap, on the average almost a half of the gap of the *M*-model. Clearly, the more cuts generated the longer the running time. Although the time for generating one cut is very short, the relaxations of the *MC*- and *ZC*-models must be resolved for each new cut added.

4.2 Efficiency of solving the MIP models for MWCSP

To compare the efficiency of the models, we have implemented our own branch-and-bound algorithms BBM for the M -model, BBZ for the Z -model, and branch-and-cut algorithms BCM for the MC -model, BCZ for the ZC -model. The depth-first tree search is used and branching is done using the most fractional (nearest to 0.5) variable. For branch-and-cut, the generated cuts are used globally. Each time a B&B (branch-and-bound) node is solved, the cuts generated locally at the node are added to the global list of cuts used for all the subsequent nodes.

Table 2 shows running times and the numbers of processed B&B nodes. Entries “3h*” indicate that an optimum could not be reached in 3 hours. Due to the poor quality of the linear relaxation of the M -model, BBM visits more B&B nodes than other models. The ZC -model needs the least number of B&B nodes, followed by the Z -model and the MC -model. Since the time to solve one B&B node for the M -model is very short, it happens that the time for solving the M -model and the MC -model is shorter than for other models for small networks. For large networks, the time for solving the ZC -model is shorter than for other models because the ZC -model needs much less B&B nodes to reach the optimum. Besides, the time for solving the Z -model is shorter than for the MC -model for large networks. It is interesting to see that the ZC -model can find an optimum for some network instances for which the M -model and the MC -model cannot find it in three hours.

5 Conclusion

We have considered an important problem in wireless networks – maximizing the weighted value of parallel link transmissions. We have investigated two basic approaches of modeling the SINR constraint and showed that the model with auxiliary variables eliminating bi-linearities can give a better upper bound than the “big M” model. Moreover, we have introduced matching inequalities and developed a minimum odd-cut generation method based on the Gomory-Hu tree algorithm for generating violated cuts. Our numerical study shows that the enhanced model increases effectiveness of solving the considered problem.

Acknowledgment. The paper was prepared under the ELLIIT project. M. Pióro was supported by National Science Center (NCN, Poland) under grant 2011/01/B/ST7/02967.

References

- [1] Andrews, M. and M. Dinitz, *Maximizing capacity in arbitrary wireless networks in the sinr model: Complexity and game theory*, in: *IEEE INFOCOM '09*, pp. 1332 –1340.
- [2] Brar, G., D. Blough and P. Santi, *Computationally efficient scheduling with the physical interference model for throughput improvement in wireless mesh networks*, in: *Proc. ACM MobiCom '06*, pp. 2–13.
- [3] Capone, A., G. Carello, I. Filippini, S. Gualandi and F. Malucelli, *Routing, scheduling and channel assignment in wireless mesh networks: Optimization models and algorithms*, *Ad Hoc Networks* **8** (2010), pp. 545–563.
- [4] Capone, A., L. Chen, S. Gualandi and D. Yuan, *A new computational approach for maximum link activation in wireless networks under the sinr model.*, *IEEE Trans. Wireless Commun.* **10** (2011), pp. 1368–1372.
- [5] ElBatt, T. and A. Ephremides, *Joint scheduling and power control for wireless ad hoc networks*, *IEEE Trans. Wireless Commun.* **3** (2004), pp. 74 – 85.
- [6] Goussevskaia, O., Y. Oswald and R. Wattenhofer, *Complexity in geometric sinr*, in: *Proc. ACM MobiHoc '07*, pp. 100–109.
- [7] Goussevskaia, O., R. Wattenhofer, M. Halldorsson and E. Welzl, *Capacity of arbitrary wireless networks*, in: *Proc. IEEE INFOCOM '09*, pp. 1872 –1880.
- [8] Kesselheim, T., *A constant-factor approximation for wireless capacity maximization with power control in the sinr model*, in: *Proc. ACM-SIAM SODA '11*, pp. 1549–1559.
- [9] Korte, B. and J. Vygen, “*Combinatorial Optimization: Theory and Algorithms*,” Springer, Germany, 2006, 3rd edition.
- [10] Li, Y., M. Pióro, D. Yuan and J. Su, *On joint optimization of link rate assignment and transmission scheduling in wireless mesh networks*, in: *Proc. NETWORKS 2012*.
- [11] Pióro, M., M. Żotkiewicz, B. Staehle, D. Staehle and D. Yuan, *On Max-Min Fair Flow Optimization in Wireless Mesh Networks*, *Ad Hoc Networks*, Special Issue on Models and Algorithms for Wireless Mesh Networks (2011).
- [12] Tsoutsoulikis, K., *Cut Tree Algorithms: An Experimental Study*, <http://www.cs.princeton.edu/~kt/cut-tree/> (1999).
- [13] Viswanathan, H. and S. Mukherjee, *Throughput-range tradeoff of wireless mesh backhaul networks*, *IEEE JSAC* **24** (2006), pp. 593 – 602.

A fast heuristic approach for train timetabling in a railway node

Fabio Furini¹

*LIPN, Université Paris 13
99 avenue Jean-Baptiste Clément, 93430 Villetteuse, France*

Martin Philip Kidd²

*DEI, Università di Bologna
Viale Risorgimento 2, 40136, Bologna, Italy*

Abstract

We consider a conflict-free scheduling problem which arises in railway networks, where ideal timetables have been provided for a set of trains, but where these timetables may be conflicting. We use a space-time graph approach from the railway scheduling literature in order to develop a fast heuristic which resolves conflicts by adjusting the ideal timetables while attempting to minimize the deviation from the ideal timetable. Our approach is tested on realistic data obtained from the railway node of Milan.

Keywords: Train timetabling, Railway network, Heuristic Algorithm.

¹ Email: fabio.furini@lipn.univ-paris13.fr

² Email: martin.kidd@unibo.it

1 Introduction

In this paper we consider the problem of conflict resolution inside a railway node, which is characterized by a number of stations in an urban area interconnected by tracks to form a railway network. An ideal timetable is provided for each train, which specifies the stations to be visited in the railway network together with ideal arrival and departure times at each of the stations. The problem arises when these timetables are conflicting, either due to, for instance, limited station capacity or violation of minimum headway times. The ideal timetables of some trains should therefore be adjusted or some trains canceled in order to reach a feasible timetable which is as close as possible to the ideal timetable.

Cacchiani and Toth [4], very recently, provided an extensive overview of railway optimization problems together with discussions on general solutions approaches and types of problems considered. ILP formulations of similar railway scheduling problems are given by Brännlund *et al.* [1] and Carey and Lockwood [6], while Flier *et al.* [7] considers a problem related to the method used in this paper where an additional train line is included into an existing timetable on a corridor. As a solution approach we extend the space-time graph approach adopted in Cacchiani *et al.* [3] and in Caprara *et al.* [5].

The contribution of this paper lies in developing a fast heuristic that can solve real-world problem instances within a short period of time. Furthermore, for the first time we take into account specific operational constraints arising from the presence of junctions in the network, as well as the fact that trains are able to travel in both directions between any two stations or use parallel tracks. These are important factors that are usually simplified or omitted in other approaches.

2 Problem description

We assume to be given a set of trains T , a set of stations S , a set of junctions J and a set of tracks R each of which connects two locations³, where some of the tracks are parallel tracks (two tracks connecting the same two locations in the same direction). The ideal timetable of a train t is specified by a set $L_t \subseteq S \cup J$ of locations to be visited, the order in which they should be visited, the track to be used in case of parallel tracks between stations, an ideal arrival time $\alpha(t, \ell)$ and an ideal departure time $\epsilon(t, \ell)$ at each location $\ell \in L_t$.

³ In what follows a location will refer to either a station or a junction.

Furthermore, for each train t an ideal profit π_t is given, which represents the profit associated with the train if its timetable is not changed.

The constraints to be satisfied are the following. For the arrival of any two trains $t, t' \in T$ at the same station $s \in S$ from track $r \in R$, a minimum headway time between the arrival time of the first train and the arrival time of the second train at s is given and denoted by $h^+(s, r)$, while similarly for the departure of the two trains from s onto r a minimum headway time between their departure times is given and denoted by $h^-(s, r)$. Furthermore, each station $s \in S$ has a maximum capacity c_s of trains simultaneously present, while a junction $j \in J$ has capacity of one with an occupation time of o_j . Finally, the travel times inside junctions and on tracks connecting locations are assumed to be fixed as specified by the ideal arrival and departure times for each train.

In order to resolve possible conflicts that may occur, trains may be either be *shifted* at their initial locations, *i.e.* depart ahead of schedule or with a delay, trains may be *stretched* at stations they visit, *i.e.* their ideal stopping times at stations may be increased, or trains may be moved from a track onto a parallel track if one exists. Trains are further restricted in terms of shifting and stretching. For each train $t \in T$ a maximum shift s_t^+ ahead of schedule and maximum shift s_t^- behind schedule is specified, as well as a maximum total stretch σ_t over all stations. Furthermore, each train is associated with three weights ω_t^+, ω_t^- and ψ_t , which correspond to the penalty for respectively shifting ahead of schedule, shifting behind schedule and stretching.

3 Description of the algorithm

The algorithm considers each of the trains individually in a predetermined order (as discussed in the next section). Let $T^t \subset T$ denote the set of trains that precede train $t \in T$ in this order. If t is the i -th train in this order for $2 \leq i \leq |T|$, then during the i -th iteration of the algorithm an approximate optimal timetable for train t is determined, subject to minimum headway and capacity constraints imposed by the optimal timetables found during the previous iterations for trains in T^t . During the first iteration the timetable for the first train is simply its ideal one.

For the purpose of determining an approximate optimal timetable for train $t \in T$, we define a *space-time* directed graph $G = (V, A)$ with node set V and arc set A , where each node represents either a departure or an arrival of t at a specific location from a specific track at a specific point in time, where time is discretized. Each arc of G either represents t traveling from one station to

another, t stopping at a station or t traveling through a junction. For a train t and a location $\ell \in L_t$, let $\mathcal{A}_{t,\ell}$ denote the set of potentially feasible time instants at which t may arrive at ℓ , and let $\mathcal{D}_{t,\ell}$ denote the set of potentially feasible time instants at which t may depart from ℓ . Hence for $\tau \in \mathcal{A}_{t,\ell}$ it holds that

$$\alpha(t, \ell) - s_t^+ \leq \tau \leq \alpha(t, \ell) + s_t^- + \sigma_t,$$

while $\tau \in \mathcal{D}_{t,\ell}$ it holds that

$$\epsilon(t, \ell) - s_t^+ \leq \tau \leq \epsilon(t, \ell) + s_t^- + \sigma_t.$$

For each location $\ell \in L_t$ such that $\ell \in S$, where $r \in R$ denotes the track used by t to arrive at ℓ , and each time instant $\tau \in \mathcal{A}_{t,\ell}$, there is a node in V representing the arrival of train t at ℓ provided that

- (a) there does not exist a train in T^t which is scheduled to arrive at ℓ on track r within the time interval $[\tau - h^+(\ell, r), \tau + h^+(\ell, r)]$, and
- (b) the number of trains in T^t which are scheduled to be present at ℓ at time τ is less than c_ℓ .

Similarly, for each location $\ell \in L_t$ such that $\ell \in S$, where $r \in R$ denotes the track used by t to depart from ℓ , and each time instant $\tau \in \mathcal{D}_{t,\ell}$, there is a node in V representing the departure of train t from ℓ provided that

- (a) there does not exist a train in T^t which is scheduled to depart from ℓ on track r within the time interval $[\tau - h^-(\ell, r), \tau + h^-(\ell, r)]$, and
- (b) the number of trains in T^t which are scheduled to be present at ℓ at time τ is less than c_ℓ .

Furthermore, for each location $\ell \in L_t$ such that $\ell \in J$ and each time instant $\tau \in \mathcal{A}_{t,\ell}$, there is a node in V representing the arrival of train t at ℓ provided that there does not exist a train in T^t which is present at ℓ during the time interval $[\tau, \tau + o_\ell]$. Similarly, for each location $\ell \in L_t$ such that $\ell \in J$ and each time instant $\tau \in \mathcal{D}_{t,\ell}$, there is a node in V representing the departure of train t from ℓ provided that there does not exist a train in T^t which is present at ℓ during the time interval $[\tau - o_\ell, \tau]$.

Each arc of G is associated with a profit, and arcs representing a train traveling between two locations or through a junction have 0 profit. If, for a location $l \in L_t$ where $l \in S$, an arc connects the vertex in V representing the arrival of train t at l at time $\tau \in \mathcal{A}_{t,\ell}$ to another vertex representing the departure of train t from location l at time $\tau' \in \mathcal{D}_{t,\ell}$, $\tau < \tau'$, (i.e. the arc represents t stopping at location l from τ to τ'), then the associated profit of

this arc is equal to

$$-\psi_t(\tau' - \tau - (\epsilon(t, \ell) - \alpha(t, \ell))),$$

i.e. the weighted negative value of the stretch at ℓ .

Finally, V contains a dummy starting vertex v_s and a dummy ending vertex v_e . If $\ell \in L_t$ is the initial location of train t , then for each $\tau \in \mathcal{D}_{t,\ell}$ there exists an arc connecting v_s to the vertex representing the departure of t from ℓ at time τ . The profit of this arc is equal to $\pi_t - \omega_t^-(\tau - \epsilon(t, \ell))$ if $\tau \geq \epsilon(t, \ell)$, and $\pi_t - \omega_t^+(\epsilon(t, \ell) - \tau)$ otherwise, i.e. the weighted shift of t at ℓ subtracted from the ideal profit of t . If $\ell \in L_t$ is the final location of train t , then for each $\tau \in \mathcal{A}_{t,\ell}$ there exists an arc connecting the vertex representing the arrival of t at ℓ at time τ to v_e , where this arc has an associated profit of 0.

The problem under consideration in this paper is an extension of the problem considered in [5], where the authors addressed the *Train Timetabling Problem* (TTP) for a single corridor imposing just a subset of constraints arising from real world applications. In this paper the authors proved that the TTP is NP-hard and they introduced the following mathematical formulation of the problem (reported here for sake of completeness) using a binary variable x_a which takes a value of 1 if a train t is using arc $a \in A$. Moreover $\delta_t^+(v)$ denotes the set of feasible leaving arcs for node $v \in V$ and $\delta_t^-(v)$ the set of feasible entering arcs, given a specific train $t \in T$.

$$\text{maximize} \quad \sum_{a \in A} p_a x_a \quad (1)$$

$$\text{subject to} \quad \sum_{a \in \delta_t^+(v_s)} x_a \leq 1 \quad t \in T \quad (2)$$

$$\sum_{a \in \delta_t^-(v)} x_a - \sum_{a \in \delta_t^+(v)} x_a = 0 \quad t \in T, v \in V \setminus \{v_s, v_e\} \quad (3)$$

$$\sum_{a \in \tilde{C}} x_a \leq 1 \quad \tilde{C} \in \tilde{\mathcal{C}} \quad (4)$$

$$x_a \in \{0, 1\} \quad a \in A \quad (5)$$

The objective function (1) maximizes the total profit, where p_a gives the profit of arc $a \in A$ as described above in this section. Constraints (2) and (3) ensure that if a train is scheduled, it follows a feasible path. Finally *clique constraints* (4) (see [5]) forbid the simultaneous selection of incompatible arcs imposing the set of operational constraints (described in Section 2) — in

particular the new operational constraints arising from the dynamics inside junctions.

Due to the fact that this mathematical formulation is characterized by large numbers of constraints and variables, which prevents the use thereof in real world applications, we decided to adopt a heuristic approach based on the following idea. Given a specific ordering of trains, the algorithm tries to schedule these trains one after the other, each time taking into consideration operational constraints imposed by previously scheduled ones. Given the space-time graph for a train t during an iteration of the algorithm, an approximate longest (maximum profit) path from v_s to v_e in this directed, acyclic graph is heuristically computed using a relaxed dynamic programming approach. If no such path exists, or if the total profit of the path is negative, the train is canceled. Otherwise, the train timetable is updated according to its optimal path.

4 Preliminary results

In this section we present some preliminary results obtained by using realistic data provided by RFI (Rete Ferroviaria Italiana, the main railway infrastructure manager in Italy) concerning the railway node of Milan. We consider an instance where there are 53 stations, 10 junctions and 146 tracks (14 parallel tracks) connecting these locations. Moreover, ideal timetables are given for 1500 trains for a period of one day, where each time instant represents a minute of the day. Also, the maximum shift ahead of schedule is equal to 1 minute, the maximum shift behind schedule is 10 minutes and the maximum stretch is 2 minutes. Finally, for the ideal profits for each train and the penalty weights for shifting and stretching, we use the realistic values used in [2] (shown here in Table 1). The algorithm was implemented in C++ and run in Linux Ubuntu (compiled using g++ with option -O3) on an Intel Core i3-2330M CPU computer clocked at 2.20GHz, with 4 GB of RAM.

Train type	Eurostar	Euronight	Intercity	Express	Combined	Direct	Local	Freight
p_{it}	200	150	120	110	100	100	100	100
ω_t^+/ω_t^-	7	7	6	5	6	5	5	2
ψ_t	10	10	9	8	9	8	6	3

Table 1
Profits and penalty weights for each train type

To improve the quality of the solutions obtained, the algorithm is repeated

with different examination orders of the trains, henceforth referred to as a *multi-run*. Two different methods for generating random orders were used, as well as a first-come-first-serve (FCFS) order. In one method for generating random orders trains are partitioned into two classes, one of which is considered to be more important than the other (priority ordering). In this case a random permutation of the more important trains are generated and examined first, followed by a random permutation of the second class of trains. In the second method for generating random orders, no distinction between the trains is made and a random permutation of all the trains is generated.

The preliminary results obtained using our algorithm and the three ordering strategies mentioned above (priority, no priority and FCFS) is shown in Table 2 for multi-runs consisting of 10, 100 and 1000 iterations. The table shows the total profit obtained summed over all the trains that were not canceled, and also contains information regarding the total number of trains canceled, the number of important trains canceled, and the number of trains canceled due to a negative profit.

The total time and the time to obtain the best solution is shown (both expressed in seconds), as well as the number of different solutions obtained by the algorithm. Furthermore, the total stretch over all the trains in the best solution found is given, together with the number of trains for which the ideal timetable was changed as well as the total number of parallel tracks used. Finally in order to give an indication of the quality of the solutions found, it is worth mentioning that a bound on the optimal solution value can easily be computed by summing the profits of all the trains. The percentage gap between this bound and the heuristic solution values obtained is given in the final column of the table.

5 Conclusion

In this paper we presented a heuristic for solving the conflict free scheduling problem on a railway network. Results were obtained using realistic data from the node of Milan. The results at this stage are preliminary and future work include an extension of Table 2 as well as extensions in the algorithm. This will include allowing trains the possibility to be rerouted when there are no feasible paths in the space-time graph. For this purpose alternative routes for each train have to be calculated or given as input to the problem. Furthermore, apart from resolving conflicts in given timetables, the algorithm proposed in this paper may also be used as a planning tool for testing network capacities or for real time conflict resolution in case of disruptions on a railway network.

	Profit	Iter.	Trains can.	Imp. trains can.	Can. due to profit	Time to sol.	Diff. sol.	Time total	Total str.	Trains aff.	Tracks changed	Gap
FCFS	144562	1	151	1	53	0.3	1	0.3	4867	567	116	12.01%
No priority	145676	10	158	5	20	0.6	10	3.2	2934	505	98	11.34%
	146147	100	152	7	28	14.1	100	32.9	2857	511	128	11.05%
	146846	1000	145	4	23	135.8	817	333.1	3021	537	120	10.62%
Priority	145561	10	160	0	26	2.7	10	3.4	2913	509	115	11.41%
	146383	100	151	0	23	6.8	99	33.0	2953	531	121	10.91%
	147034	1000	143	0	20	100.7	817	326.8	3217	542	110	10.51%

Table 2
Results obtained for the node of Milan

In the latter case the approach would especially prove useful due to the fact that the computation times are small.

References

- [1] Brännlund, U., Lindberg, P.O., Nöu, A. and Nilsson, J.E., 1998, Allocation of scarce track capacity using lagrangian relaxation, *Transportation Science*, 32, pp. 358–369.
- [2] Cacchiani, V., Caprara, A. and Toth, P., 2008. A column generation approach to train timetabling on a corridor, *4OR*, 6, pp. 125–142.
- [3] Cacchiani, V., Caprara, A. and Toth, P., 2010. Scheduling extra freight trains on railway networks, *Transportation Research Part B*, 44, pp. 215–231.
- [4] Cacchiani, V. and Toth, P., 2012, Nominal and robust train timetabling problems, *European Journal of Operational Research*, 219, pp. 727–737
- [5] Caprara, A., Fischetti, M. and Toth, P., 2002. Modeling and solving the train timetabling problem. *Operations Research*, 50, pp. 851–861.
- [6] Carey, M. and Lockwood D., 1995, A model, algorithms and strategy for train pathing, *Journal of the Operational Research Society*, 46, pp. 988–1005.
- [7] Flier, H., Graffagnino, T. and Nunkesser, M., 2009, Planning additional trains on corridors. In: *Proceedings of the 8th International Symposium on Experimental Algorithms*.

A polyhedral study of the Hamiltonian p -median problem

Lena Hupp^{1,2} Frauke Liers^{1,3}

*Department Mathematik
Friedrich-Alexander-Universität Erlangen-Nürnberg,
Cauerstraße 11
91058 Erlangen
Germany*

Abstract

Given an edge-weighted graph $G = (V, E)$, the Hamiltonian p -median problem (HpMP) asks for determining p cycles in G whose total length is minimized such that each node is contained in exactly one cycle. As the travelling salesman problem (TSP) corresponds to the choice $p = 1$, the HpMP can be interpreted as a generalization of the TSP. In this paper, we study the polytope associated with the HpMP. To this end, we investigate several known classes of valid inequalities with respect to their facet inducing properties. Furthermore, we show that a subset of the well-known 2-matching inequalities from the TSP define facets of the Hamiltonian p -median polytope.

Keywords: Travelling salesman problem, polyhedral study

¹ Financial support from the German Science Foundation is acknowledged under contract Li 1675/1.

² Email: Lena.Hupp@math.uni-erlangen.de

³ Email: frauke.liers@math.uni-erlangen.de

1 Introduction

The Hamiltonian p -median problem was first introduced by Branco and Coelho [1] and can be interpreted as a generalization of the well-studied travelling salesman problem (TSP). We consider the symmetric Hamiltonian p -median problem (HpMP) that can be defined on an undirected complete graph $G(V, E)$ with $|V| = N$ nodes, $|E| = N(N - 1)/2$ edges and edge costs $c_e \geq 0$, $e \in E$ as follows. For $p \in \mathbb{Z}$, find p mutually node-disjoint cycles in $G = (V, E)$ of minimal total edge cost, so that each node $i \in V$ is contained in exactly one of the p cycles. For $p = 1$, we get the symmetric travelling salesman problem (TSP). As in the case for the TSP, the HpMP is an \mathcal{NP} -hard combinatorial optimization problem ([6]). For $p > 1$, research has mainly concentrated on modeling the HpMP as integer programs (IPs) ([1], [4], [5], [6], [7]) and on evaluating branch-and-cut algorithms for their solution ([6]). Except from [4], little research has been done in order to understand the polytope associated with the HpMP. In this work, we aim at filling this gap by presenting results of a polyhedral study of the HpMP.

The paper is structured as follows. In Section 2 we review IP formulations in the natural variable space. In Section 3 we investigate the associated Hamiltonian p -median polytope (HpM-polytope). We study the inequalities from the IP formulations of Section 2 from a polyhedral point of view. We present a new class of valid inequalities that forbids the existence of less than p cycles. In Section 3.3 we prove that some of the well-known 2-matching inequalities [3] from the TSP induce facets of the HpM-polytope.

2 IP Formulations in the natural variable space

In this section we present the IP formulations given in [4] and [6] in the natural variable space that solely use edge variables $x_e = x_{ij}$ associated with an edge $e = (i, j) \in E$. To this end, we first introduce some notations.

Let $P = \{S_1, \dots, S_m\}$ be a partition of V in $m \in \mathbb{N}$, $2 \leq m \leq |V|$ subsets S_i , $i = 1, \dots, m$. Then $P_m^l(V)$ is defined as the set of all partitions $P = \{S_1, \dots, S_m\}$ of V in m subsets of cardinality greater than or equal to l with $S_i \subset V$, $S_i \neq \emptyset$, $|S_i| \geq l$, $i \in \{1, \dots, m\}$, $S_i \cap S_j = \emptyset$ for $i \neq j$, $\bigcup_{i=1}^m S_i = V$.

For each element $\{S_1, \dots, S_m\} \in P_m^l(V)$ we denote by $E(S_1, \dots, S_m) := \{(i, j) \in E \mid i \in S_v, j \in S_w, S_v \neq S_w, S_v, S_w \in P\}$ an (undirected) m -cut. For the special case $m = 2$ we define for $W \subset V$, $W \neq \emptyset$, $\delta(W) = \{(i, j) \in E \mid i \in W, j \in V \setminus W\}$ the undirected cut associated with the node set W .

For a subset $S \subseteq V$ with $|S| = l$ we denote by $C_k = ((v_1, v_2), \dots, (v_{n_k}, v_1))$ with $v_i \in S$, $v_i \neq v_j$, $i \neq j$ and $|C_k| = n_k \leq l$ a cycle in S . If $n_k = l$ holds, we call it a Hamiltonian cycle HC_k . Further we define $E(S)$ as the subset of edges $(i, j) \in E$ with both nodes i and j in S . Correspondingly, for a subset $T \subseteq E$ we denote by $V(T)$ the subset of nodes, induced by the edges in T . Further, let \mathbf{C}_p be defined as

$$\mathbf{C}_p := \{C \subseteq E \mid |C| = |V| \text{ and the edges in } C \text{ form at most } p \text{ pairwise node-disjoint cycles}\}.$$

We define the set of all Hamiltonian p -medians by

$$\mathcal{H}^{N,p} := \{H = \{C_1, \dots, C_p\} \mid C_i \in E, C_i \text{ defines a cycle in } V, V(C_i) \cap V(C_j) = \emptyset, i \neq j, \bigcup_{i=1}^p V(C_i) = V\}.$$

For a vector $x \in \{0, 1\}^{\frac{N(N-1)}{2}}$ and a subset $T \subseteq E$ we define $x(T) = \sum_{e \in T} x_e$. An IP formulation in the natural variable space for the HpMP is given in [4]:

$$\begin{aligned} \min & \sum_{e \in E} c_e x_e && (\mathbf{IP1}) \\ x(\delta(i)) &= 2 & i = 1, \dots, N & (\text{DEG}) \\ x(E(S_1, \dots, S_{p+1})) &\geq 2 & \{S_1, \dots, S_{p+1}\} \in P_{p+1}^3(V) & (\text{SNC}) \\ x(C) &\leq N - p & C \subset E, C \text{ Hamiltonian cycle in } V & (\text{HAMC}) \\ x_e &\in \{0, 1\} & e \in E & (\text{BIN}) \end{aligned}$$

Equalities (DEG) are the well-known degree constraints. The subtour number constraints (SNC) ensure that a solution consists of no more than p cycles. Inequalities (HAMC) guarantee that there are at least p cycles in a solution. Instead of inequalities (HAMC) in (IP1) the class of inequalities (MINC) is used in [6], namely

$$\sum_{e \notin C} x_e \geq 2, \quad C \in \mathbf{C}_{p-1}. \quad (\text{MINC})$$

The results presented in [6] indicate that solving (IP1) by branch-and-cut is effective in practice. Therefore we study the polytope associated with (IP1) next.

3 The HpM-Polytope

We define the HpM-polytope as the convex hull of the set of the incidence vectors of all Hamiltonian p -medians on $G(V, E)$:

$$\mathcal{Q}^{N,p} = \text{conv} \left\{ x^H \mid H \text{ is a Hamiltonian } p\text{-median in } G(V, E) \right\}.$$

The dimension $\dim(\mathcal{Q}^{N,p})$ of the Hamiltonian p -median polytope for $N \geq 3p + 1$ is given in [4] as $\dim(\mathcal{Q}^{N,p}) = (N(N - 1)/2) - N$. Therefore, the dimension of the HpM-polytope equals the dimension of the TSP-polytope. According to a remark in [4] for $N > 3p + 1$, $(N, p) \neq (4, 1)$, $(N, p) \neq (5, 1)$ the non-negativity constraints $x_{ij} \geq 0$, $1 \leq i < j \leq N$ and the trivial inequalities $x_{ij} \leq 1$ for $N > 3p$, $(N, p) \neq (4, 1)$ define facets of $\mathcal{Q}^{N,p}$.

An experimental study using [2] shows that for $p = 2$, $N = 6$ a full linear description of the HpM-polytope is given by

$$\mathcal{Q}^{6,2} = \left\{ x \in \mathbb{R}^{|E|} \mid x(\delta(i)) = 2 \forall i \in V, \sum_{\substack{i,j \in W \\ i,j \neq 1}} x_{ij} \geq 1, W \subseteq V, |W| = 3 \right\}.$$

For $N = 7$ and $p = 2$ more than 466,000 inequalities are needed for a full linear description of $\mathcal{Q}^{7,2}$. In the next sections we present classes of inequalities for general N and p . We begin with the analysis of the inequalities from the IP formulation (**IP1**).

3.1 Polyhedral study of the inequalities from (**IP1**)

In [4] it is proved that inequalities (**SNC**) induce facets of $\mathcal{Q}^{N,p}$. Therefore the question arises if a similar result can be achieved for (**HAMC**) and (**MINC**).

The first observation is that for $p = 2$ inequalities (**HAMC**) and (**MINC**) are equivalent. As in the TSP context, taking an arbitrary inequality (**MINC**), summing up the degree constraints (**DEG**) from (**IP1**) and adding them to the inequality (**MINC**) results in an equivalent formulation

$$\sum_{e \in C} x_e \leq N - 2, \quad C \in \mathbf{C}_{p-1}. \quad (\text{MINC}^*)$$

Whereas (**HAMC**) and (**MINC**) are equal for $p = 2$, this does not hold for $p > 2$. Inequalities (**MINC**) can be used to show the following lemma.

Lemma 3.1 For $p = 2$, $N \geq 3p$ inequalities (MINC*) and (HAMC) do not define facets of $\mathcal{Q}^{N,2}$.

A similar result can be achieved for $p > 2$ for inequalities (MINC*).

Lemma 3.2 For $p > 2$, $N \geq 3p$, (MINC*) do not define facets of $\mathcal{Q}^{N,p}$.

Let $F = \{x \in \mathcal{Q}^{N,p} \mid a^T x = a_0\}$ be a face of $\mathcal{Q}^{N,p}$ induced by a valid inequality $a^T x \leq a_0$. Then we define the *dimension gap* induced by this face as $\dim(\mathcal{Q}^{N,p}) - 1 - \dim(F)$. It can be proved that for $p = 2$ the dimension gap for (MINC*) or (HAMC) grows at least linearly in N . The same holds for $p > 2$ for (MINC*).

3.2 A new class of inequalities that forbid less than p cycles

We introduce a new valid class of inequalities that makes sure that there are at least p cycles in a solution.

Theorem 3.3 Let $p \geq 2$ and $N \geq 3p$, $(N, p) \neq (6, 2)$. Let $W \subseteq V$, $|W| = N - p - 2$ and let C_W be a Hamiltonian cycle in W . Further let $T_1, \dots, T_p \subseteq V$ with $|T_i| = 2$, $|T_i \cap W| = 1 \forall i = 1, \dots, p$ and $T_i \cap T_j = \emptyset$, $i \neq j$, $i, j \in \{1, \dots, p\}$. Then

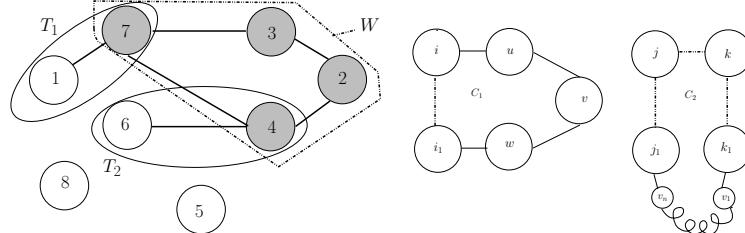
$$\sum_{e \in C_W} x_e + \sum_{i=1}^p x(E(T_i)) \leq |W| \quad (1)$$

is a valid inequality for $\mathcal{Q}^{N,p}$. If a binary vector x satisfies (DEG) in (IP1) and inequalities (1), it contains at least p cycles.

As an example the support graph of inequalities of the form (1) for $N = 8$, $p = 2$ with $W = \{2, 3, 4, 7\}$, $T_1 = \{1, 7\}$, $T_2 = \{4, 6\}$ is presented in Figure 1(a). Computational experiments with [2] for $p = 2$ and $N = 7, \dots, 11$ imply that except from $N = 7$, $p = 2$ inequalities (1) do not induce facets of $\mathcal{Q}^{N,p}$. For example for $N = 11$, $p = 2$ the dimension gap for inequalities of the form (1) amounts to 14 whereas for (MINC*) respectively (HAMC) it is 11.

3.3 Facets and valid inequalities from the 2-matching relaxation of the TSP

In this section we show that the close relationship of the HpMP to the TSP enables us to identify a class of facet inducing inequalities for the HpMP. By solving the IP formulation without constraints (HAMC) and (SNC) we get the well-known 2-matching problem [3] that is a relaxation for the HpMP. Therefore all valid inequalities for the 2-matching problem are also valid for the HpMP. The question arises if all or some of the 2-matching inequalities



(a) Support graph: inequality of the form (1)
(b) Handle and teeth subdivided in C_1 and C_2 in proof of Theorem 3.4

Fig. 1. Support graphs

[3] are also facet inducing for the HpM-polytope. The next theorem answers part of this question for $p > 1$.

Theorem 3.4 *Let $p \geq 2$. Then all 2-matching inequalities [3] are valid for $\mathcal{Q}^{N,p}$. For $N > 3p + 3$ the 2-matching inequalities of the form*

$$x(E(H)) + \sum_{i=1}^3 x(E(T_i)) \leq 4 \quad (2)$$

with $H, T_1, \dots, T_3 \subseteq V$, $|H| = 3$ and $|H \cap T_i| = 1$, $|T_i| = 2$, $\forall i = 1, \dots, 3$ and $T_i \cap T_j = \emptyset$, $1 \leq i < j \leq 3$ define facets of $\mathcal{Q}^{N,p}$.

As in the TSP context, we call the node set H *handle* and the node sets T_1, \dots, T_3 *teeth*. Inequalities (2) are the special case of the 2-matching inequalities [3] for the TSP where the handle H consists of three nodes and the node sets T_1, \dots, T_k form $k = 3$ teeth. The cardinality of inequalities (2) is given by $\mathcal{O}(N^6)$. Next, we give a sketch of the proof of Theorem 3.4.

Proof. Validity is clear from the discussion above. The proof of the facet property is sketched next. The proof can be seen as an extension of the TSP proof in [8]. Let $F_M = \{x \in \mathcal{Q}^{N,p} \mid x(E(H)) + \sum_{i=1}^3 x(E(T_i)) = 4\}$ be the face induced by an inequality (2) and let D_M be the coefficient matrix corresponding to the linear equality system

$$x(\delta(i)) = 2 \quad i = 1, \dots, N \quad (3)$$

$$x(E(H)) + \sum_{i=1}^3 x(E(T_i)) = 4. \quad (4)$$

Furthermore for $i, j, k, i_1, k_1, j_1 \in V$ let $H = \{i, j, k\}$ be the handle with the teeth $T_1 = \{i, i_1\}$, $T_2 = \{j, j_1\}$ and $T_3 = \{k, k_1\}$. We consider the edge set $\mathcal{F} = \delta(i) \cup \{(j, k)\}$ with the coefficient matrix $D_{\mathcal{F}}$ that consists of the corresponding

columns of D_M . Further we define $T \subseteq V$ by $T = \{H \cup T_1 \cup T_2 \cup T_3\}$.

We show that each hyperplane \mathcal{H} with $F_M \subseteq \mathcal{H} := \{x \in \mathbb{R}^{|E|} \mid a^T x = a_0\}$ is a linear combination of (3) and (4). More precisely, we show that there exists a $\lambda \in \mathbb{R}^{N+1}$ so that with the expanded matrix $(D_{\mathcal{F}}, (\mathbf{2}_N, 4)^T)$ the relationship $b_{uv} = b_0 = 0 \quad \forall uv \in E$ holds for the transformed vector $(b^T, b_0) = (a^T, a_0) + \lambda^T(D_{\mathcal{F}}, (\mathbf{2}_N, 4)^T)$. By $\mathbf{2}_N$ we denote the N dimensional vector of twos. It is not difficult to see that the expanded matrix $(D_{\mathcal{F}}, (\mathbf{2}_N, 4)^T)$ has full rank. Thus for the transformed vector (b^T, b_0) it holds $b_{uv} = b_0 = 0 \quad \forall uv \in \delta(i) \cup \{(j, k)\}$. We show that this implies $b_{uv} = b_0 = 0 \quad \forall uv \in E$.

Let H_m and H_n , $m, n \in \mathbb{N}$ be two Hamiltonian p -medians, for whose incidence vectors x^{H_m} and x^{H_n} equality (4) holds. We compare coefficients of the Hamiltonian p -medians H_1 and H_2 that only differ in the first cycle. For this comparison we choose Hamiltonian p -medians that either contain T in the first cycle or we take those where T is subdivided in the first and second cycle. For the first case, we can use some comparisons from [8]. An example for the second case is displayed in Figure 1(b). The comparison of coefficients of suitable Hamiltonian p -medians results in $b_{uv} = 0 \quad \forall u, v \in E$. \square

The 2-matching inequalities can be separated in polynomial time [9]. Therefore the question arises, if Theorem 3.4 can be expanded for general $|H|$, k and p . We conjecture that this is true because computational experiments with [2] indicate that the 2-matching inequalities for $p = 2$ and general $|H|$ and k induce facets for $N > 3p + 3$. The results for $p = 2$, $N \leq 11$ are presented in Table 1. For $N = 11$, $p = 2$ and $|H| = 7$, $k = 1, 3$ and $|H| = 8$, $k = 3$ the corresponding 2-matching inequalities also define facets.

4 Conclusions

In this paper, we investigate the polyhedral structure of the HpM-polytope. We show that inequalities (MINC) and (HAMC) introduced in [4] [6], that ensure at least p cycles in a solution, do not define facets. We present a new class of valid inequalities, that displays an alternative to them. It would be interesting to compare these three mentioned classes of inequalities within a branch-and-cut algorithm. Furthermore, we show validity of the 2-matching inequalities from the TSP for the HpMP and prove facet defining properties for a subset. On the theoretical side the question remains whether all 2-matching inequalities define facets of the HpM-polytope as it is suggested from experimental tests. On the empirical side, it is interesting to test the impact of the 2-matching inequalities in a branch-and-cut algorithm.

N	$ H = 3$ $k = 3$	$ H = 4$ $k = 1$	$ H = 4$ $k = 3$	$ H = 5$ $k = 1$	$ H = 5$ $k = 3$	$ H = 5$ $k = 5$	$ H = 6$ $k = 3$	$ H = 6$ $k = 5$	$\dim(\mathcal{Q}^{N,2})$
7	2	-	2	-	-	-	-	-	14
8	13	10	8	-	13	-	-	-	20
9	26	26	20	26	20	-	26	-	27
10	34	34	34	34	28	34	34	-	35
11	43	43	43	43	43	43	43	43	44

Table 1

Dimension of a face induced by a 2-matching inequality for $p = 2$, $N \leq 11$.

References

- [1] Branco, I. M., and J.D. Coelho, *The Hamiltonian p -Median Problem*, European Journal of Operational Research **47**(1) (1990), 86–95.
- [2] Christof, T., and A. Löbel, *PORTA: POlyhedron Representation Transformation Algorithm*, Konrad-Zuse-Zentrum für Informationstechnik, Berlin Germany, 2009, version 1.4.1., URL: http://www.typo.zib.de/opt-long_projects/Software/Porta/index.html.
- [3] Edmonds, J., *Maximum matching and a polyhedron with 0,1 vertices*, Journal of Research of the National Bureau of Standards Sect B 69 (1965), 125–130.
- [4] Glaab, H., “Eine Variante des Travelling Salesman Problems mit mehreren Handlungsreisenden: Modelle, Algorithmen und Anwendung,” Ph.D thesis, Universität Augsburg, 2000.
- [5] Glaab, H., and A. Pott, *The Hamiltonian p -Median Problem*, The Electronic Journal of Combinatorics, **7** (2000), <http://www1.combinatorics.org>.
- [6] Gollowitzer, S., and L. Gouveia, and G. Laporte, and D. L. Pereira, and A. Wojciechowski, *A Comparison of Several Models for the Hamiltonian p -Median Problem*, Technical Report (2012).
- [7] Gollowitzer, S., and D. L. Pereira, and A. Wojciechowski, *New Models for and Numerical Tests of the Hamiltonian p -Median Problem*, Lecture Notes in Computer Science, **6701** (2011), 385–394, Proceedings of the 5th International Conference on Network Optimization, INOC 2011.
- [8] Grötschel, M., *Polyedrische Charakterisierung kombinatorischer Optimierungsprobleme*, Mathematical Systems in Economics, **36** (1977).
- [9] Padberg M.W., and M.R. Rao, *Odd Minimum Cut-Sets and b -Matchings*, Mathematics of Operations Research **7**(1) (1982), 67–80.

Mathematical Analysis of caching policies and cooperation in YouTube-like services

Pablo Romero Franco Robledo Pablo Rodríguez-Bocca
Claudia Rostagnol

{promero,frobledo,prbocca,crostag}@fing.edu.uy

Departamento de Investigación Operativa

Universidad de la República

Montevideo, Uruguay

Abstract

Currently, most video on-demand services offered over the Internet do not exploit the idle resources available from end-users. We discuss the benefits of user-assistance in video on-demand systems, where users are both clients and servers, helping with the task of video distribution. The mathematical machinery for the systematic analysis of video on-demand services is not mature yet. In this paper we develop a deterministic fluid model to determine the expected evolution of user-assisted on-demand video streaming services. We theoretically prove that cooperative systems always outperform non-cooperative solutions. A combinatorial optimization problem is proposed, where the goal is to distribute a set of video items into repositories trying to offer the minimum waiting times to end-users. This combinatorial problem is proved to be in the class of NP-Complete computational problem, and is heuristically solved with a GRASP methodology. Predictions inspired in YouTube scenarios suggest the introduction of cooperation is both robust and extremely attractive from an economical viewpoint as well.

Keywords: Video on-demand, Fluid model, Combinatorial Optimization Problem, GRASP.

1 Motivation

Established in 2005, YouTube has become the most successful Internet site providing a new generation of short video sharing service, comprising approximately nearly 10% of all traffic on the Internet [1]. However, the network access is yet working with a client-server architecture, and the operator (Google corporation) must afford more than one million dollars per day just for bandwidth requirements, which is a clear motivation to exploit idle uploading resources from YouTube’s users [4]. Experimental works also converge to the fact that user-assistance offloads the server and provides high scalability to video on-demand systems [3,12]. Peer-to-peer networks represent a promising alternative. The design of efficient scheduling policies must face several challenges: resources are highly dynamic, peers are heterogeneous (they have different broadband connections), and there are non-altruistic peers called free-riders (i.e peers who wish to exploit the resources of the whole system without contributing with it).

Scarce mathematical tools have been developed to understand the trade-offs in the design of user assisted on-demand systems. The first analytical model for a mesh-based P2P-VoD system was introduced in 2008 by Jue Lu et. al. [5]. It is similar in spirit to the classical file-sharing model from Qiu and Srikant [7], but proposed some ingredients inherent to VoD systems, with different file size, variable arrival rates and seeders aborting the system as a function of time, which makes it non-linear.

Our goal is to mathematically study the stability and performance of assisted on-demand services versus raw client-server architectures. Sequential systems represent an outstanding service, where users can download several video items serially (not simultaneously). A more ambitious service is the concurrent one, where users keep simultaneous downloads. For the sake of brevity, we will concentrate on the sequential service. It is important to emphasize that sequential systems possess the major practical interest (human beings do not watch two videos simultaneously). The reader can find an in-depth analysis of concurrent systems in [10]. This paper is structured as follows. Section 2 presents assisted and traditional sequential fluid models. Related theoretical results are summarized in Section 3. A caching problem is presented in Section 4, where the issue is to define the video items that should be stored in repositories to offer minimal waiting times to end-users. In Section 5, we take statistical information with a passive YouTube-crawler and compare our caching solution versus a traditional client-server architecture. Finally, concluding remarks and technological concerns are discussed.

2 Fluid models for on-demand video streaming

Consider an open system with K video items with sizes $\{s_1, \dots, s_K\}$. Peers join the network, download video items sequentially and abort the system when they wish. Denote $x_j(t)$ the number of peers downloading video j in the current time t . They join the network following a poissonian process with rates λ_j , and abort the system with exponential law with rate θ . The number of peers seeding video j at instant t is denoted by $y_j(t)$, and depart the system exponentially with rates γ . We shall assume identical peers, with respective upload and download capacities denoted by μ and c . The exchange effectiveness between peers is a coefficient $\eta : 0 \leq \eta \leq 1$. Super-peers behave like seeders, but they do not leave the system. The number of super-peers seeding video j are denoted by z_j , and have upload capacity ρ . Super-peers as well as seeders upload video streams with fairness, and peers apply an incentive-based policy: their upload capacity is linearly related with the level of altruism. All this information can be summarized in a *Sequential Fluid Model (P2P-SFM)*, specified as follows:

$$\left\{ \begin{array}{l} \frac{dx_j(t)}{dt} = \lambda_j - \theta x_j(t) - \min \left\{ \frac{c}{s_j} x_j(t), \frac{\mu}{s_j} (\eta x_j(t) + y_j(t)) + \frac{\rho}{s_j} z_j \right\} \end{array} \right. \quad (1a)$$

$$\left\{ \begin{array}{l} \frac{dy_j(t)}{dt} = \min \left\{ \frac{c}{s_j} x_j(t), \frac{\mu}{s_j} (\eta x_j(t) + y_j(t)) + \frac{\rho}{s_j} z_j \right\} - \gamma y_j(t) \end{array} \right. \quad (1b)$$

The *P2P-SFM* is just a balance of entrance and departure of peers in a fair system, including the presence of special components like seeders and super-peers, and cooperation in a BitTorrent fashion (the reader can find more general models in [10]). As soon as peers completely download the video stream, they are promoted to seeders, explaining the additive term on the right hand of (1b). The minimum function means that the bottleneck is either in the cumulative download or upload. This model represents an extension of [11], and was vastly studied in [9,10].

A traditional CDN (with a client-server paradigm) can be viewed as a particular case of this analytical approach. Specifically, users do not cooperate ($\mu = 0$) and super-peers are now static servers. Replacing $\mu = 0$ in (1a), the *Content Delivery Network-Sequential Fluid Model (CDN-SFM)* is defined by:

$$\frac{dx_j(t)}{dt} = \lambda_j - \theta x_j(t) - \min \left\{ \frac{c}{s_j} x_j(t), \rho_j z_j \right\} \quad (2)$$

3 Stability and Performance Analysis

The *P2P-SFM* is a linear switched system of ordinary differential equations. By algebraic means, the rest-point can be found forcing $\frac{dx_j(t)}{dt} = \frac{dy_j(t)}{dt} = 0$:

$$\bar{x}_{j,SFM}^{P2P} = \max \left\{ \frac{\lambda_j s_j}{\theta s_j + c}, \frac{\lambda_j (\lambda_j - \rho_j z_j - \frac{\mu \lambda_j}{\gamma s_j})}{\lambda_j (\theta + \frac{\eta \mu}{s_j} - \frac{\mu \theta}{\gamma s_j})} \right\}, \quad \bar{y}_{j,SFM}^{P2P} = \frac{\lambda_j - \theta \bar{x}_{j,SFM}^{P2P}}{\gamma}. \quad (3)$$

Another reason to study sequential systems is the following result:

Theorem 3.1 *The P2P-SFM is always globally stable (proof in [10]).*

The proof is long and tricky, in a large way inspired by the proof from Dongyu Qiu and Wei Qian Sang [6], where the authors study a very similar system, but with a single file and no super-peer assistance. Theorem 3.1 is perhaps surprising, because we can tune all the parameters of the system and it will always converge to the rest-point. Denote T_{SFM}^{P2P} and T_{SFM}^{CDN} the expected waiting times under regime for the respective systems *P2P-SFM* and *CDN-SFM*. A valuable corollary from Theorem 3.1 and Little's law is the following:

Theorem 3.2 $T_{SFM}^{P2P} \leq T_{SFM}^{CDN}$

Proof. The rest-point for the *CDN-SFM* is just $\bar{x}_{j,SFM}^{CDN} = \bar{x}_{j,SFM}^{P2P}|_{\mu=0}$. Given that $\bar{x}_{j,SFM}^{P2P}$ is monotonically decreasing with μ , we get that $\bar{x}_{j,SFM}^{P2P} \leq \bar{x}_{j,SFM}^{CDN}$, and the equality holds only when both rest-points do not depend on μ . Denote $\lambda = \sum_j \lambda_j$ the sum rate. By Little's law the following chain of inequalities holds: $T_{SFM}^{P2P} = \sum_j \frac{\bar{x}_{j,SFM}^{P2P}}{\lambda} \leq \sum_j \frac{\bar{x}_{j,SFM}^{P2P}}{\lambda} = T_{SFM}^{CDN}$. \square

So far, we know peer-assisted sequential systems are always stable and outperform raw client-server architectures. In the following section we will further explore the peer-assisted benefits, with scope in a major cause of concern in on-demand peer-to-peer systems: the video caching policies, where special nodes should store video items with constrained resources, in order to minimize the expected waiting time T_{SFM}^{P2P} .

4 Caching Problem

Let P be the number of super-peers (servers) in the system, u_n be the unit column vector of n elements (all its entries are 1), $s = (s_1, \dots, s_K)^t$ the video sizes, and $S = (S_1, \dots, S_P)^t$ the super-peers' storage capacity. The decision

variable is a boolean matrix E of size $P \times K$, whose entries are $E(p, j) = 1$ if and only if we store video item j in super-peer p . We study the worst cooperative scenario, where seeders are selfish ($\gamma = \infty$). Then, we define the *Caching Problem* in matrix form as follows:

$$\min_E \sum_{j=1}^K \max \left\{ \frac{\lambda_j s_j}{\theta s_j + c}, \frac{\lambda_j s_j - \rho z_j}{\theta s_j + \eta \mu} \right\}$$

$$s.t. \begin{cases} E \times s \leq S & (4a) \\ E^t \times u_P = z & (4b) \\ z \geq 2u_K & (4c) \\ E(p, j) \in \{0, 1\}, \forall p \in [P], j \in [K]. & (4d) \end{cases}$$

The objective is to minimize the mean waiting times of all users in the system. It is worth to notice that the Caching Problem is useful for both *SFM – P2P* and *SFM – CDN* systems (where $\mu = 0$ in the latter). Constraint (4a) states that super-peers' storage capacity cannot be exceeded. Constraint (4b) relates the number of replicas z_j for video item j with the matrix E . Constraint (4c) imposes that each video item must be available in the network at least twice, whereas Constraint (4d) states E is a boolean matrix. A solution is feasible whenever all the previous constraints hold.

Theorem 4.1 *CACHING – FEASIBILITY* is NP-Complete.

Proof. We can decide in polytime whether the solution is feasible or not, so *CACHING – FEASIBILITY* is in NP. We will show that if we can determine feasibility of an arbitrary instance of the Caching Problem in polynomial time, then we would solve *PARTITION* in polynomial time as well. Recall *PARTITION* is an NP-Complete decision problem, where the issue is to find a subset of integers that sum the half of the global sum [2]. Consider an arbitrary set of positive integers $A = \{a_1, \dots, a_n\}$ and let us call $a_{sum} = \sum_{i=1}^n a_i$. We construct the following Caching Problem instance, with $P = 3$, $S_1 = S$, $S_2 = S_3 = a_{sum}/2$ and $s_j = a_j$ for all $j \in [n]$. This transformation is polynomial. Since $S_1 + S_2 + S_3 = 2 \sum_{i=1}^n a_i$, Constraint (4c) forces the three super-peers to store video items at their full capacity. Therefore, a feasible solution complies that $S_2 = \sum_B a_i = \sum_{B^C} a_i = S_3 = a_{sum}/2$, for a certain $B \subset A$. As a consequence, if *CACHING – FEASIBILITY* can be solved for every instance in polynomial time, then every instance of *PARTITION* can be solved in polynomial time as well. This concludes that *CACHING – FEASIBILITY* is an NP-Complete decision problem. \square

We solved the Caching Problem following the GRASP (Greedy Randomized Adaptive Search Procedure) metaheuristic [8]. The algorithm works as follows. We first sort the content respect to a cost-to-benefit ratio $w_i = s_i \bar{x}_i^{P2P}$, and iteratively pick the two super-peers with highest remaining resources to store these videos, respecting Constraint 4a, until Constraint 4c holds. Therefore, a feasible solution is found (provided that superpeers are resourceful), and a classical local search is applied to delete, add or swap videos into super-peers, until a local minimum waiting time is finally returned. The reader is referred to [9,10] for details. In the remainder, the GRASP algorithm is applied in both systems ($P2P - SFM$ and $CDN - SFM$), with real-life instances taken from YouTube.

5 Results and Conclusions

5.1 Application to a real scenario: YouTube

We further explore the performance and robustness properties in real-life scenarios for both P2P and CDN architectures, under the naive GRASP caching algorithm. For that purpose, we captured statistical information of nearly 60.000 video files using a passive YouTube crawler, in order to contrast P2P network with raw CDN technology. We use selfish seeders ($\gamma = \infty$)

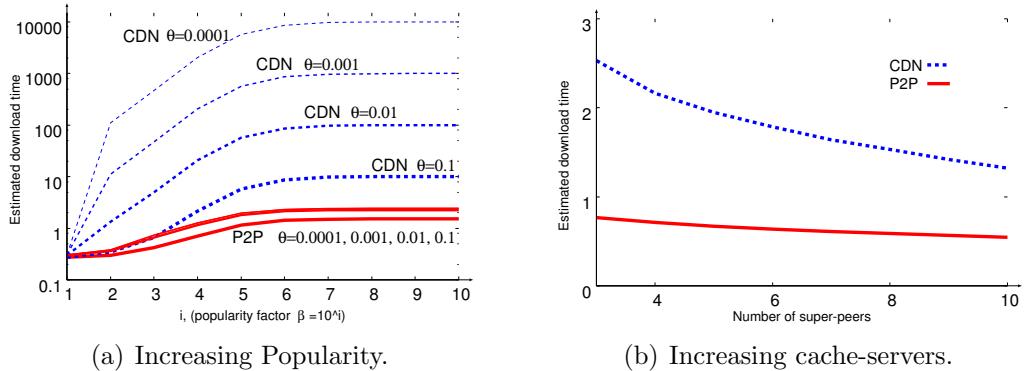


Fig. 1. Download time for CDN and P2P.

and medium effectiveness $\eta = 0, 5$, and perform a statistical estimation of video popularities λ_j , stressing the system with an artificial popularity factor $\beta = 10^m$ ($m \in \{1, \dots, 10\}$) to the vector $(\lambda_1, \dots, \lambda_K)$. In this way, we can contrast the performance of a CDN vs P2P deployment under different scenarios. We use abortion rates $\theta = 10^{-n}$ with $n \in \{1, 2, 3, 4\}$, a common download rate of $c = 1$ Megabytes per second, $\mu = c/4$ following asymmetrical

broadband accesses from end-users, and $P = 4$ super-peers (or servers) with $\rho = 10$ Megabytes per second each, holding $K = 59000$ video items.

Figure 1(a) shows the performance of both P2P and CDN models versus β . It underlines three essential features. First, the expected time for a P2P sequential system is never worse than the one of a raw CDN technology, as predicted by Theorem 3.2. Second, the performance of both systems presents no gap for low-populated scenarios, but peer savings are remarkable under high-populated ones. Third, the expected waiting time is highly sensitive to abortions in a CDN system, whereas the P2P system shows to be more stable, outstanding its high scalability.

Figure 1(b) illustrates the expected waiting time for both P2P and CDN systems (with solid and dashed lines respectively) versus P , when $\beta = 10^3$ and $\theta = 0, 1$. It helps to figure-out robustness: how the system's performance can be affected in terms of single point of failures. For a fixed popularity factor we want to find the mean waiting time for different number of super-peers (servers). From this experiment, we conclude that P2P systems are resilient under environmental failures, while a raw CDN can be considerably damaged when a server fails.

5.2 Conclusions

In this paper, a general framework for the analysis of sequential video on-demand assisted services is provided. Under this framework of expected evolution, the sequential system is always globally stable, converging to a known rest-point. We found closed expressions for the expected waiting times in both CDN and P2P approaches, and we theoretically proved that the peer-to-peer philosophy always outperforms traditional CDNs. An experimental validation of the P2P and CDN systems and their performance is presented regarding real-traces passively taken from a YouTube crawler. The results are encouraging, showing that a P2P assisted platform preserves its resilience against adverse environments like flash crowds. Our trends for future work include the stability and capacity analysis in concurrent video on-demand assisted scenarios.

References

- [1] Cheng, X., Dale, C., Liu, J.: Understanding the Characteristics of Internet Short Video Sharing: YouTube as a Case Study. Tech. rep., Cornell University (2007)

- [2] Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-Completeness. W. H. Freeman & Company, New York, NY, USA (1979)
- [3] Hefeeda, M., Bhargava, B.K., Yau, D.K.Y.: A hybrid architecture for cost-effective on-demand media streaming. *Computer Networks* **44**(3), 353–382 (2004)
- [4] Huang, C., Li, J., Ross, K.W.: Can internet video-on-demand be profitable? *SIGCOMM Computer Communication Review* **37**(4), 133–144 (2007)
- [5] Lu, Y., Mol, J.D., Kuipers, F., Mieghem, P.V.: Analytical Model for Mesh-Based P2PVoD. In: Proceedings of the 2008 Tenth IEEE International Symposium on Multimedia, ISM '08, pp. 364–371. IEEE Computer Society, Washington, DC, USA (2008). URL <http://dx.doi.org/10.1109/ISM.2008.30>
- [6] Qiu, D., Sang, W.: Global stability of Peer-to-Peer file sharing systems. *Comput. Commun.* **31**(2), 212–219 (2008)
- [7] Qiu, D., Srikant, R.: Modeling and Performance Analysis of BitTorrent-Like Peer-to-Peer Networks. In: Proceedings of SIGCOMM'04, pp. 367–378. ACM, New York, NY, USA (2004). DOI <http://doi.acm.org/10.1145/1015467.1015508>
- [8] Resende, M.G.C., Ribeiro, C.C.: Greedy Randomized Adaptive Search Procedures. In F. Glover and G. Kochenberger, editors, *Handbook of Metaheuristics*, Kluwer Academic Publishers (2003)
- [9] Robledo, F., Rodríguez-Bocca, P., Romero, P., Rostagnol, C.: A new caching policy for cloud assisted peer-to-peer video-on-demand services. In: Proceedings of the 12th IEEE International Conference on Peer-to-Peer Computing (IEEE P2P'12) (2012)
- [10] Romero, P.: Mathematical analysis of scheduling policies in Peer-to-Peer video streaming networks. Ph.D. thesis, Facultad de Ingeniería, Universidad de la República, Montevideo, Uruguay (2012). URL <http://www.fing.edu.uy/inco/pedeciba/bibliote/tesis/tesisd-romero.pdf>
- [11] Tian, Y., Wu, D., Ng, K.W.: Analyzing Multiple File Downloading in BitTorrent. In: Proceedings of ICPP'06, pp. 297–306. IEEE (2006). DOI 10.1109/ICPP.2006.23
- [12] Yu, H., Zheng, D., Zhao, B.Y., Zheng, W.: Understanding user behavior in large-scale video-on-demand systems. *SIGOPS Oper. Syst. Rev.* **40**(4), 333–344 (2006)

Stronger Lower Bounds for the Quadratic Minimum Spanning Tree Problem with Adjacency Costs

Dilson Lucas Pereira ^{a,1,3}, Michel Gendreau ^{b,4},
Alexandre Salles da Cunha ^{c,2,5}

^a CAPES Foundation, Ministry of Education of Brazil, Brasília - DF 70.040-020,
Brazil

^b École Polytechnique de Montréal, Montréal - Canada

^c Departamento de Ciéncia da Computação, Universidade Federal de Minas
Gerais, Belo Horizonte - MG, Brazil

Abstract

We address a particular case of the quadratic minimum spanning tree problem in which interaction costs only apply for adjacent edges. Motivated by the fact that Gilmore-Lawler procedures in the literature underestimate the contribution of interaction costs to compute lower bounds, we introduce a reformulation that allows stronger linear programming bounds to be computed. An algorithm based on dynamic column and row generation is presented for evaluating these bounds. Our computational experiments indicate that the reformulation introduced here is indeed much stronger than those in the literature.

Keywords: quadratic 0-1 programming, spanning trees, column generation

1 Introduction

Given a connected, undirected graph $G = (V, E)$, with $n = |V|$ vertices and $m = |E|$ edges, and a matrix $Q = (q_{ij} \geq 0)_{i,j \in E}$ of interaction costs, the quadratic minimum spanning tree problem (QMSTP) asks for a spanning tree of G whose incidence vector $\mathbf{x} \in \mathbb{B}^m$ minimizes $\sum_{i,j \in E} q_{ij}x_i x_j$. In this work, we focus on a particular case of QMSTP where costs only apply for adjacent edges, i.e., $q_{ij} = 0$ if i and j do not share an endpoint.

To our knowledge, this variant was first studied by Assad and Xu [1], where it was named the adjacent only QMSTP (AQMSTP). The authors proved that both QMSTP and AQMSTP are NP-Hard. Lower bounding procedures and branch-and-bound (BB) algorithms were devised. The general case of the QMSTP was also investigated by Cordone and Passeri [2] and Öncan and Punnen [4]. A BB algorithm is presented in the former, while only a lower bounding procedure is presented in the latter. Applications for the QMSTP and the AQMSTP can be found in hydraulic, communication, and transportation networks [1].

Motivated by the fact that Gilmore-Lawler procedures in the literature underestimate the contribution of interaction costs to compute lower bounds, we introduce a reformulation that allows stronger linear programming (LP) bounds to be computed. An algorithm based on dynamic column and row generation is presented for evaluating them.

2 Formulations

2.1 A QMSTP formulation coming from the literature

In order to formulate QMSTP as a binary linear program, consider binary decision variables $\mathbf{x} = (x_i)_{i \in E}$ to determine whether ($x_i = 1$) or not ($x_i = 0$) an edge $i \in E$ appears in the solution. Consider also linearization variables $\mathbf{y} = (\mathbf{y}_i)_{i \in E}$, $\mathbf{y}_i = (y_{ij})_{j \in E}$, to represent the product $x_i x_j$, $i, j \in E$. We use the following notation throughout the paper. Given $S \subseteq V$, $E(S) \subseteq E$ denotes the edges with both endpoints in S and $\delta(S)$ denotes the edges with only one endpoint in S . If S has only one vertex, say v , we replace $\delta(\{v\})$ by

¹ Dilson Lucas Pereira was funded by CAPES, BEX 2418/11-8.

² Alexandre Salles da Cunha is partially funded by CNPq grants 302276/2009-2, 477863/2010-8 and FAPEMIG grant PRONEX APQ-01201-09

³ Email: dilson@dcc.ufmg.br

⁴ Email: michel.gendreau@cirrelt.ca

⁵ Email: acunha@dcc.ufmg.br

$\delta(v)$. X denotes the set of all incidence vectors of spanning trees of G and $X_0 = X \cup \{\mathbf{0}\}$.

The following QMSTP formulation was introduced by Assad and Xu [1],

$$\min \sum_{i,j \in E} q_{ij} y_{ij} \quad (1)$$

$$\text{s.t.} \quad \mathbf{x} \in X, \quad (2)$$

$$\mathbf{y}_i \in X_0, \quad i \in E, \quad (3)$$

$$y_{ii} = x_i, \quad i \in E, \quad (4)$$

$$\sum_{j \in E} y_{ji} = (n - 1)x_i, \quad i \in E. \quad (5)$$

Consider a relaxation for the QMSTP model above, where constraints (5) are dropped from the formulation. To obtain an optimal solution $(\tilde{\mathbf{x}}, \tilde{\mathbf{y}}) \in \mathbb{B}^{m+m^2}$ to such a relaxation, one proceeds as follows. For each $i \in E$, solve

$$\bar{q}_i = \min \left\{ \sum_{j \in E} q_{ij} y_{ij} : \mathbf{y}_i \in X, y_{ii} = 1 \right\}, \quad (6)$$

and let $\bar{\mathbf{y}}_i \in \mathbb{B}^m$ be its solution. Then, find the spanning tree implied by

$$\bar{\mathbf{x}} \in \arg \min \left\{ \sum_{i \in E} \bar{q}_i x_i : \mathbf{x} \in X \right\}. \quad (7)$$

Once (7) is solved, set $\tilde{\mathbf{x}} = \bar{\mathbf{x}}$ and $\tilde{\mathbf{y}}_i = x_i \bar{\mathbf{y}}_i$, $i \in E$. This process is precisely the well known Gilmore-Lawler [3] procedure, applied for QMSTP.

Assad and Xu [1] devised a Lagrangian relaxation algorithm in which after relaxed, Lagrangian multipliers are attached to constraints (5). A similar scheme was employed by Öncan and Punnen [4]. They relax both (5) and the QMSTP valid inequalities

$$\sum_{i \in \delta(v)} y_{ij} \geq x_j, \quad j \in E, v \in V. \quad (8)$$

In their approach, multiplier adjustment is only implemented for (8).

Of course, the lower bounding procedure outlined above can be applied to AQMSTP. However, Gilmore-Lawler lower bounds, even with multipliers adjustment, tend to be much weaker when applied to AQMSTP. The reason is the following. When solving (6) for an edge $i = \{u, v\} \in E$, the optimal solution $\bar{\mathbf{y}}_i$ will be implied by a spanning tree in which only one edge adjacent to i is selected, namely the edge $j \in \delta(u) \cup \delta(v)$ with the smallest q_{ij} cost. This fact

suggests how underestimated the costs \bar{q}_i may be for the computation of the Gilmore-Lawler lower bound. Next, we present a binary linear programming reformulation for AQMSTP that attempts to overcome this drawback.

2.2 A reformulation for AQMSTP

Let a star of G centered at $v \in V$ be any non-empty subset of edges of $\delta(v)$. Denote by \mathcal{S}_v the set of all stars of G centered at $v \in V$, let $\mathcal{S} = \bigcup_{v \in V} \mathcal{S}_v$, and consider the vector of binary decision variables $\mathbf{t} = (t_H)_{H \in \mathcal{S}}$, used to determine whether ($t_H = 1$) or not ($t_H = 0$) a star $H \in \mathcal{S}$ is included in a AQMSTP solution. Consider also the variables \mathbf{x} defined before. A reformulation for AQMSTP is given by

$$(F_{\text{star}}) \quad \min \sum_{H \in \mathcal{S}} q_H t_H \quad (9)$$

$$\text{s.t.} \quad \sum_{H \in \mathcal{S}_v} t_H = 1, \quad v \in V, \quad (10)$$

$$\sum_{H \in \mathcal{S}_u : i \in H} t_H = \sum_{H \in \mathcal{S}_v : i \in H} t_H, \quad i = \{u, v\} \in E, \quad (11)$$

$$x_i = \sum_{H \in \mathcal{S}_v : i \in H} t_H, \quad i = \{u, v\} \in E, \quad (12)$$

$$\mathbf{x} \in X, \quad (13)$$

$$\mathbf{t} \in \mathbb{B}^{|\mathcal{S}|}. \quad (14)$$

In the reformulation above, $q_H = \sum_{i,j \in H : i \neq j} q_{ij} + (1/2) \sum_{i \in H} q_{ii}, H \in \mathcal{S}$. For stating constraints (11) and (12), we assume $u < v$, for an edge $i = \{u, v\} \in E$.

Constraints (10) guarantee that exactly one star centered at v must be selected. Edges in this star connect v to its neighbors in the tree we are looking for. Constraints (11) impose that, if an edge $\{u, v\}$ appears in the star centered at v , it must also appear in the star centered at u . Constraints (12) couple \mathbf{x} and \mathbf{t} variables. Finally, constraints (13) impose that the set of selected edges must imply a spanning tree of G .

3 Algorithms

We assume that X is represented by the extreme points of the integral polytope implied by $\{\sum_{i \in E} x_i = n - 1, 0 \leq x_i \leq 1, i \in E\}$ and the exponentially many

subtour elimination constraints (SEC)

$$\sum_{i \in E(S)} x_i \leq |S| - 1, \quad S \subset V, |S| > 2. \quad (15)$$

Formulation F_{star} thus involves exponentially many rows and columns, one for each star centered at $v \in V$. We now describe how its LP bounds are evaluated by a dynamic generation of columns and cutting planes (15). To that aim, assume that a restricted set of constraints (15) and stars are present in a restricted linear programming master program (RLMP) that derives from F_{star} , after replacing (14) by $\mathbf{t} \geq \mathbf{0}$. New columns and cuts are added to RLMP in rounds of column and row generation. We solve the first RLMP and then proceed with column generation. When no stars with negative reduced cost are found, we separate (15). If violated SECs are found, they are appended into a new reinforced RLMP, which is re-optimized. A new round of column generation then follows. The procedure stops when no violated cuts nor columns with negative reduced cost are found. At that moment, the LP relaxation of F_{star} is solved.

The separation of (15) is conducted by means of the exact separation algorithm of Padberg and Wolsey [5]. For solving the pricing problems, assume that optimal dual variables $\hat{\boldsymbol{\alpha}} = (\hat{\alpha}_v)_{v \in V}$, $\hat{\boldsymbol{\beta}} = (\hat{\beta}_i)_{i \in E}$, and $\hat{\boldsymbol{\gamma}} = (\hat{\gamma}_i)_{i \in E}$, respectively assigned to constraints (10), (11), and (12), are available after RLMP is solved. For a given $v \in V$, the associated pricing problem consists in finding a star $H \in \mathcal{S}_v$ that violates the constraint

$$q_H - \hat{\alpha}_v - \sum_{i=\{v,u\} \in H} \hat{\beta}_i + \sum_{i=\{u,v\} \in H} \hat{\beta}_i + \sum_{i=\{u,v\} \in H} \hat{\gamma}_i \geq 0. \quad (16)$$

For each $v \in V$, let $\mathbf{w}_v = (w_{vi})_{i \in \delta(v)}$ be a vector of binary variables. The pricing problem for $v \in V$ is formulated as the following unconstrained quadratic binary program

$$\min \left\{ \sum_{i,j \in \delta(v)} q'_{ij} w_{vi} w_{vj} : \mathbf{w}_v \in \mathbb{B}^{|\delta(v)|} \right\} - \hat{\alpha}_v. \quad (17)$$

For i and $j \in \delta(v)$, the following costs q'_{ij} are used. If $i \neq j$, $q'_{ij} = q_{ij}$. If $i = j$, let $\eta(v)$ be the neighbor of v in i . We have two cases: (i) if $v > \eta(v)$, $q'_{ii} = q_{ii} + \hat{\beta}_i + \hat{\gamma}_i$ and (ii) if $v < \eta(v)$, $q'_{ii} = q_{ii} - \hat{\beta}_i$.

3.1 Solving the pricing problems

To solve the pricing problem assigned to $v \in V$, we first linearize (17) and then implement a cut-and-branch algorithm. To that aim, consider the introduction of linearization variables $\mathbf{z}_v = (\mathbf{z}_{vi})_{i \in \delta(v)}$, where $\mathbf{z}_{vi} = (z_{vij})_{j \in \delta(v):j>i}$, $i \in \delta(v)$ ($z_{vji} := z_{vij}$ if $j > i$). A linear integer program (IP) for the pricing problem is

$$\min \sum_{i,j \in \delta(v):i \neq j} q'_{ij} z_{vij} + \sum_{i \in \delta(v)} q'_{ii} w_{vi} - \hat{\alpha}_v \quad (18)$$

$$\text{s.t.} \quad z_{vij} \leq w_{vi}, \quad i < j \in \delta(v), \quad (19)$$

$$z_{vij} \leq w_{vj}, \quad i < j \in \delta(v), \quad (20)$$

$$w_{vi} + w_{vj} - 1 \leq z_{vij}, \quad i < j \in \delta(v), \quad (21)$$

$$(\mathbf{w}_v, \mathbf{z}_v) \in \mathbb{B}^{|\delta(v)|} \times \mathbb{R}_+^{(|\delta(v)|^2 - |\delta(v)|)/2}. \quad (22)$$

Note that $q'_{ij} \geq 0$ for $i \neq j \in \delta(v)$ and that we are interested in stars with negative reduced cost. Given any integer feasible solution $(\bar{\mathbf{w}}_v, \bar{\mathbf{z}}_v)$ for (18)-(22), if for a $i \in \delta(v)$, $\bar{w}_{vi} = 1$ and $q'_{ii} \geq 0$ or $\sum_{j \in \delta(v):j \neq i} (q'_{ij} + q'_{ji}) \bar{w}_{vj} \geq |q'_{ii}|$, then setting $\bar{w}_{vi} = 0$ cannot yield a solution with worse cost. This observation implies the optimality cut

$$\sum_{j \in \delta(v):j \neq i} (q'_{ij} + q'_{ji}) z_{vij} \leq |q'_{ii}| w_{vi}. \quad (23)$$

If, for a set $S \subseteq (\delta(v) \setminus \{i\})$, $\sum_{j \in S} (q'_{ij} + q'_{ji}) \geq |q'_{ii}|$, we also have the cut

$$w_{vi} + \sum_{j \in S} w_{vj} \leq |S|. \quad (24)$$

Our cut-and-branch algorithm for the resolution of (18)-(22) is initialized with (23) and some cuts of type (24), obtained as follows. For each $i \in \delta(v)$ with $q'_{ii} < 0$ we sort the elements of $\delta(v) \setminus \{i\}$ as $\{e_1, \dots, e_{|\delta(v)|-1}\}$ in a non-decreasing order of their costs $q'_{ij} + q'_{ji}$. Then, let \bar{k} be the minimum k for which $\sum_{l=1}^k (q'_{ie_l} + q'_{e_l i}) \geq |q'_{ii}|$. We add $w_{vi} + \sum_{l=1}^{\bar{k}} w_{vj} \leq \bar{k}$ and also

$$\sum_{j \in \delta(v) \setminus \{i\}} z_{vij} \leq (\bar{k} - 1) w_{vi} \quad (25)$$

to the formulation. It should be remarked that these cuts are not globally valid. However, at least one optimal solution for (18)-(22) is satisfied by them.

4 Computational experiments and future work

Instances considered in our computational study for AQMSTP were generated as follows. For each QMSTP instance in [4], one corresponding AQMSTP instance was generated by setting $q_{ij} = 0$, whenever edges i and j do not share one endpoint. Our computational experiments were conducted with a 3GHz Intel Core 2 Duo machine, with 5GB of RAM memory, under Linux operating system. The IP solver CPLEX 12 was used to solve the pricing problems. CPLEX LP solver was also used to solve each RLMP.

In Table 1, we compare three lower bounds for AQMSTP. The first two are Lagrangian relaxation bounds: $Lag_{(5)}$ (obtained by [1], by relaxing (5)) and $Lag_{(8)}$ (obtained in [4], by relaxing (8)). The last bound is $LP(F_{\text{star}})$, the LP relaxation bound implied by F_{star} . All lower bounds coming from the literature were evaluated by our own implementation of their procedures. In the first three columns of the table, we provide n, m , and ub , an upper bound on the optimum (obtained by running a simple greedy heuristic). For each lower bounding scheme, we provide the lower bound lb , the duality gap (computed with ub) and the time t , in seconds, needed to evaluate the bound. In the table we present results only for some selected instances; the ones with the smallest and the largest duality gaps, for each size n, m .

Compared to the bounds in the literature, all of them being of the Gilmore-Lawler type, $LP(F_{\text{star}})$ is remarkably stronger. The computational effort involved in the evaluation of $LP(F_{\text{star}})$, however, is much higher.

We plan to implement a BB algorithm based on formulation F_{star} , and compare it to existing algorithms. One line of research is to investigate ways of speeding up the resolution of the pricing subproblems, which have structure to be explored. That could be accomplished, for example, by reinforcing (18)-(22) with valid inequalities for the boolean quadric polytope.

References

- [1] A. Assad and W. Xu. The quadratic minimum spanning tree problem. *Naval Research Logistics (NRL)*, 39(3):399–417, 1992.
- [2] R. Cordone and G. Passeri. Heuristic and exact approaches to the quadratic minimum spanning tree problem. In *Seventh Cologne-Twente Workshop on Graphs and Combinatorial Optimization*, 2008.
- [3] Y. Li, P.M. Pardalos, K.G. Ramakrishnan, and M.G.C. Resende. Lower bounds

Inst.			$Lag_{(5)}$			$Lag_{(8)}$			LP(F_{star})		
n	m	ub	lb	gap	t	lb	gap	t	lb	gap	t
10	45	205	136.26	33.52	0	122.55	40.21	0.53	205	0	1.22
10	45	257	145.14	43.52	0	135.56	47.24	0.52	242.8	5.52	1.38
20	190	304	101.51	66.6	0.01	103.16	66.06	8.26	302	0.65	83.17
20	190	386	128.9	66.6	0.01	138.7	64.06	8.36	357.1	7.48	82.3
30	435	416	137.84	66.86	0.06	139.88	66.37	46.84	402.92	3.14	2214.73
30	435	473	130.16	72.48	0.06	130.29	72.45	44.26	425.33	10.07	2024.5
10	45	17608	15502.1	11.95	0	15149	13.96	0.53	17608	0	1.66
10	45	17046	14984.27	12.09	0	14386.66	15.6	0.52	15921	6.59	1.32
20	190	22724	15539.27	31.61	0.01	15726	30.79	8.25	22678.5	0.2	192
20	190	21181	13270.65	37.34	0.02	12764.42	39.73	8.25	19688	7.04	209.48
30	435	28059	17963.14	35.98	0.07	18040.18	35.7	46.52	28020.75	0.13	22487.21
30	435	27553	14742.15	46.49	0.02	15027.62	45.45	45.77	25472.25	7.55	18272.15
10	45	538	305	43.3	0	322.29	40.09	0.53	486.89	9.49	2.16
10	45	512	293.88	42.59	0	270.5	47.16	0.51	368.2	28.08	1.84
20	190	17889	17799.27	0.5	0	17187	3.92	0.07	17889	0	0.11
20	190	17703	14884.6	15.92	0	13056.5	26.24	0.09	17468	1.32	0.16
30	435	945	490.93	48.04	0.07	506.09	46.44	60.31	783.61	17.07	2522.87
30	435	1071	507.03	52.65	0.06	497.17	53.57	44.52	776.12	27.53	2512.09

Table 1
Comparison of lower bounds

for the quadratic assignment problem. *Annals of Operations Research*, 50:387–410, 1994.

- [4] T. Öncan and A. P. Punnen. The quadratic minimum spanning tree problem: A lower bounding procedure and an efficient search algorithm. *Computers and Operations Research*, 37(10):1762–1773, 2010.
- [5] M. W. Padberg and L. A. Wolsey. Trees and cuts. In *Combinatorial Mathematics Proceedings of the International Colloquium on Graph Theory and Combinatorics*, volume 75, pages 511–517. 1983.

An Interval Assignment Problem for Resource Optimization in LTE Networks

Mohamad Assaad ¹

*Department of Telecommunications, Supelec
Gif sur Yvette, France*

Walid Ben-Ameur²

*Samovar, CNRS UMR 5157
Télécom SudParis, Evry, France*

Faiz Hamid³

*Samovar, CNRS UMR 5157
Télécom SudParis, Evry, France*

Abstract

In this paper we study the problem of resource allocation in SC-FDMA (Single Carrier Frequency Division Multiple Access) which is adopted as the multiple access scheme for the uplink in the 3GPP-LTE (3rd Generation Partnership Project - Long Term Evolution) standard. The problem can be modeled as an assignment problem where each user can be given a subset of consecutive channels seen as an interval. After introducing the problem, we first prove that it is NP-hard to solve. Then we review some cases where the problem is solvable in polynomial time. An efficient cutting plane algorithm is presented with experimental results.

Keywords: resource allocation, integer programming, polyhedra, complexity

1 Introduction

To meet the requirements of fast-growing demand for wireless services, LTE standard has been adopted for the future generation of wireless cellular networks. In this standard, OFDMA (Orthogonal Frequency Division Multiple Access) and SC-FDMA are used as multiple access schemes respectively for downlink (base station to users) and uplink (users to base station) communications. In this system, a sub-channel is formed by grouping together 12 sub-carriers [1]. Each sub-channel allows transmission at a given data rate and a user may be allocated a subset of sub-channels either scattered over the whole bandwidth or adjacent to each other. Therefore, the allocation of sub-channels to the users has an impact on the overall transmission data rate of the system. The problem of resource allocation (sub-channels allocation) in OFDMA and SC-FDMA systems is therefore an active area of research. In OFDMA systems, the users are orthogonal in the sense that two users cannot share the same sub-channel at a time. In SC-FDMA, in addition to users' orthogonality restriction, the multiple sub-channels allocated to a user should be adjacent to each other [1]. This makes the resource allocation problem in the LTE uplink (i.e. SC-FDMA) more challenging.

The resource allocation problem in uplink SC-FDMA has been addressed in a number of publications. Heuristic greedy suboptimal schedulers for allocating frequency bands to the users are developed in [3],[6]. In [7], a weighted-sum rate maximization is considered and a greedy sub-optimal algorithm is proposed. In all the previous works, the resource allocation problem in SC-FDMA systems has been studied heuristically without any theoretical formulation or proof of NP hardness. In this paper, we first prove that the SC-FDMA resource allocation problem is NP hard and then propose an efficient cutting plane method to solve it.

The rest of this paper is organized as follows. Sections 2 provides the system model and problem formulation. The problem complexity is discussed in Section 3 and some polynomial cases of the problem are presented in Section 4. After a brief description of the solution approach in Section 5, we describe our computational study in Section 6. Finally, we end up with conclusions and future research directions in the remainder of Section 6.

¹ Email: Mohamad.Assaad@supelec.fr

² Email: walid.benameur@telecom-sudparis.eu

³ Email: faiz.hamid@telecom-sudparis.eu

2 System Model and Problem Formulation

We consider the uplink of a single cell LTE system that utilizes localized SC-FDMA. The generalization to multi-cell scenario is straightforward. Let $K = \{1, 2, \dots, m\}$ be the set of users and $S = \{1, 2, \dots, n\}$ the set of sub-channels. According to [1], a sub-channel cannot be shared by more than one user at a time and the sub-channels assigned to a user are required to be consecutive. We therefore denote by an *interval* $[a, b]$ the subset of sub-channels that are consecutive starting from a and ending at b . Let $U_{a,b}^k$ denote the pay-off obtained by assigning interval $[a, b] \subset S$ to user $k \in K$. The description of the pay-off function lies out of the scope of this paper. In practice, the pay-off function is in general an increasing function with respect to the data rate or SNR of the users and it depends on the Quality of Service required by the users. An example of the method of computing $U_{a,b}^k$ is given in Section 6. The problem can be formulated mathematically as follows:

$$\text{Maximize} \sum_{k \in K} \sum_{[a,b] \subset S} U_{a,b}^k x_{a,b}^k \quad (1)$$

subject to

$$\sum_{k \in K} \sum_{a \leq i \leq b} x_{a,b}^k \leq 1 \quad \forall i \in S \quad (2)$$

$$\sum_{[a,b] \subset S} x_{a,b}^k \leq 1 \quad \forall k \in K \quad (3)$$

$$x_{a,b}^k \in \{0, 1\} \quad \forall k \in K \quad \forall [a, b] \subset S \quad (IA)$$

In the above formulation, the binary variable $x_{a,b}^k = 1$ if interval $[a, b]$ is assigned to user k , otherwise 0. The objective function (1) maximizes the total pay-off by assigning the intervals to the users. Constraint (2) models the requirement that each sub-channel can be assigned to at most one user. Constraint (3) implies each user can be assigned at most one interval. Since the problem described above is an interval assignment problem, it will be denoted by (IA) in the rest of the paper. Next we prove that the problem (IA) is NP-hard.

3 Complexity

Proposition 3.1 (IA) is NP-hard.

Proof. We prove the NP-hardness of (IA) by reduction from the 3-SAT problem with at most 3 occurrences for each variable. This is the Boolean satisfa-

bility problem where each clause contains at most 3 variables and where each variable appears at most in 3 clauses. This variant is also NP-hard.

It is assumed that each variable x_j will occur at least twice and as both x_j and \bar{x}_j . Otherwise, it is always possible to simplify the formula by giving the right truth assignment to x_j (resp. \bar{x}_j) and eliminating all clauses containing x_j (resp. \bar{x}_j). Let T be an instance of this variant of 3-SAT with m clauses c_1, \dots, c_m and n variables x_1, \dots, x_n . To transform the 3-SAT problem into problem (IA), let each clause c_i correspond to one of the users.

We consider any arbitrary ordering of the variables in the clauses to form the sub-channels. Suppose variable x_j appears in clauses c_i and c_k while \bar{x}_j in c_l . The two similar occurrences of x_j form consecutive sub-channels. Three intervals are created, namely $c_i.x_j$, $c_k.x_j$ and $c_l.\bar{x}_j$. Interval $c_i.x_j$ contains the first sub-channel, $c_k.x_j$ contains the second sub-channel while $c_l.\bar{x}_j$ contains both the previous sub-channels as they are consecutive (see Figure 1 for an illustration). The case where x_j appears once and \bar{x}_j appears twice can be modeled in the same way. If a variable x_j appears only twice, once as x_j in c_i (say) and other as \bar{x}_j in c_k (say), then we consider one sub-channel and two similar intervals $c_i.x_j$ and $c_k.\bar{x}_j$ containing the same sub-channel.

Figure 1 gives a layout for the case $n = 6$, $m = 5$ and $T = (x \vee \bar{y} \vee z) \wedge (\bar{x} \vee y \vee \bar{z}) \wedge (\bar{x} \vee y \vee w) \wedge (u \vee \bar{v} \vee \bar{w}) \wedge (\bar{u} \vee v \vee w)$. The clauses/users are represented by the shaded boxes. Each occurrence of the variables/sub-channel is represented by black circle. Intervals are represented by single square brackets. Intervals obtained by combining sub-channels are denoted by bigger square brackets containing the black circles. The dotted lines indicate possible user-interval assignments considering the constraints of problem (IA).

For a valid user-interval assignment, let the pay-off be 1 unit; otherwise it is strictly less than 1. We claim that there is a truth assignment satisfying all clauses of the 3-SAT instance if and only if the solution to the problem (IA) constructed above has pay-off $\geq m$.

Suppose that such a truth assignment exists. Consider a variable x_j appearing in clauses c_i and c_k while \bar{x}_j occurs in c_l . If $x_j = \text{TRUE}$, then we can assign to user c_i (resp. c_k) the interval $c_i.x_j$ (resp. $c_k.x_j$). Notice that these two intervals do not intersect. If $x_j = \text{FALSE}$, then we can assign the interval $c_l.\bar{x}_j$ to user c_l . The situation where x_j appears once in c_i (say) and \bar{x}_j appears once in c_k (say) leads to the assignment of interval $c_i.x_j$ (resp. $c_k.\bar{x}_j$) if $x_j = \text{TRUE}$ (resp. FALSE). It is now clear that each user is assigned an interval. Moreover, all assigned intervals do not intersect. Therefore, the pay-off of the assigned intervals is equal to m .

Let us now assume that the solution of problem (IA) has a pay-off $\geq m$

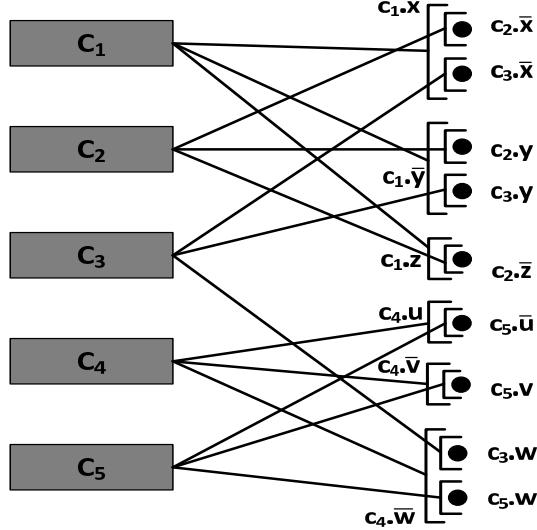


Fig. 1. An instance of 3-SAT transformed to an instance of (IA)

and let us prove that the 3-SAT instance has a truth assignment satisfying all clauses. If an interval of type $c_i \cdot x_j$ is assigned then we take $x_j = \text{TRUE}$, while x_j will be equal to FALSE if an interval of type $c_i \cdot \bar{x}_j$ is assigned. Observe that since the assigned intervals do not intersect, we can never simultaneously assign $c_i \cdot x_j$ and $c_k \cdot \bar{x}_j$. In other words, there is no ambiguity to determine the truth assignment. Using the fact that the pay-off is $\geq m$, we deduce that each user is assigned an interval. Consequently, each clause contains at least one variable having the right assignment to make the clause TRUE. \square

4 Some Polynomial Cases

Although the problem (IA) is NP-hard, we present hereinafter some simple/special cases of this problem that can be solved in polynomial time.

CASE 1: Given any interval, the pay-off is same for all users, i.e., $U_{a,b}^1 = U_{a,b}^2 = \dots = U_{a,b}^m \equiv U_{a,b} \forall [a, b]$. This problem can be solved in polynomial time using dynamic programming approach.

Consider the function $f(k, b)$ representing the maximum pay-off that we can get by assigning to k users sub-channels in the subset $\{1, \dots, b\}$. The following induction formulation is clearly valid:

$$f(k, b) = \max \left\{ f(k, b-1), f(k-1, b), f(k-1, b-1) + U_{b,b}, \dots, f(k-1, 1) + U_{2,b}, U_{1,b} \right\} \quad (4)$$

The optimal solution is obtained by computing $f(m, n)$. The computational complexity is obviously given by $O(mn^2)$.

CASE 2: Given any user k , the pay-off $U_{a,b}^k$ depends only on the length of the interval $(b-a+1)$. In other words, $U_{1,b-a+1}^k = U_{2,b-a+2}^k = \dots = U_{a,b}^k = \dots = U_{n-b+a,n}^k$. As above, the problem can be solved by dynamic programming.

CASE 3: When one user can be assigned more than one interval. The condition implies relaxation of constraint (3). This problem can be phrased as a maximum independent set problem in an interval graph and can be solved in polynomial time using dynamic programming (see, e.g., [2]).

CASE 4: When intervals are assigned to users in some order. In other words, we assume here that if user k is assigned an interval $[a, b]$ and user $k + 1$ is assigned an interval $[c, d]$, then necessarily $c > b$. This case can be easily handled by dynamic programming.

CASE 5: When intervals are already formed. This implies that the intersection of sub-channels between intervals is already taken care of. Then problem (IA) becomes a maximum weight matching problem that can be solved by several polynomial-time algorithms [5].

5 Cutting Plane Algorithm

Let $\text{conv}(IA)$ denote the convex hull of all feasible solutions of problem (IA). We first compute the dimension of $\text{conv}(IA)$. The proof of the propositions are not presented in this paper due to page limitation.

Proposition 5.1 *Polyhedron $\text{conv}(IA)$ is full dimensional.*

Our approach for solving the problem (IA) is to consider its linear relaxation and tighten the upper bound using valid inequalities. Following is a valid inequality that prohibits assignment of intersecting intervals.

$$\sum_{[i,j] \cap [a,b] \neq \emptyset} x_{i,j}^k + \sum_{k' \neq k} \sum_{[c,d] \supset [a,b]} x_{c,d}^{k'} \leq 1 \quad \forall k \in K, \forall [a,b] \subset S \quad (5)$$

Proposition 5.2 *Inequalities (5) define facets of $\text{conv}(IA)$.*

The number of inequalities (5) is polynomial, therefore they can be separated by simple enumeration.

6 Computational Study

The cutting plane algorithm has been tested on several randomly generated problem instances. The key problem parameters identified were: number of users and number of sub-channels. It is obvious that increase in anyone of the problem parameters increase the number of variables and constraints, and makes the problem difficult to solve. For a given set of problem parameters, each experiment was carried out on 100 different instances of the randomly generated problems. Each instance represent a given setting of the wireless network (i.e. users' positions, channel conditions, etc.). The users are uniformly distributed in a macro cell of radius 1km. The pay-off function of each user is the bit rate given by the widely used Shannon capacity formula:

$$U_{a,b}^k = (b - a) * \log(1 + \gamma_{a,b}^k) \text{ where } \gamma_{a,b}^k = \left(\frac{1}{\frac{1}{b-a} \sum_{i \in [a,b]} \frac{P_k G_{k,i}}{(b-a)+P_k G_{k,i}}} - 1 \right)^{-1}$$

This is the Signal to Noise Ratio (SNR) of user k if interval $[a,b]$ is assigned. P_k is the max power of user k and $G_{k,i}$ is the channel quality gain of user k on sub-channel i (for more details on the above computations refer to [1],[4]).

The objective of these experiments is to gauge the strength of the LP formulation and the potential of the facet inequalities (5) in reducing the integrality gap. The performance measures that have been used are:

- (i) percentage gap of bound of LP relaxation (**UB1**), expressed as

$$\text{UB1} = \frac{\text{LP relaxation objective value} - \text{IP optimal objective value}}{\text{IP optimal objective value}} \times 100;$$

- (ii) percentage gap of bound of LP relaxation with inequalities (5) (**UB2**), expressed similarly as above;
- (iii) percentage gap reduction - **GR**=(UB1-UB2)/UB1×100;
- (iv) node count of the branch-and-bound (B&B) tree for problems without and with inequalities (5); denoted by **NC1** and **NC2**, respectively;
- (v) B&B tree node count reduction percentage, **NCR**=(NC1-NC2)/NC1×100.

All computational experiments were conducted on a Intel Xeon processor with 3.4 GHz clock speed running Windows 7. Callable library of CPLEX 12.1 was used for solving LPs and IPs. All CPLEX generated cuts were suppressed while running B&B. Inequalities (5) were added only at the root node. When no more inequality can be added, the problem was submitted to B&B. The experimental results are presented in the table below. Each row represents the average of the performance measures on 100 random problem instances.

CPU time indicates the total time taken to solve the LP relaxation (without the facet inequalities) and the IP.

$ K $	$ S $	UB1	UB2	GR	No. of ineq.	NC1	NC2	NCR	CPU Time
		%	%	%	(5) added			%	(seconds)
30	50	0.368	0.093	74.728	449.570	10.910	3.550	67.461	1.150
35	50	0.296	0.059	80.068	469.340	7.760	1.840	76.289	1.630
40	50	0.286	0.048	83.217	383.420	7.580	1.630	78.496	1.940
45	50	0.343	0.076	77.843	558.080	10.910	3.920	64.070	1.960
50	50	0.369	0.090	75.610	478.200	8.970	3.770	57.971	2.190
60	40	0.332	0.085	74.398	268.840	5.870	2.070	64.736	1.280
65	40	0.354	0.093	73.729	274.340	6.100	2.340	61.639	1.480
70	40	0.218	0.050	77.064	238.570	2.140	0.730	65.888	1.430
75	40	0.314	0.079	74.841	222.640	9.540	2.180	77.149	1.550
80	40	0.255	0.080	68.627	283.390	2.930	1.840	37.201	1.870

The results show that the initial linear relaxation of (*IA*) is already very tight. The linear relaxation is substantially improved by adding inequalities (5). The cutting plane approach allowed us to solve real-life size problems. More experiments with other kinds of random pay-off functions are needed to evaluate the robustness and the limits of the cutting plane approach. It will be worthwhile to explore approximation schemes with tight approximation ratios.

References

- [1] 3rd Generation Partnership Project, Technical Specification Group Radio Access Network, Physical layer aspects for evolved Universal Terrestrial Radio Access (UTRA), 3GPP Std. TR 25.814 v. 7.0.0, 2006.
- [2] Golumbic, M. C., “Algorithmic Graph Theory and Perfect Graphs,” 1st Ed., Academic Press, New York, 1980.
- [3] Lee, S., I. Pefkianakis, A. Meyerson, S. Xu, and S. Lu, *Proportional fair frequency-domain packet scheduling for 3GPP LTE uplink*, Proc. IEEE INFOCOM, Rio de Janeiro, 2009.
- [4] Lim, J., H. G. Myung, K. Oh, and D. J. Goodman, *Proportional fair scheduling of uplink single-carrier FDMA systems*, Proc. IEEE PIMRC, Helsinki, 2006.
- [5] Lovász, L., and M. D. Plummer, “Matching Theory,” North-Holland, Amsterdam, 1986.
- [6] Rawi, M. A., R. Jantti, J. Torsner, and M. Sagfors, *Opportunistic uplink scheduling for 3G LTE systems*, Proc. IEEE Innov. Info. Tech., 2007.
- [7] Wong, I. C., O. Oteri, and W. McCoy, *Optimal resource allocation in uplink SC-FDMA systems*, IEEE Trans. Wireless Comm. **8** (2009), 2161–2165.

Stochastic Survivable Network Design Problems

Ivana Ljubić^{a,1,2}, Petra Mutzel^{b,2}, and Bernd Zey^{b,2}

^a *Department of Statistics and Operations Research, University of Vienna,
Vienna, Austria*

^b *Department of Computer Science, TU Dortmund, Dortmund, Germany*

Abstract

In this paper we introduce survivable network design problems under a two-stage stochastic model with fixed recourse and finitely many scenarios. We propose a new cut-based formulation based on orientation properties which is stronger than the undirected cut-based model. We use a two-stage branch&cut algorithm for solving the decomposed model to provable optimality. In order to accelerate the computations, we suggest a new cut strengthening technique for the decomposed L-shaped optimality cuts that is computationally fast and easy to implement.

Keywords: stochastic survivable network design, stochastic integer programming, branch-and-cut, Benders decomposition

1 Introduction

Motivation. We consider the edge-connectivity version of the survivable network design problem (SNDP) which is one of the most fundamental problems

¹ I. Ljubić is supported by the APART Fellowship of the Austrian Academy of Sciences (OEAW). This work was partially done during I. Ljubić's research stay at the TU Dortmund.

² Emails: ivana.ljubic@univie.ac.at, {petra.mutzel,bernd.zey}@tu-dortmund.de

in the design of telecommunication networks. Many of the classical network design problems like, e.g., the shortest path problem, the spanning tree problem, the Steiner tree problem, etc., are all special cases of the SNDP.

The (*deterministic*) SNDP is defined as follows: We are given a simple undirected graph $G = (V, E)$ with edge costs $c_e \geq 0$, $\forall e \in E$, and a symmetric $|V| \times |V|$ *connectivity requirement* matrix $\mathbf{r} = [r_{ij}]$. Thereby, $r_{ij} \in \mathbf{N} \cup \{0\}$ represents the minimal required number of *edge-disjoint paths* between two distinct vertices $i, j \in V$. The goal is to find a subset of edges $E' \subseteq E$ that satisfies the connectivity requirements and minimizes the overall solution cost being defined as $\sum_{e \in E'} c_e$. We assume that the connectivity requirements imply that each feasible solution comprises a single connected component, in which case the problem is called the *unitary* SNDP.

In practice, however, network planners often have to deal with uncertain data, e.g., the importance of a node and the associated connectivity requirements are not known in advance, or the costs of establishing links (installing new pipes, cables, etc.) may be subject to uncertainty. One promising approach to deal with this type of uncertainty is stochastic programming. Thereby, the uncertain data is modeled using random variables and a set of scenarios defines their possible outcomes.

In the two-stage stochastic network design problem the network planner wants to establish profitable connections now (*first stage*, on Monday) while taking all possible uncertain outcomes (*scenarios*) into account. In the future (*second stage*, on Tuesday) the actual scenario is revealed and additional edges can be purchased (*recourse action*) to satisfy the now known requirements. The objective is to optimize the *expected cost* of the solution, i.e., the sum of the first stage cost plus the expected cost of the second stage. Thereby, all connectivity requirements for all scenarios have to be satisfied.

Previous work. There exists a large body of work on various variants of the deterministic SNDP. We refer to [4] for a comprehensive literature overview. Regarding the stochastic variants of the SNDP, there are significantly less results published so far. The two-stage stochastic Steiner tree problem is a special case in which connectivity requirements are zero or one. For this problem, both approximation algorithms (see, e.g., [3]) and MIP approaches (see [1]) were developed. For the more general case in which connectivity requirements are arbitrary natural numbers, up to our knowledge, there only exists an $O(1)$ approximation algorithm (see [2]) for a special case of the SSNDP.

2 Problem Definition and ILP models

Problem Definition. Let $G = (V, E)$ be an undirected network with known first-stage edge costs $c_e^0 \geq 0$, for all $e \in E$. Connectivity requirements as well as the costs of edges to be purchased in the second stage are known only in the second stage. These values together form a random variable ξ , for which we assume that it has a finite support. It can therefore be modeled using a finite set of scenarios $\mathcal{K} = \{1, \dots, K\}$, $K \geq 1$. The realization probability of each scenario is given by $p^k > 0$, $k \in \mathcal{K}$; we have $\sum_{k \in \mathcal{K}} p^k = 1$. Denote by $c_e^k \geq 0$ the cost of an edge $e \in E$ if it is bought in the second stage under scenario $k \in \mathcal{K}$. W.l.o.g. we assume that $\sum_{k \in \mathcal{K}} p^k c_e^k \geq c_e^0$, for all $e \in E$. Furthermore, let \mathbf{r}^k be the matrix of *unitary* connectivity requirements under the k -th scenario.

The *two-stage stochastic survivable network design problem* (SSNDP) can then be formulated as follows: Determine the subset of edges $E^0 \subseteq E$ to be purchased in the first stage, and the sets E^k of additional edges to be purchased in each scenario $k \in \mathcal{K}$, such that the overall cost defined as $\sum_{e \in E^0} c_e^0 + \sum_{k \in \mathcal{K}} p^k \sum_{e \in E^k} c_e^k$ is minimized, while $E^0 \cup E^k$ satisfies all connectivity requirements between each pair of nodes defined by \mathbf{r}^k , for all $k \in \mathcal{K}$. Observe that each feasible solution of the deterministic SNDP is a connected subgraph of G , whereas the optimal first-stage solution of the SSNDP is not necessarily connected. In fact, the optimal solution may contain several disjoint fragments depending on the connectivity requirements throughout different scenarios or depending on the second-stage cost structure.

Semi-directed model. Formulating the SSNDP with an undirected cut-based formulation is straight-forward, cf. [6]. However, MIP models on bidirected graphs are known to provide better LP-based lower bounds for the deterministic SNDP, in particular when feasible SNDP solutions are allowed to contain two or more edge-biconnected components. Therefore we are looking for a possibility to strengthen the undirected model by bi-directing the given graph G and replacing edge- by arc-variables in the same model. The main difficulty with the SSNDP is that the edges of the first-stage solution cannot be oriented, even though the deterministic counterpart allows an orientation. We will use a semi-directed model instead to overcome these difficulties and provide a MIP model that is strictly stronger than the undirected one.

To provide a semi-directed MIP model, we will exploit the ideas of Magnanti and Raghavan [8] used for the deterministic SNDP. If connectivity requirements are $\{0, 1, \text{even}\}$, the underlying orientation approach uses the fact that any optimal solution consists of edge-biconnected components connected

with each other by bridges. Each of those edge-biconnected components can be oriented in such a way that for each pair of distinct nodes i and j from the same component, there exist $r_{ij}/2$ directed paths between i and j , and $r_{ij}/2$ directed paths between j and i . In the orientation procedure a node v_r is chosen for which we know that it is a part of an edge-biconnected component, and each bridge is oriented away from this component. In this approach we basically orient edge-biconnected components, shrink them into single nodes and orient the obtained tree away from the “root” v_r . The corresponding MIP model, which is stronger than the undirected one, uses binary arc variables that are associated to this orientation.

To model the general SNDP—i.e., the SNDP with arbitrary connectivity requirements $r_{ij} \in \mathbb{N} \cup \{0\}$ —Magnanti and Raghavan [8] present an *extended MIP formulation*, using these orientation properties. Thereby, they use the same model from above, with the only difference that the binary arc variables are relaxed to be continuous. This change makes the model valid for arbitrary values of r_{ij} and is provably stronger than its undirected counterpart.

Unfortunately, for the SSNDP, the first stage solution E^0 is not necessarily connected. Therefore, it is impossible to use this orientation idea for the edges from E^0 , since each arc is not used in exactly the same direction over all scenarios. Hence, the first stage decision variables remain associated with undirected edges. However, we can provide a directed formulation once the solution gets completed in the second stage, i.e., we can “orient” the edges of $E^0 \cup E^k$ —independently for each scenario. We set the root v_r^k for each scenario $k \in \mathcal{K}$ to be one of the nodes with the highest connectivity requirement, and search for individual orientations of each of the K scenario solutions.

Let $\mathcal{W}_1^k := \{W \mid W \subset V, \max_{i \in W, j \notin W} r_{ij}^k = 1, v_r^k \notin W\}$ and $\hat{\mathcal{W}}^k := \{W \mid W \subset V, \max_{i \in W, j \notin W} r_{ij}^k \geq 2\}$ be the set of *critical cutsets* and *regular cutsets*, respectively, with the associated values of $f^k(W)$ defined as $f^k(W) := 1$, if $W \in \mathcal{W}_1^k$ and $f^k(W) := \max_{i \in W, j \notin W} r_{ij}^k/2$, if $W \in \hat{\mathcal{W}}^k$.

The following model orients the second stage solution, given the installation of (undirected) edges from the first stage. We use binary variables (x^0, y^1, \dots, y^K) to model the solution edges: for each edge e variable x_e^0 for the first stage and y_e^k for the second stage (scenario k). The latter variables include the edges that are already bought in the first stage, i.e., we have $y_e^k = 1$ if $e \in E^0 \cup E^k$, and $y_e^k = 0$, otherwise. To “orient” the edges from $E_0 \cup E_k$ continuous variables z_{ij}^k are used. The model will be called *SD*:

$$(SD) \quad \min \sum_{e \in E} c_e^0 x_e^0 + \sum_{k \in \mathcal{K}} p^k \sum_{e \in E} c_e^k (y_e^k - x_e^0)$$

$$\begin{aligned}
z^k(\delta^-(W)) \geq f^k(W), & \quad \forall W \in \mathcal{W}_1^k \cup \hat{\mathcal{W}}^k, \forall k \in \mathcal{K} & (1) \\
z_{ij}^k + z_{ji}^k \geq x_e^0, & \quad \forall e = \{i, j\} \in E, \forall k \in \mathcal{K} & (2) \\
z_{ij}^k + z_{ji}^k \leq y_e^k, & \quad \forall e = \{i, j\} \in E, \forall k \in \mathcal{K} & (3) \\
z_{ij}^k \geq 0, & \quad \forall (i, j) \in A, \forall k \in \mathcal{K} & (4)
\end{aligned}$$

$(x^0, y^1, \dots, y^K)^T \in \{0, 1\}^{(K+1)|E|}$

Constraints (1) are separated by *minimum cut* computations and model the “orientation” of the solution, independently for each of the scenarios. Variables z_{ij}^k are fractional, and therefore, the model is valid for any $r_{ij}^k \in \mathbb{N} \cup \{0\}$. Finally, constraints (2) and (3) ensure that variables z_{ij}^k can be used only along the edges that are either purchased in the first stage, or added in the second stage.

Lemma 2.1 *Formulation (SD) models the deterministic equivalent of the two-stage stochastic survivable network design problem correctly and is stronger than the undirected formulation.*

3 Decomposition

Notice that, even for a constant number of scenarios, our model contains an exponential number of constraints that can be typically solved in one branch&cut (b&c) approach. The main drawback of such a b&c approach is that we still have to deal with a large set of variables. Alternatively, in [1] we have proposed to combine the cutting plane algorithm with a Benders-like decomposition approach in which the variables of the first stage are kept in the master problem, and the second stage variables are projected out and replaced by a single variable per scenario (Θ^k).

For a fixed first stage decision \tilde{x}^0 , the problem decomposes into K subproblems, each of which can be independently solved using a b&c approach. Dual variables of the LP-relaxations of these subproblems impose *L*-shaped cuts that are added to the master while the exact solutions of the subproblems impose integer *L*-shaped cuts [5]. This new decomposition approach that combines the b&c in the master problem with the b&c in the subproblems is called *two-stage b&c algorithm*. Computational results of [1], applied to the stochastic Steiner tree problem, have shown that two-stage b&c significantly outperforms the b&c applied directly to the deterministic equivalent of the extended formulation.

Hence, in this paper we propose to solve the SSNDP using the two-stage

b&c from [1] applied to the semi-directed model (*SD*).

For each fixed—and possibly fractional—first-stage solution \tilde{x}^0 , the second-stage problem decomposes into K independent subproblems, which we will refer to as *restricted deterministic SNDP*'s.

By removing the integrality constraints and using dual variables α_W , β_e , γ_e , η_{ij} and τ_e associated to constraints (1), (2), (3), (4), and $y_e^k \leq 1$, respectively, we obtain the following dual problem, for each fixed $k \in \mathcal{K}$ and the first stage solution \tilde{x}^0 :

$$(D:SD) \max \sum_{W \in \mathcal{W}_1^k \cup \hat{\mathcal{W}}^k} f^k(W) \alpha_W + \sum_{e \in E} (\tilde{x}_e^0 \beta_e - c_e^k \tilde{x}_e^0 - \tau_e) \quad \begin{aligned} \gamma_e - \tau_e &\leq c_e^k, \quad \forall e \in E \\ \sum_{W \in \mathcal{W}_1^k \cup \hat{\mathcal{W}}^k : (i,j) \in \delta^-(W)} \alpha_W + \beta_e - \gamma_e + \eta_{ij} &\leq 0, \quad \forall (i,j) \in A \end{aligned} \quad (5)$$

$$(\alpha, \beta, \gamma, \eta, \tau) \geq 0 \quad (7)$$

Let $\tilde{\alpha}_W, \tilde{\beta}_e, \tilde{\gamma}_e, \tilde{\eta}_{ij}, \tilde{\tau}_e$ be an optimal solution to (D:SD). A (decomposed) *L-shaped optimality cut* is then defined as follows:

$$\Theta^k + \sum_{e \in E} x_e^0 (c_e^k - \tilde{\beta}_e) \geq \sum_{W: W \in \mathcal{W}_1^k \cup \hat{\mathcal{W}}^k} f^k(W) \tilde{\alpha}_W - \sum_{e \in E} \tilde{\tau}_e \quad (8)$$

4 Strengthening *L-shaped cuts*

Notice that the number of master iterations of the decomposition approach—and hence, the overall running time—is highly influenced by the strength of the generated *L-shaped cuts*. In this paper we propose a new and fast way of generating strengthened *L-shaped cuts*. In contrast to the previously proposed strengthening approach that searches for Pareto optimal L-shaped cuts (cf. [7]) we do not require solving an auxiliary LP in order to generate a stronger cut, but rather, we are able to find it in linear time.

The L-shaped cuts for the formulation (*SD*) can be strengthened as follows: If for an edge $e \in E$ the first stage solution \tilde{x}^0 is such that $\tilde{x}_e^0 = 0$, then the corresponding β_e variable does not appear in the objective function of the dual (*SD*). Furthermore, variables γ_e and η_{ij} do not appear in the objective function neither, and therefore, LP-optimal solutions frequently have a positive slack in constraints (6). Our idea is to reduce this slack to zero, and to thereby increase the value of β_e as follows: Let $(\tilde{\alpha}, \tilde{\beta}, \tilde{\gamma}, \tilde{\eta}, \tilde{\tau})$ be an optimal solution to

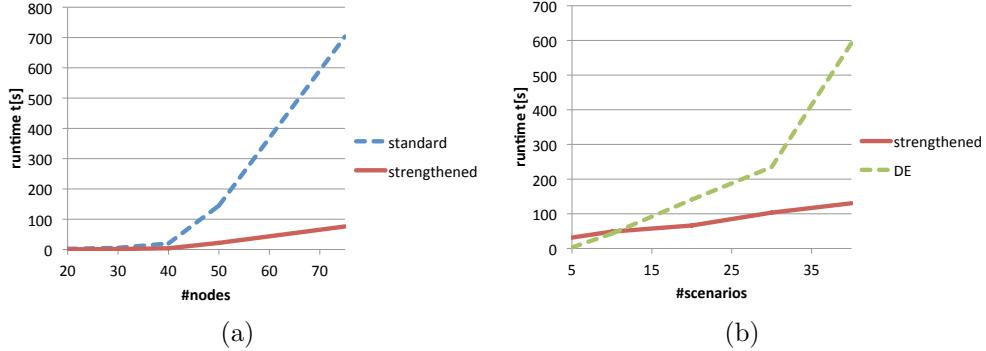


Fig. 1. Comparison of the runtime between (a) two-stage b&c with standard and strengthened L-shaped cuts, respectively, and (b) between two-stage b&c with strengthened cuts and the b&c for the extended formulation, respectively.

$(D:SD)$ as before. Let $\hat{\beta}_e := \tilde{\gamma}_e - \max_{a \in \{(i,j),(j,i)\}} \{ \sum_{W:a \in \delta^-(W)} \tilde{\alpha}_W + \tilde{\eta}_a \}, \forall e = \{i,j\} \in E$ s.t. $\tilde{x}_e^0 = 0$.

If $\hat{\beta}_e > \tilde{\beta}_e$ holds for at least one $e \in E$ the *strengthened L-shaped cut* is given as:

$$\Theta^k + \sum_{e \in E} x_e^0 (c_e^k - \hat{\beta}_e) \geq \sum_{W:W \in \mathcal{W}_1^k \cup \hat{\mathcal{W}}^k} f^k(W) \tilde{\alpha}_W - \sum_{e \in E} \tilde{\tau}_e. \quad (9)$$

Lemma 4.1 *The strengthened L-shaped (9) cuts are valid and strictly stronger than the standard L-shaped cuts (8).*

5 Computational Results

We implemented the decomposition of model (SD) using Abacus 3.0 as a generic b&c framework. We use IBM CPLEX (version 12.1) via the interface COIN-Osi 0.102 as LP solver. All experiments were performed on an Intel Core-i7 2.67 GHz Quad Core machine with 12 GB RAM under Ubuntu 11.4. Each run was performed on a single core. Stochastic instances were generated by adopting frequently used graph generators from the network design community; for details see ls11-www.cs.uni-dortmund.de/staff/zey/ssndp.

To analyze the benefit of using strengthened L-shaped cuts we compare the computation time as well as the number of master iterations. Fig. 1 (a) depicts the computation time of the two-stage b&c algorithm with standard and strengthened L-shaped cuts, respectively. Each data point is the average over 5 runs and 5 instances per k scenarios, $k \in \{5, 10, 20, 30, 40\}$ for each

number of nodes $n \in \{20, 30, 40, 50, 75\}$.

The gained speedup in running time by using the strengthened L-shaped cuts is significant: It is about 4 (for $n \in \{20, 30, 40\}$), 6 ($n = 50$), and over 9 ($n = 75$) times, respectively. For example, instances with 75 vertices are solved in approx. 12 minutes without and in 1.5 minutes with the strengthening (on average). The average number of master iterations also decreases rapidly, e.g., from 288 to 58, for $n = 75$.

The benefit of using the decomposition over the extended formulation (EF) can be seen in Fig. 1 (b) which illustrates the increase of the running time with the increase of the number of scenarios. For small instances ($k \leq 20$ or $n \leq 40$) the EF has a faster computation time. But with an increasing number of scenarios or graph size the decomposition clearly outperforms the EF which, e.g., was not able to solve all of the instances with $n \geq 75$ in two hours.

References

- [1] Bomze, I., M. Chimani, M. Jünger, I. Ljubić, P. Mutzel and B. Zey, *Solving two-stage stochastic Steiner tree problems by two-stage branch-and-cut*, in: *ISAAC* (1), 2010, pp. 427–439.
- [2] Gupta, A., R. Krishnaswamy and R. Ravi, *Online and stochastic survivable network design*, in: *ACM-SIAM STOC* (2009), pp. 685–694.
- [3] Gupta, A., R. Ravi and A. Sinha, *LP rounding approximation algorithms for stochastic network design*, *Math. of Operations Research* **32** (2007), pp. 345–364.
- [4] Kandyba, M., “Exact Algorithms for Network Design Problems using Graph Orientations,” Ph.D. thesis, Technische Universität Dortmund (2011).
- [5] Laporte, G. and F. Louveaux, *The integer L-shaped method for stochastic integer programs with complete recourse*, *Oper. Res. Lett.* **13** (1993), pp. 133–142.
- [6] Ljubić, I., P. Mutzel and B. Zey, *Stochastic survivable network design problems*, Technical Report TR12-1-003, TU Dortmund (2012).
- [7] Magnanti, T. and R. Wong, *Accelerating Benders’ decomposition: Algorithmic enhancement and model selection criteria*, *Operations Research* **29** (1981), pp. 464–484.
- [8] Magnanti, T. L. and S. Raghavan, *Strong formulations for network design problems with connectivity requirements*, *Networks* **45** (2005), pp. 61–79.

Optimizing Toll Enforcement in Transportation Networks: a Game-Theoretic Approach

Ralf Borndörfer, Julia Buwaya, Guillaume Sagnol, Elmar Swarat¹
Zuse Institut Berlin (ZIB), Department Optimization, Berlin, Germany²

Abstract

We present a game-theoretic approach to optimize the strategies of toll enforcement on a motorway network. In contrast to previous approaches, we consider a network with an arbitrary topology, and we handle the fact that users may choose their Origin-Destination path; in particular they may take a detour to avoid sections with a high control rate. We show that a Nash equilibrium can be computed with an LP (although the game is not zero-sum), and we give a MIP for the computation of a Stackelberg equilibrium. Experimental results based on an application to the enforcement of a truck toll on German motorways are presented.

Keywords: Game Theory; Stackelberg Equilibrium; Mixed Integer Programming

1 Introduction

In 2005 Germany introduced a distance-based toll for trucks weighing twelve tonnes or more in order to fund growing investments for maintenance and extensions of motorways. The enforcement of the toll is the responsibility of the German Federal Office for Goods Transport (BAG), who has the task to carry out a network-wide control. To this end, 300 vehicles make control tours on the entire highway network. In this paper, we present some theoretical work obtained in the framework of our cooperation with the BAG, whose final goal is to develop an optimization tool to schedule the control tours of the inspectors. This real-world problem is subject to a variety of legal constraints, which we handle by mixed integer programming [2]. In a follow-up work, we plan to use the results of the present article as a target for the real-world problem.

¹ This work was funded by the German general office for Good Transport (BAG)

² {borndoerfer,buwaya,sagnol,swarat}@zib.de

In this paper, the problem of allocating inspectors to spatial locations of a transportation network in order to enforce the payment of a transit toll is studied from a game-theoretic point of view. This problem presents several similarities with recent studies on the application of game theory to a class of problems where the goal is to randomize different kind of inspections, in a strategical way; this includes a work on the optimal selection of checkpoints and patrol routes to protect the LA Airport towards adversaries [5], a study of the scheduling and allocation of air marshals to a list of flights in the US [3], or the problem of optimally scheduling fare inspection patrols in LA Metro [6].

The core of this work is to handle the difficulties arising from the large number of available paths for the network users, while taking into account the additional traveling costs when users make a detour. In contrast, previous approaches used the trivial topology of a single metro line [6], or assumed that each user takes the shortest path [1]. In a network security game [3], the exponential number of actions of the defender was handled with the help of a branch-and-price algorithm. In this article, we represent user strategies by network flows, which allows us to give a compact LP formulation for the computation of a Nash equilibrium of the game. We next use some ideas of [5] to find a Stackelberg equilibrium of the game by mixed integer programming (MIP). Experimental results based on real traffic data (averaged over time) are given in section 4, and suggest that the Nash equilibrium strategy is a good trade-off between computation time and efficiency of the controls.

2 A Spot-checking game

In this section we extend the game theoretic model presented in [1], which studies the interaction between the fare inspectors and the users of a transportation network, to handle the case where every user is free to choose its path in the network to reach its destination.

We first recall the notion of *best response* in game theory: In a game with N players where each player may choose a strategy \mathbf{p}_i in a set Δ_i , and wishes to maximize his own payoff $u_i(\mathbf{p}_i, \mathbf{p}_{-i}) \equiv u_i(\mathbf{p}_1, \dots, \mathbf{p}_N)$, we say that \mathbf{p}_i is a best response to the set \mathbf{p}_{-i} of strategies of the other players if

$$\forall \mathbf{p}'_i \in \Delta_i, \quad u_i(\mathbf{p}'_i, \mathbf{p}_{-i}) \leq u_i(\mathbf{p}_i, \mathbf{p}_{-i}).$$

Model settings The transportation network is represented by a weighted directed graph $G(V, E, \mathbf{w})$, where the weight w_e represents the traveling cost on edge $e \in E$. We assume that the users of the network are distributed over a set of commodities $\mathcal{K} = \{k_1, \dots, k_m\}$, which represent Origin-Destination pairs of the network $k = (\text{src}(k), \text{dst}(k))$. We denote by S the set of commodity sources $\{s : \exists(s, t) \in \mathcal{K}\}$ and for $s \in S$ we define $D_s := \{t : (s, t) \in \mathcal{K}\}$.

For all $k \in \mathcal{K}$, we denote by \mathcal{R}_k the set of all paths from $\text{src}(k)$ to $\text{dst}(k)$. In particular, $R_k^* \in \mathcal{R}_k$ is the shortest path through k (with respect to weights w_e). In addition, we are given the demand x_k of commodity k , i.e., the number of users who make a trip on commodity k during a given period of time.

The users of commodity k are expected to pay a toll fee T_k . If a user evades the toll, he takes the risk to pay a penalty $P >> T_k$. If an inspector is present on edge e , we denote by σ_e the probability that an individual passing on e is controlled (σ_e depends e.g. on the ratio *inspectors speed* over *trucks speed*).

Inspectors' strategy The set of edges is partitioned as $E = E_{\text{pay}} \cup E_{\text{free}}$, where E_{free} represent some *toll-free* edges, where users shall not be controlled. There are γ teams of inspectors over the network, who can each control an edge $e \in E_{\text{pay}}$. Their pure strategies hence correspond to the subsets of E_{pay} of cardinality γ . A mixed strategy is a probability distribution over those subsets, but we will see that our model only depends on the marginal probabilities q_e that some inspector is present on e , that must satisfy

$$\sum_{e \in E_{\text{pay}}} q_e = \gamma \quad \text{and} \quad \forall e \in E_{\text{pay}}, \quad 0 \leq q_e \leq 1. \quad (1)$$

Conversely, for every vector \mathbf{q} satisfying (1), we point out that we can find a probability distribution over the subsets of cardinality γ whose marginal equals \mathbf{q} . We assume hereafter that $q_e = 0$ is a constant for every $e \in E_{\text{free}}$.

Network users, fare evasion and path selection We associate the users of commodity k with a single player (called player k). Player k can either pay the toll and take the shortest path, or try to evade the toll by taking any path $R \in \mathcal{R}_k$ (which might be a detour). His mixed strategy can be interpreted as the proportion of k -users who pay or evade on a particular path $R \in \mathcal{R}_k$. For the sake of simplicity we create an artificial edge e_k^* with weight $w_{e_k^*} := \sum_{e \in R_k^*} w_e + T_k$ which goes directly from the origin to the destination of k , and we define $\bar{E}_k := E \cup \{e_k^*\}$; the interpretation is that player k pays the toll if he takes e_k^* . Our model depends only on the probabilities p_e^k that player k uses edge e , that must form a flow of value one through commodity k :

$$\forall v \in V, \quad \sum_{\{u:(v,u) \in \bar{E}_k\}} p_{(v,u)}^k - \sum_{\{u:(u,v) \in \bar{E}_k\}} p_{(u,v)}^k = \begin{cases} 1 & \text{if } v = \text{src}(k); \\ -1 & \text{if } v = \text{dst}(k); \\ 0 & \text{otherwise.} \end{cases} \quad (2)$$

The probability to be controlled on an edge $e \in R$ is $q_e \sigma_e$, and hence the expected number of times player k is subjected to a control during his trip is $\sum_{e \in R} p_e^k q_e \sigma_e$. We approximate the total expected cost of player k by

$$\text{Payoff}_k(\mathbf{p}, \mathbf{q}) = - \left(\sum_{e \in \bar{E}_k} p_e^k w_e + \sum_{e \in E} p_e^k q_e \sigma_e P \right), \quad (3)$$

where the first term accounts for travel and toll costs, while the second is the expected fine, i.e. we do as if evaders could be fined several times (for a realistic number of controllers, our results show that the risk of being controlled more than once is very small; a similar approximation has been used in [6] and [1]). Note that the payoff of any user does not depend on the strategy of other users. Hence the game is equivalent to a Bayesian game where the controller plays against a random user $k \in \mathcal{K}$, with probabilities proportional to x_k [5].

A consequence of Equation (3) is that the best response of player k to the inspectors' strategy \mathbf{q} is to take the shortest path for commodity k in the graph $G(V, \bar{E}_k, \mathbf{w}')$ with modified edge weights $w'_e = w_e + q_e \sigma_e P$, where $q_{e_k^*}$ is a constant set to 0 for the artificial edge e_k^* .

Inspectors' payoff We introduce two parameters α and β , where $\alpha \in [0, 1]$ indicates the fraction of the revenue from penalties to take into account, and β_e is a reward for each user who takes edge e . Hence, the total payoff for the controllers is:

$$\text{Payoff}_C(\mathbf{p}, \mathbf{q}) = \sum_k x_k \sum_{e \in \bar{E}_k} p_e^k (\alpha \sigma_e q_e P + \beta_e). \quad (4)$$

If we set $\alpha = 1$ (resp. $\alpha = 0$) and $\beta_e = 0$ for all edges except the artificial ones, where $\beta_{e_k^*} = T_k$, the payoff defined in (4) corresponds to the total revenues from toll and penalties (resp. the toll revenues only). We denote this setting as MAXPROFIT (resp. MAXTOLL). Another interesting case, called MAXPAYERS, is $\alpha = 0$ and $\beta_{e_k^*} = 1$, where the goal is to maximize the number of users who have an incentive to pay the toll.

3 Computation of Equilibria

The notion of equilibrium is essential in game theory. Depending on the ability of the players to observe the others' actions, committing to a Nash or a Stackelberg equilibrium may be better suited [4]. We first show that in the case of MAXPROFIT, our game can be transformed into a *zero-sum game* which has the same Nash equilibria. As a consequence, a Nash equilibrium strategy can be computed by linear programming.

Computation of a Nash equilibrium for MAXPROFIT A Nash equilibrium is a collection of mixed strategies (\mathbf{p}, \mathbf{q}) such that every player plays with best response to the others' strategies. As seen in the last section, this means that $\lambda_k := -\text{Payoff}_k(\mathbf{p}, \mathbf{q})$ equals the length of the shortest path for commodity k in the graph $G(V, \bar{E}_k, \mathbf{w}')$, where $w'_e = w_e + q_e \sigma_e P$. Now, for a fixed strategy \mathbf{p} of the network users, the goal of the controller is to maximize his total revenue $\sum_k x_k (\sum_{e \in E} p_e^k \sigma_e q_e P + p_{e_k^*}^k T_k)$ with respect to \mathbf{q} . Equivalently, the controller's goal is to maximize

$$\sum_k x_k \left(\sum_{e \in E} p_e^k \sigma_e q_e P + p_{e_k^*}^k T_k \right) + \sum_k x_k \left(\sum_{e \in E} p_e^k w_e + p_{e_k^*}^k (w_{e_k^*} - T_k) \right) = \sum_k x_k \lambda_k,$$

because the term which was added does not depend on \mathbf{q} . We can now formulate a linear program (LP) which computes a Nash equilibrium strategy:

$$\max_{\mathbf{q}, \boldsymbol{\lambda}, \mathbf{y}} \quad \sum_k x_k \lambda_k \tag{5a}$$

$$\text{s. t.} \quad y_v^s - y_u^s \leq w_{(u,v)} + \sigma_{(u,v)} q_{(u,v)} P, \quad \forall s \in S, \forall (u,v) \in E; \tag{5b}$$

$$y_s^s = 0, \quad \forall s \in S; \tag{5c}$$

$$\lambda_k \leq y_{\text{dst}(k)}^{\text{src}(k)} \quad \forall k \in \mathcal{K} \tag{5d}$$

$$\lambda_k \leq w_{e_k^*}, \quad \forall k \in \mathcal{K}; \tag{5e}$$

$$0 \leq q_e \leq 1, \quad \forall e \in E; \tag{5f}$$

$$\sum_{e \in E} q_e = \gamma. \tag{5g}$$

The constraints (5b)-(5c) are from the classical linear programming formulation of the single-source shortest path problem, and bound the potential y_v^s from above by the shortest path length from s to v in the graph $G(V, E, \mathbf{w}')$. Constraints (5d) and (5e) further bound λ_k from above by the length of the shortest path for commodity k in the augmented graph $G(V, \bar{E}_k, \mathbf{w}')$. Finally the constraints (5f)-(5g) force \mathbf{q} to be a feasible strategy for γ inspectors.

We point out that the optimal dual variables of constraints (5b) and (5e) define a flow in the graph $G(V, \cup_k \bar{E}_k)$, from which a Nash equilibrium strategy \mathbf{p}^k for player k can be inferred.

Computation of a Stackelberg equilibrium In a Stackelberg game, it is assumed that a player is the leader (in our case, the controller), who plays first, and the other players (called followers) react with a best response to the leader's action. Stackelberg games are arguably more adapted to the present spot-checking game because of the asymmetry between controllers and network users, and have already been used in several applications [5,3,6]. A Stackelberg equilibrium is a profile of strategies (\mathbf{p}, \mathbf{q}) which maximizes the leader's payoff, among the set of all the profiles where the followers' strategies \mathbf{p} are best responses to the leader's action \mathbf{q} . Note that the definition implicitly implies that when a follower has several best response actions available, he will select one that favors the leader most. This can be justified in our spot-checking game, since strategies that favor the controller correspond to shorter paths (more penalties, less travel charges).

Using ideas similar as in [5], a mixed integer program (MIP) can be formulated for the computation of a Stackelberg equilibrium (\mathbf{p}, \mathbf{q}) . We reduce drastically the number of required variables, by using a single-source multi-sink flow \mathbf{p}^s for each $s \in S$ instead of using a flow \mathbf{p}^k for every commodity.

This however requires attention, since only the users of commodity k are allowed to take the artificial edge e_k^* :

$$\max_{\mathbf{q}, \mathbf{y}, \boldsymbol{\lambda}, \boldsymbol{\mu}, \boldsymbol{\rho}} \quad \sum_k x_k (\alpha \lambda_k + \mu^k (\beta_{e_k^*} - \alpha w_{e_k^*})) + \sum_{s \in S} \sum_{e \in E} \rho_e^s (\beta_e - \alpha w_e) \quad (6a)$$

s.t.

$$0 \leq w_{(u,v)} + \sigma_{(u,v)} q_{(u,v)} P - (y_v^s - y_u^s) \leq M_{(u,v)} (1 - \mu_{(u,v)}^s), \quad \forall s \in S, \quad \forall (u,v) \in E; \quad (6b)$$

$$y_s^s = 0, \quad \forall s \in S; \quad (6c)$$

$$0 \leq y_{\text{dst}(k)}^{\text{src}(k)} - \lambda_k \leq M_1^k \mu^k, \quad \forall k \in \mathcal{K}; \quad (6d)$$

$$0 \leq w_{e_k^*} - \lambda_k \leq M_2^k (1 - \mu^k), \quad \forall k \in \mathcal{K}; \quad (6e)$$

$$0 \leq q_e \leq 1, \quad \forall e \in E; \quad (6f)$$

$$\sum_{e \in E} q_e = \gamma; \quad (6g)$$

$$\sum_{\{u:(v,u) \in E\}} \rho_{(v,u)}^s - \sum_{\{u:(u,v) \in E\}} \rho_{(u,v)}^s = \delta_v^s(\boldsymbol{\mu}), \quad \forall s \in S, \quad \forall v \in V; \quad (6h)$$

$$0 \leq \rho_e^s \leq M^s \mu_e^s, \quad \forall s \in S, \quad \forall e \in E; \quad (6i)$$

$$\mu_e^s \in \{0, 1\}, \quad \mu^k \in \{0, 1\}, \quad \forall (s, e, k) \in S \times E \times \mathcal{K}. \quad (6j)$$

As in Problem (5), constraints (6b)-(6e) bound λ_k from above by the shortest path length for k in the graph $G(V, \bar{E}_k, \mathbf{w}')$ with modified weights, and constraints (6f)-(6g) force \mathbf{q} to be a feasible strategy for the γ inspectors. We introduce a binary variable μ_e^s which can take the value 1 only if edge e belongs to a shortest path tree rooted in s (second inequality in (6b)), and a binary variable μ^k which indicates whether player k 's best response is to pay the toll (second inequalities in (6d)-(6e), μ^k is free when $\lambda_k = y_{\text{dst}(k)}^{\text{src}(k)} = w_{e_k^*}$). The right hand side in (6h) is defined as

$$\delta_v^s(\boldsymbol{\mu}) = \begin{cases} \sum_{d \in D_s} x_{(s,d)} (1 - \mu^{(s,d)}) & \text{if } s = v; \\ -x_{(s,v)} (1 - \mu^{(s,v)}) & \text{if } v \in D_s; \\ 0 & \text{otherwise,} \end{cases}$$

so that $\boldsymbol{\rho}^s$ defines a single-source multi-sink flow rooted in s , whose demand in $d \in D_s$ corresponds to the number of evaders on the commodity (s, d) . Constraint (6i) ensures that the flow $\boldsymbol{\rho}^s$ only uses edges from a shortest path tree rooted in s . Now, $\boldsymbol{\rho}^s$ can be decomposed as $\sum_{d \in D_s} x_{(s,d)} \mathbf{p}^{(s,d)}$, where $\mathbf{p}^{(s,d)}$ is a flow from s to d of value $1 - \mu^{(s,d)}$. If (s, d) is the k^{th} commodity, i.e., $k = (s, d)$, we set $p_{e_k^*}^k := \mu^k$, and \mathbf{p}^k becomes a flow of value one from $\text{src}(k)$ to $\text{dst}(k)$ in the augmented graph $G(V, \bar{E}_k, \mathbf{w}')$. By construction, \mathbf{p}^k is a flow of minimal cost $\lambda_k = \sum_{e \in \bar{E}_k} p_e^k (w_e + q_e \sigma_e P)$, and it follows that \mathbf{p}^k is a best response to \mathbf{q} . Finally, the objective function (6a) rewrites to the controller's payoff (4) when replacing λ_k and ρ_e^s by their values as a function of p_e^k . We point out that the big- M constants M_e, M_1^k, M_2^k and M^s can all be

chosen in the same order of magnitude as the other coefficients of the problem.

Note that the problem becomes easier for MAXTOLL or MAXPAYERS, where $\alpha = 0$ and $\beta_e = 0$ for all non artificial edges $e \in E$, and the flows of network users ρ^s are not involved anymore. In this case, the second inequality of (6b) vanishes, as well as (6h) and (6i).

4 Experimental results on German motorways

We have solved the models presented in this paper for several instances based on real data from the German motorways network. We present here a brief analysis of our results.

In Figure 1(a), a near-Stackelberg equilibrium strategy of the inspectors on the whole German network is represented, for the MAXPROFIT case. Here it was assumed that $\gamma = 50$ controllers are simultaneously present on the network, which has 319 nodes, 2948 edges and 5013 commodities (dotted edges on the figure are toll-free edges $e \in E_{\text{free}}$). We first computed a Nash Equilibrium with the LP (5); this took 29s on a PC with 8 processors at 3.2GHz. Then, we computed the shortest path through k in $G(V, \bar{E}_k, \mathbf{w}')$ for all $k \in \mathcal{K}$, which yields a feasible solution for the MIP (6) that can be used for a *warm start*. We used CPLEX, and an optimality gap of 1.5% was reached after 350s. We point out that the Nash Equilibrium differs from the strategy MAXPROFIT on a few edges only, and captures 99.7% of the optimal profit.

Further tests on a smaller network representing the region of Berlin-Brandenburg (45 nodes, 130 edges, 596 commodities) confirm that the Nash Equilibrium strategy might be a good trade-off between the computation time and the efficiency of the controls. Figures 1(b)-1(d) compare 4 strategies in function of the number of controllers γ : the strategies MAXPROFIT and MAXTOLL, the Nash equilibrium strategy computed by LP (5), and a strategy in which control intensities are proportional to traffic volumes on each edge. Plot (b) shows the profit collected when committing to one of these strategies (in the Stackelberg model, i.e. drivers select a best response which favors the controller most). We see on Plot (c) that the Nash strategy is always near-optimal in terms of profit; we want to investigate this fact in future research. However, we point out that the MAXTOLL strategy outperforms the others in terms of toll enforcement (Plot (d)), at the price of a small loss in total profit (7% for $\gamma = 2$ and 2% for $\gamma = 4$). In another experiment, we have set $\gamma = 3$ and we have played with the parameter α , which joins MAXPROFIT ($\alpha = 1$) to MAXTOLL ($\alpha = 0$). Plot (e) shows that for $\alpha = 0.75$, one can find a solution with a profit that is almost as in MAXPROFIT, but with a higher fraction coming from the toll, and hence a less evasion.

Conclusion We have presented a LP / MIP approach to solve a spot-checking

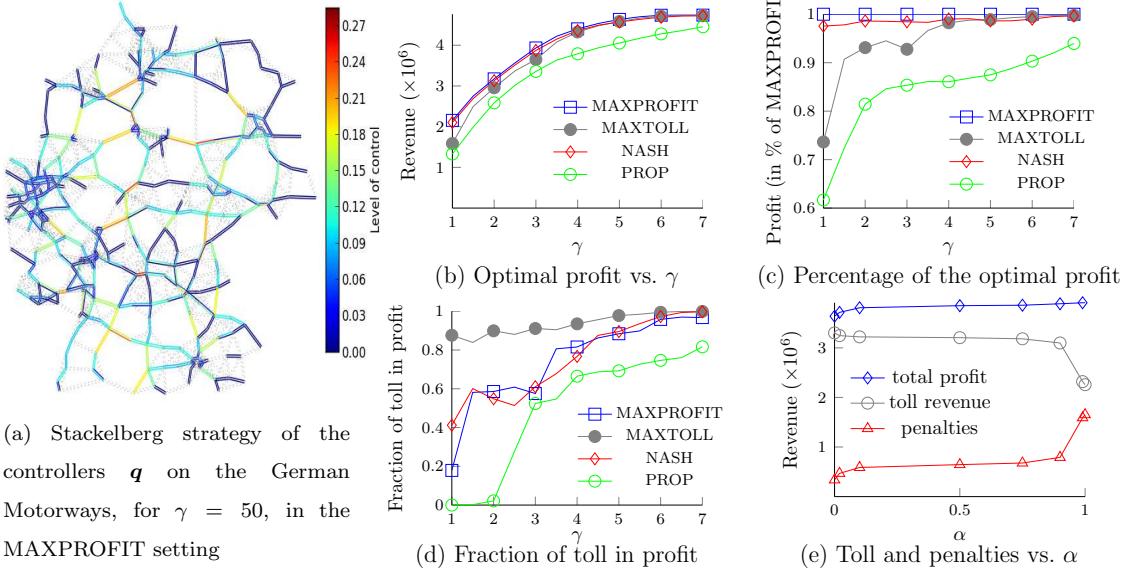


Fig. 1. Experimental Results for Germany (a) and Berlin-Brandenburg (b)-(e).

game on a transportation network. Users strategies are represented by flows in the network, which makes it possible to compute a Nash Equilibrium of the game very efficiently. Our experiments suggest that this “Nash strategy” approximates a Stackelberg Equilibrium maximizing the inspectors’ profit.

References

- [1] Borndörfer, R., B. Omont, G. Sagnol, and E. Swarat. *A Stackelberg game to optimize the distribution of controls in transportation networks*. In *Proceedings of GAMENETS 2012, Vancouver, Canada, LNICST(105)*, pages 224–235, 2012.
- [2] Borndörfer, R., G. Sagnol, and E. Swarat. *An IP Approach to Toll Enforcement Optimization on German Motorways*. In *Operations Research Proceedings 2011*, pages 317–322. Springer, 2011.
- [3] Jain, M., E. Kardes, C. Kiekintveld, F. Ordonez, and M. Tambe. *Security games with arbitrary schedules: A branch and price approach*. In *Proc. of The 24th AAAI Conference on Artificial Intelligence*, pages 792–797. AAAI Press, 2010.
- [4] Korzhik, D., Z. Yin, C. Kiekintveld, V. Conitzer, and M. Tambe. *Stackelberg vs. Nash in security games: an extended investigation of interchangeability, equivalence, and uniqueness*. *J. Artif. Int. Res.*, **41**(2):297–327, 2011.
- [5] Paruchuri, P., J.P. Pearce, J. Marecki, M. Tambe, F. Ordonez, and S. Kraus. *Playing games for security: an efficient exact algorithm for solving Bayesian Stackelberg games*. In *Proc. of the 7th AAMAS*, pages 895–902. ACM, 2008.
- [6] Yin, Z., A. Jiang, M. Johnson, M. Tambe, C. Kiekintveld, K. Leyton-Brown, T. Sandholm, and J. Sullivan. *TRUSTS: Scheduling Randomized Patrols for Fare Inspection in Transit Systems*. In *Proc. of The 24th Conference on Innovative Applications of Artificial Intelligence (IAAI)*. AAAI Press, 2012.

A Tabu Search Approach for the Prize Collecting Traveling Salesman Problem

Odivaney Pedro ^{1,3} Rodney Saldanha ⁴

*Programa de Pós-Graduação em Engenharia Elétrica
Universidade Federal de Minas Gerais
Av. Antonio Carlos 6627, 31270-901, Belo Horizonte, MG, Brazil*

Ricardo Camargo ^{2,5}

*Departamento de Engenharia de Produção
Universidade Federal de Minas Gerais
Belo Horizonte, Brazil*

Abstract

The Prize Collecting Traveling Salesman Problem is a generalization of the Traveling Salesman Problem. A salesman collects a prize for each visited city and pays a penalty for each non visited city. The objective is to minimize the sum of the travel costs and penalties, but collecting a minimum pre-established amount of prizes. This problem is here addressed by a simple, but efficient tabu search approach which had improved several upper bounds of the considered instances.

Keywords: Prize Collecting Traveling Salesman Problem, tabu search, Meta-heuristics

1 Introduction

The Prize Collecting Travelling Salesman Problem (PCTSP) is a generalization of the Traveling Salesman Problem (TSP), a well-known NP-hard combinatorial optimization problem. A traveling salesman has to visit some cities, with each one having a prize and a penalty associated with it. The prize is collected whenever a city is visited; and the penalty is applied whenever the city is not visited. Further, there is a cost for traveling between two cities. The objective is to minimize the sum of the travel costs and penalties paid, but ensuring the collection of a minimum prize. The PCTSP has several practical applications such as scheduling the daily operation of a steel rolling mill [15]; the helicopter tour planning for offshore oil platforms; the design of tourist routes [1,2].

Several authors proposed different approaches to tackle the PCTSP. Fischetti and Totti [10] presented an exact method to solve it relying on Lagrangian bounds and a specialized branch and bound algorithm. Bienstock et al. [3] presented an approximation algorithm with constant bounds. Dell'Amico et al. [9] reported large bounds based also on a Lagrangian sub-gradient method. Gomes et al. [13] utilized a GRASP heuristic with a variable neighborhood descent scheme to obtain upper bounds. Chaves et al. [4] extended the algorithm devised by [13] and proposed a GRASP procedure with a variable neighborhood descent combined with a variable neighborhood search to get good upper bounds.

Chaves and Lorena proposed [5] two procedures based on clustering search to solve the PCTSP. The main difference between the two procedures is on how the initial solutions are generated: While the first procedure produces solutions by means of an evolutionary algorithm; the second creates starting solutions via a GRASP method associated with VNS/VND. In addition, Chaves and Lorena [4,7] developed a similar approach by devising hybrid algorithms based on clustering search associated with GRASP and VNS/VND.

In this paper, a tabu search (TS) approach is devised to solve an asymmetric version of the PCTSP. The proposed algorithm incorporates well-known features for the TSP: The construction phase based on the GENIUS procedure [11], and the 2-opt local search [14]. Although there are several studies show-

¹ The author thanks CNPq and FAPEMIG for their financial support.

² The author was supported by CNPq grants 305446/2010-0 and 480295/2012-3, Brazil.

³ Email: nutorp@yahoo.com.br

⁴ Email: rodney@cpdee.ufmg.br

⁵ Email: rcamargo@dep.ufmg.br

ing the successful application of the TS method and the GENIUS procedure to solve other TSP and vehicle routing problem variants[16,8]; oddly enough, there are none, as far as the authors know, reporting the performance of these methods for the PCTSP.

The remainder of this paper is organized as follows: The notation and the formulation are described in §2. §3 details the devised algorithm. Finally, the computational experiments and the final remarks are presented in §4 and §5, respectively.

2 Notation and Formulation

Let $G = (V, A)$ be a complete, directed graph in which V and A are the sets of nodes and arcs, respectively, and $|V| = n$. For each node $i \in V$, there is a penalty γ_i and a prize p_i associated with it. Each arc $(i, j) \in A$ has a travel cost c_{ij} . The origin node s has a fixed prize $p_s = 0$ and penalty $\gamma_s = \infty$. The minimum amount of prize to be collected is given by p_{min} . The formulation requires the integer variables $x_{ij} \in \{0, 1\}$ which is equal to one, if arc $(i, j) \in A$ is part of the tour; or to zero, otherwise; and $y_i \in \{0, 1\}$ if node $i \in V$ is visited ($y_i = 1$) or not ($y_i = 0$); and the non-negative variables $f_{ij} \geq 0$ used to avoid sub-tours. The model for the PCTSP can be written as:

$$\min \phi(x) = \sum_{(i,j) \in A} c_{ij}x_{ij} + \sum_{i \in V} \gamma_i(1 - y_i) \quad (1)$$

$$s.t.: \sum_{(i,j) \in A} x_{ij} = y_i \quad \forall i \in V \quad (2)$$

$$\sum_{(i,j) \in A} x_{ij} = y_j \quad \forall j \in V \quad (3)$$

$$\sum_{i \in V} p_i y_i \geq p_{min} \quad (4)$$

$$\sum_{(j,i) \in A} f_{ji} - \sum_{(i,j) \in A} f_{ij} = y_i \quad \forall i \in V \quad (5)$$

$$f_{ij} \leq (n - 1)x_{ij} \quad \forall (i, j) \in A \quad (6)$$

$$x_{ij} \in \{0, 1\} \quad \forall (i, j) \in A \quad (7)$$

$$y_i \in \{0, 1\} \quad \forall i \in V \quad (8)$$

$$f_{ij} \geq 0 \quad \forall (i, j) \in V \quad (9)$$

The objective function (1) minimizes the total travel costs and the incurred penalties. Constraints (2) and (3) ensure that if a node i is visited, then only

one arc arrives at it and one leaves this node. Constraint (4) guarantees the minimum prize amount to be collected. Constraints (5)-(6) avoid the existence of sub-tours.

3 The proposed tabu search algorithm

The TS [12] is a non-stochastic iterative metaheuristic which has the ability of escaping from local optima solutions. In each iteration, the search moves towards the best solution in the neighborhood by not accepting movements that lead to solutions recently visited. To avoid these moves, a list, called tabu list, containing the information of these recent movements is kept for a time limit or for a few iterations. The amount of iterations that a movement is labeled tabu is known as tenure. The tenure and the list size are parameters of the TS algorithm. Although the tabu list objective is to avoid the cycling of solutions, whenever a movement yields a solution better than the best solution found so far, then this move is executed even though it may be on the tabu list. It is said that the move reached an aspirational criteria. The TS has two phases: In the first one, an initial feasible solution is built; while, in the second, the solution is refined.

3.1 Constructing the initial solution

The initial solution is created by using an adapted version of the GENIUS method [11]. This method is simple to implement and efficient to generate good solutions. It is divided into two steps: The first one, called GENI, generates an initial solution; while the second, named US, refines it. The GENI procedure iteratively builds a solution by inserting nodes into the tour. The insertion of a new node does not necessarily take place between two adjacent nodes. It considers any two nodes already in the tour, and verifies the two possible paths connecting these nodes. It then computes the cost of inserting the new node by analyzing which arcs will be deleted and which path segments will be reversed, and the produced cost. Usually, depending which two nodes are selected to be analyzed, there are two possible ways of insertion. The verification is carried on for all pair of nodes already in the tour and the least cost insertion is done. After a feasible solution is obtained by the GENI, the US step is performed. A node is removed from the solution–unstringing (U) the tour–and it is then reinserted into the tour–stringing (S) the tour–by using the procedures of the GENI step.

3.2 Solution refinement

The devised TS algorithm has three refining movements: *(i)* RemoveStep, in which a node is removed from the tour; *(ii)* InsertionStep, in which a node is inserted into the tour by using the GENIUS features; *(iii)* SwapStep, in which a node in the tour is exchanged by one that is not in the tour. At the begin of each TS iteration, a 3-opt is performed; while after each movement, a 2-opt [14] is applied to refine the current solution. The TS algorithm iterates for a pre-established maximum number of iterations, returning then the best solution found.

3.3 Intensification and Diversification

The TS algorithm may stay locked in a search space with no promising solutions or, likewise, it may not stay longer and miss therefore a good solution. In order to avoid this, a procedure to intensify and diversify the current solution is used. By keeping a record with the frequencies in which the nodes appear in the tour, whenever the current solution does not improve after a pre-established number of iterations, the procedure exchanges the nodes that are in the tour with the ones that are not, if the tour nodes have a larger frequency. The frequency records are then reset. Conversely, after the procedure is executed and no improvements are perceived in the current best solution for a fixed number of iterations, the procedure is once again performed, but by observing the nodes with the smaller frequencies now.

3.4 Outline of the devised algorithm

An outline of the devised algorithm is shown in Figure 1.

4 Computational experiments

The data set used is the same one of [4,6] and has instances with the number of nodes $n \in \{31, 51, 101, 251, 501\}$; and uniformly randomly generated values for arc cost $c_{ij} = U[50, 100]$, prize $p_i = U[1, 100]$ and penalty $\gamma_i = U[1, 750]$; and the $p_{min} = \frac{3}{4} \sum_{i \in V} p_i$. The TS algorithm was coded in C++, while the formulation (1)-(9) was solved by means of CPLEX 12.1 in order to get the optimal solutions for instances up to size 251. The computational tests were carried on a PC with a Dualcore processor with 2.00GHz, 3GB of RAM, running Windows XP. The TS algorithm was run for the maximum of 2,000 iterations; with a tenure being uniformly generated using $U[3, 6]$; and a tabu

```

Tabu Search algorithm:
begin
    Input:  $G(V, A)$ ,  $p_{min}$ ,  $p$ ,  $\gamma$ 
    Output:  $s^*$  //best solution found
     $s \leftarrow \text{GENIUS}(G, p_{min}, p, \gamma)$ 
     $s^* \leftarrow s$ 
    for  $h = 1$  to maximum number of iterations do
         $3\text{-opt}(s)$ 
         $s^1 \leftarrow \text{RemoveStep}(s)$ 
         $2\text{-opt}(s^1)$ 
         $s^2 \leftarrow \text{InsertionStep}(s)$ 
         $2\text{-opt}(s^2)$ 
         $s^3 \leftarrow \text{SwapStep}(s)$ 
         $2\text{-opt}(s^3)$ 
         $s \leftarrow \min(s^1, s^2, s^3)$ 
        AspirationCriteria( $s, s^*$ )
        IntensifyDiversify( $s, s^*$ )
    end for
end

```

Fig. 1. Pseudocode for the TS algorithm.

list size equal to n . The intensification and diversification procedure was called whenever the best solution was not improved for the last 50 iterations.

Each instance was solved 30 times using a different seed (1000, 2000, ..., 30000) for the C++ random number generator. The average results were then compared with the available optimal solutions and with the ones obtained by a clustering search (CS) approach presented in [6]. Table 1 reports the attained preliminary results for the devised TS algorithm and the ones obtained by the CS approach. Columns **Best**, **Average**, **AvDev** are the best overall solution found by the methods in all of the 30 runs, the average solution value attained, and the average percentage deviation from the best known solution, respectively. The CS approach obtained better results than the devised TS algorithm for instances up to 101 nodes. However for the larger size instances, the TS attained better results.

5 Final remarks

The devised TS algorithm with only basic features was able to provide better solutions for larger instances than a more elaborated method such as the Cluster Search approach of [4]. New movements and/or an adaptive memory scheme may be implemented to improve the efficiency of the proposed pro-

Table 1
Results of experiments - PCTSP

Instance	CPLEX	TS			CS		
		Best	Average	AvDev[%]	Best	Average	AvDev[%]
31a	3582	3582	3605	0.64	3582	3582	0.00
31b	2515	2515	2530	0.6	2515	2515	0.00
31c	3236	3236	3305	2.13	3236	3242	0.19
51a	4328	4328	4377	1.13	4328	4335	0.16
51b	3872	3872	3938	1.7	3872	3878	0.15
101a	6762	6846	6895	1.97	6785	6822	0.89
101b	6760	6824	6867	1.58	6782	6844	1.24
251a	14083	14385	14458	2.66	14483	14608	3.73
251b	13632	13934	14046	3.04	14078	14176	3.99
501a	-	26647	26747	-	27189	27230	-
501b	-	27049	27228	-	28133	28334	-

cedure. Further, better tuning of the TS parameters may be sought, as well as different values for p_{min} parameter should be investigate in order to better assess the performance of the algorithm.

References

- [1] Balas, E., *The prize collecting traveling salesman problem*, Networks **19** (1989), pp. 621–636.
- [2] Balas, E., *The prize collecting traveling salesman problem and its applications*, in: G. Gutin and A. Punnen, editors, *The Traveling Salesman Problem and Its Variations*, Combinatorial Optimization **12**, Springer US, 2004 pp. 663–695.
- [3] Bienstock, D., M. Goemans, D. Simchi-Levi and W. D., *A note on the prize-collecting traveling salesman problem*, Mathematical Programming (1993), pp. 413–420.
- [4] Chaves, A. A., F. L. Biajoli, O. M. Mine and M. J. F. Souza, *Metaheurísticas híbridas para resolução do problema do caixeiro viajante com coleta de prêmios*, Produção **17** (2007), pp. 263 – 272.
- [5] Chaves, A. A. and L. A. N. Lorena, *Hybrid algorithms with detection of promising areas for the prize collecting travelling salesman problem*, in: HIS, 2005, pp. 49–54.

- [6] Chaves, A. A. and L. A. N. Lorena, *Aplicao do algoritmo clustering search aos traveling salesman problems with profits*, in: *Anais 39 do Simpósio Brasileiro de Pesquisa Operacional*, Fortaleza, Brazil, 2007, pp. 1472–14830.
- [7] Chaves, A. A. and L. A. N. Lorena, *Hybrid metaheuristic for the prize collecting travelling salesman problem*, in: *EvoCOP*, 2008, pp. 123–134.
- [8] Cordeau, J.-F. and G. Laporte, *A unified tabu search heuristic for vehicle routing problems with time windows*, Journal of the Operational Research Society **52** (2001), pp. 928–936.
- [9] Dell'Amico, M., F. Maffioli and A. Sciomarchen, *A Lagrangian heuristic for the prize collecting traveling salesman problem*, Annals of Operations Research **81** (1998), pp. 289–306.
- [10] Fischetti, M. and P. Toth, “An additive approach for the optimal solution of the prize collecting traveling salesman problem,” Golden, B. L. and Assad, A. A., North-Holland, 1988 pp. 319–343.
- [11] Gendreau, M., A. Hertz and G. Laporte, *New insertion and post optimization procedures to the traveling salesman problem*, Operations Research **40** (1992), pp. 1086–1094.
- [12] Glover, F., *Tabu search - Part I*, ORSA Journal on Computing **1** (1989), pp. 190–206.
- [13] Gomes, L. M., V. B. Diniz and C. A. Martinhon, *An hybrid GRASP+VND metaheuristic for the prize collecting traveling salesman problem*, in: *Anais 32 do Simpósio Brasileiro de Pesquisa Operacional*, Viçosa, Brazil, 2000, pp. 1657–1665.
- [14] Lin, S., *Computer solutions of the traveling-salesman problem*, Bell System Technology Journal **44** (1965), pp. 2245–2269.
- [15] Lopez, L., M. W. Carter and M. Gendreau, *The hot strip mill production scheduling problem: A tabu search approach*, European Journal of Operational Research **106** (1998), pp. 317–335.
- [16] Toth, P. and D. Vigo, *The vehicle routing problem*, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2001 pp. 1–26.

Minimizing waiting times in a route design problem with multiple use of a single vehicle

Francisco Angel-Bello^{1,2}

*Department of Industrial and Systems Engineering
Tecnológico de Monterrey
Monterrey, Nuevo León, México*

Iris Martínez-Salazar³, Ada Alvarez⁴

*Graduated Program in Systems Engineering
Universidad Autónoma de Nuevo León
Monterrey, Nuevo León, México*

Abstract

In this study we introduce a routing problem with multiple use of a single vehicle and service time in demand points (clients) with the aim of minimizing the sum of clients waiting time to receive service. This problem is relevant in the distribution of aid, in disaster stricken communities, in the recollection and/or delivery of perishable goods and personnel transportation, among other situations, where reaching clients to perform service, fast and fair, is a priority. We consider vehicle capacity and travel distance constraints which force multiple use of the vehicle in the planning horizon. This paper presents and compares two mixed integer formulations for this problem, based on a multi-level network.

Keywords: Multi trip vehicle routing problems, Total Latency, Mixed integer model, Routing for relief.

¹ This work was partially sponsored by the Mexican National Council of Science and Technology (Grant CB 167019) and by the Research Chair in Industrial Engineering of Tecnológico de Monterrey (ITESM Research Fund CAT128).

² Email: fangel@itesm.mx

³ Email: iris.martinezslz@uanl.edu.mx

⁴ Email: ada.alvarezs@uanl.mx

1 Introduction

We study a single-vehicle routing problem with multiple trips (MTVRP) minimizing the total waiting time of customers. There are studies on capacitated vehicle routing problem (CVRP) allowing vehicles to do more than one trip per planning horizon, but all of them focus on minimizing the total distance traveled or the overall number of multiple trips (see [5]). However, in many practical situations, minimizing the clients total waiting time is essential. This objective appears in literature as minimizing the total latency of the system. The addressed problem can be seen as a customer-centric routing problem, which is important in a variety of contexts like the procurement of humanitarian aid in the event of natural disasters or the recipient optimization in wireless telecommunication systems. There are very few published works dealing with customer-centric routing problems [6]. They focus on the cumulative capacitated vehicle routing problem (CCVRP), minimizing the sum of arrival times at customers, but considering single use of vehicle. To the best of our knowledge, there are no publications in the literature for the problem addressed in this work. The main contribution of this paper is the development of two mixed integer formulations to represent the problem.

2 Problem Formulation

Let $G = (V, E)$ be an undirected complete graph. The node set is $V = \{0, 1, \dots, n, n+1, \dots, n+m-1\}$ where node 0 is the depot, nodes $1, \dots, n$ are the clients, nodes $n+1, \dots, n+m-1$ represent $m-1$ copies of the depot and m is an upper bound for the number of trips. Each customer i has a specific demand d_i and a service time s_i . For the depot and its copies, s_0 is the time for loading/unloading operations. E is the edge set; each edge $(i, j) \in E$ has a weight $t_{ij} = t_{ji}$ that represents the traveling time between nodes i and j . Q denotes the vehicle capacity. D will denote the maximum length for the overall spread time of the working shift.

The goal is to define a set of trips (working shift) servicing all the customers once, without violating neither the capacity constraint on each single route, nor the time limit D of a working shift and minimizing the total waiting time of all customers . A solution will be represented by a permutation of the nodes belonging to V , where the sum of the demands of nodes between two copies of the depot should be less or equal than Q . Our first choice was to adapt the classical 2-index VRP model and some models developed for MTVRP that minimize the total distance (see [2], [4]). Unfortunately, preliminary

experiments showed that they require more than 1 hour to reach the optimal solution even for instances with 10 clients and 2 trips. Therefore, we decided to explore models developed for latency problems, even though they do not allow more than one route per vehicle (see [7], [3], [1]). Based on them we will propose two formulations for the addressed problem.

To do that, we used the multi-level network shown in Figure (1), which depicts N copies of nodes $\{1, 2, \dots, N\}$ where $N = n + m - 1$. Note that there exist a one-to-one correlation between Hamiltonian paths and permutations, in the graphical representation this means that only one node should be selected on each level, and it should be different from those nodes selected on other levels. However, one more constraint is needed: a set of nodes between two nodes representing the depot should satisfy the capacity constraint.

As node 0 should occupy the first position in all permutations, we refer to this position as position 0 and it corresponds to level 0 on the graphic. Other positions (levels in the graphic) correspond with positions $1, 2, \dots, N$. The contribution to the objective function of each used arc in a feasible solution depends on the levels connected by this arc in the network. If the arc (i, j) connects level k with level $k+1$, then its contribution to the objective function is $(N - k)c_{ij}$, where $c_{ij} = s_i + t_{ij}$, $(i, j = 0, 1, \dots, n; j \neq i)$, $c_{ij} = c_{0j}$ ($i = n + 1, \dots, N; j = 1, 2, \dots, n$), $c_{ij} = c_{i0}$ ($i = 1, 2, \dots, n; j = n + 1, \dots, N$), $c_{ij} = 0$ ($i, j = 0, n + 1, \dots, N$). This graphical representation is based on the one proposed in [7] for time-dependent traveling salesman problem (TDTSP).

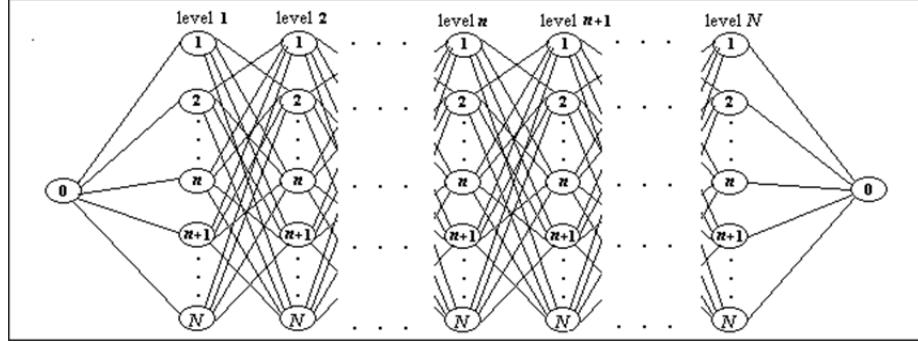


Fig. 1. Graphical representation of the MTVRP problem by a multi-level network

To establish the first formulation, let variables w^k be the cumulative demand until position k of the permutation, counted from the last depot (copy) node before position k ; x_i^k a binary variable equal to 1 if node i is in the position k in a permutation and y_{ij}^k a binary variable equal to 1 if node i is in the position k and node j in position $k + 1$ in a permutation.

Model I

$$\min z = N \sum_{i=1}^N c_{0i} x_i^1 + \sum_{i=1}^N \sum_{j=1, j \neq i}^N c_{ij} \sum_{k=1}^{N-1} (N-k) y_{ij}^k \quad (1)$$

subject to:

$$\sum_{k=1}^N x_i^k = 1 \quad (i = 1, \dots, N) \quad (2)$$

$$\sum_{i=1}^N x_i^k = 1 \quad (k = 1, \dots, N) \quad (3)$$

$$\sum_{j=1, j \neq i}^N y_{ij}^k = x_i^k \quad (i = 1, \dots, N; k = 1, \dots, N-1) \quad (4)$$

$$\sum_{j=1, j \neq i}^N y_{ji}^k = x_i^{k+1} \quad (i = 1, \dots, N; k = 1, \dots, N-1) \quad (5)$$

$$\sum_{i=1}^N c_{0i} x_i^1 + \sum_{i=1}^N \sum_{j=1, j \neq i}^N c_{ij} \sum_{k=1}^{N-1} y_{ij}^k + \sum_{i=1}^N c_{i0} x_i^N \leq D \quad (6)$$

$$w^1 = \sum_{i=1}^n d_i x_i^1 \quad (7)$$

$$w^{k+1} \geq w^k + \sum_{i=1}^n d_i x_i^{k+1} + Q \left(\sum_{i=1}^n x_i^{k+1} - 1 \right) \quad (k = 1, \dots, N-1) \quad (8)$$

$$w^{k+1} \leq Q \sum_{i=1}^n x_i^{k+1} \quad (k = 1, \dots, N-1) \quad (9)$$

$$\begin{aligned} x_i^k &\in \{0, 1\} \quad (i, k = 1, 2, \dots, N); \quad w^k \geq 0 \quad (k = 1, 2, \dots, N) \\ y_{ij}^k &\geq 0 \quad (i, j = 1, 2, \dots, N; j \neq i; k = 1, 2, \dots, N-1) \end{aligned} \quad (10)$$

Note that this model uses variables x_i^1 to calculate the contribution to total latency of any arc between positions 0 and 1, while variables y_{ij}^k for the contribution of arcs linking other consecutive positions. Variables x_i^N are useful for prevent exceeding the value of D . Constraints (2) guarantee that each node occupies a single position in any feasible solution, while constraints (3) guarantee that no more than one node occupy each position. Constraints

(4) ensure that from position k can leave just one arc and it is indeed from the node that occupies that position. Constraints (5) impose that at position $k + 1$ can arrive just one arc and it is indeed to the node that occupies that position. Constraint (6) ensures that the time limit D of a working shift is not exceeded. Constraints (7),(8) and (9) compute the cumulative demand and guarantee that the sum of the demands of nodes between two copies of the depot is less or equal than Q .

The second formulation (Model II below) can be obtained from the previous one by eliminating the binary variables x_i^k and forcing variables y_{ij}^k to be binary. To establish the model let us define additional binary variables $x_{0i}(x_{i0})$ equal to 1 if node i is in the position 1 (N) in a permutation.

Model II

$$\min z = N \sum_{i=1}^N c_{0i} x_{0i} + \sum_{i=1}^N \sum_{j=1, j \neq i}^N c_{ij} \sum_{k=1}^{N-1} (N - k) y_{ij}^k \quad (11)$$

Subject to:

$$\sum_{i=1}^n x_{0i} = 1 \quad (12)$$

$$\sum_{j=1, j \neq i}^N y_{ij}^1 - x_{0i} = 0 \quad (i = 1, 2, \dots, N) \quad (13)$$

$$\sum_{j=1, j \neq i}^N (y_{ij}^k - y_{ji}^{k-1}) = 0 \quad (i = 1, 2, \dots, N; k = 2, 3, \dots, N-1) \quad (14)$$

$$\sum_{j=1, j \neq i}^N y_{ji}^{N-1} - x_{i0} = 0 \quad (i = 1, 2, \dots, N) \quad (15)$$

$$\sum_{i=1}^N x_{i0} = 1 \quad (16)$$

$$\sum_{j=1, j \neq i}^N x_{0i} + \sum_{k=1}^{N-1} \sum_{j=1, j \neq i}^N y_{ji}^k = 1 \quad (i = 1, 2, \dots, N) \quad (17)$$

$$\sum_{i=1}^N c_{0i} x_{0i} + \sum_{i=1}^N \sum_{j=1, j \neq i}^N c_{ij} \sum_{k=1}^{N-1} y_{ij}^k + \sum_{i=1}^N c_{i0} x_{i0} \leq D \quad (18)$$

$$w^1 = \sum_{i=1}^n d_i x_{0i} \quad (19)$$

$$w^{k+1} \geq w^k + \sum_{i=1}^n d_i \sum_{j=1, j \neq i}^n y_{ji}^k + Q \left(\sum_{i=1}^n \sum_{j=1, j \neq i}^n y_{ji}^k - 1 \right) \quad (k = 1, 2, \dots, N-1) \quad (20)$$

$$w^{k+1} \leq Q \sum_{i=1}^n \sum_{j=1, j \neq i}^n y_{ji}^k \quad (k = 1, 2, \dots, N-1) \quad (21)$$

$$\begin{aligned} x_{0i}, x_{i0} &\in \{0, 1\} \quad (i = 1, 2, \dots, N); \quad w^k \geq 0 \quad (k = 1, 2, \dots, N) \\ y_{ij}^k &\in \{0, 1\} \quad (i, j = 1, 2, \dots, N; j \neq i; k = 1, 2, \dots, N-1) \end{aligned} \quad (22)$$

Constraints (12) ensure that a single node occupies position 1 in the permutation. Constraints (15) guarantee that from position 1 leaves just one arc and it is indeed from the node that occupies that position. Constraints (14) are equivalent to (4)–(5). Constraints (15) and (16) are similar to constraints (13) and (12) respectively, but for the arc that arrives to position N . Constraints (17) guarantee no repetition of nodes among positions. Constraints (18), (19), (20) and (21) have the same spirit that (6), (7), (8) and (9) respectively.

The main difference between both models is in the capacity constraints: Model I uses position variables (constraints (8) and (9)), while Model II uses arc variables (constraints (20) and (21)). This fact motivates that the values of linear relaxation are better for Model I, as will be shown in the next section.

It should be remarked that when m is overestimated the solution may include empty routes, which are represented by consecutive nodes associated to copies of the depot. However, due to the objective function they will be at the beginning of the permutation, as these arcs have zero cost.

3 Computational experiments

Instances with 10, 15, 20, 25 and 30 clients were randomly generated from points with real coordinates using a uniform distribution between 0 and 100, taking the rounded Euclidean distances as travel times t_{ij} . The number of trips was fixed in 2 for 10-clients instances and 3 for other sizes. Depending on the size of the instance, Q was set to 120, 120, 140, 180, 200. Demand d_i was randomly assigned with a value of 10, 20 or 30 in such a way that the sum of the demands is among $(m+1)Q$ and mQ to ensure feasibility. Service time was the same for all clients. There are 25 instances for each value of n .

Optimal solutions were found using Cplex 12.4, run in an Intel Core i7–2600 CPU at 3.40GHz and 8 GB RAM processor. Column 1 in Table 1 displays the number of clients of the instances, while column 2 indicates the number of routes. Average values were calculated over 25 instances. The minimum, maximum and average elapsed CPU time, in seconds, are shown from column 3 to 5 for Model I, and from column 6 to 8 for Model II. The symbol “–” means that the solver was unable to find an optimal solution due to lack of memory. Column 9 and 11 show the average gap (%) between the linear programming (LP) relaxation and the optimal solution for Model I and Model II, respectively. Columns 10 and 12 present the average CPU time in seconds for solving the LP relaxation of Model I and Model II respectively.

Instance size		Model I			Model II			Model I		Model II	
<i>n</i>	<i>m</i>	Min	Max	Average	Min	Max	Average	Gap	Time	Gap	Time
10	2	1.24	3.13	1.92	1.11	2.15	1.439	27.70	0.03	27.88	0.03
15	3	8.88	112.82	34.27	38.66	525.90	185.45	44.52	0.11	45.02	0.11
20	3	61.62	8860.27	699.13	140.89	19279.5	5329.98	44.54	0.20	44.91	0.33
25	3	338.93	31887.5	6003.69	–	–	–	42.17	0.41	42.43	0.53

Table 1
Models comparison concerning CPU time and linear relaxation

As can be observed, Model I solve to optimality all instances up to 25 clients. For 30-client instances, it could solve to optimality 23 out of 25 instances, for the remaining 2 instances, CPLEX reported an average gap of 14.17% within 9 hours limit. On the other hand, Model II solve to optimality all instances up to 20 clients. For 25-client instances, it could solve to optimality 12 out of 25 instances, for the remaining 13 instances, CPLEX reported an average gap of 12.15% within 9 hours limit.

Models I and II can be considered as generalizations of previous models developed for TDTSP in [7] and [3], which are equivalent in terms of LP relaxation [3]. However, for the generalizations developed in this work, Model I shows better values of LP relaxation and less CPU time. This is due to Model I and Model II developed in this work use different ways to express the capacity constraints as was explained in section 2.

4 Conclusions

For the single-vehicle routing problem with multiple routes and the objective of minimizing the total latency, the strategy of adapting a minimum latency mathematical model, to include a single vehicle multi trip vehicle routing features, is a better approach than its counterpart. Despite of the fact that formulations like Model II have lately attracted the interest of the researchers, Model I have a better performance for the problem addressed in this paper.

References

- [1] Angel-Bello, F., A. Alvarez and I. García, *Two improved formulations for the minimum latency problem*, Applied Mathematical Modelling **37** (2013), pp. 2257–2266.
- [2] Azi, N., M. Gendreau and J.-Y. Potvin, *An exact algorithm for a single-vehicle routing problem with time windows and multiple routes*, European Journal of Operational Research **178** (2007), pp. 755–766.
- [3] Gouveia, L. and S. Voss, *A classification of formulations for the (time-dependent) traveling salesman problem*, European Journal of Operational Research **83** (1995), pp. 69–82.
- [4] Koc, C. and I. Karaoglan, *A branch and cut algorithm for the vehicle routing problem with multiple use of vehicles*, in: *41st International Conference on Computers & Industrial Engineering*, 2011, pp. 554–559.
- [5] Macedo, R., C. Alves, J. Valério de Carvalho, F. Clautiaux and S. Hanafi, *Solving the vehicle routing problem with time windows and multiple routes exactly using a pseudo-polynomial model*, European Journal of Operational Research **214** (2011), pp. 536–545.
- [6] Mattos Ribeiro, G. and G. Laporte, *An adaptive large neighborhood search heuristic for the cumulative capacitated vehicle routing problem*, Computers & Operations Research **39** (2012), pp. 728–735.
- [7] Picard, J. C. and M. Queyranne, *The time-dependent traveling salesman problem and its application to the tardiness problem in one-machine scheduling*, Operations Research **26** (1978), pp. 86–110.

Iterated Local Search for Fiber Installation in Optical Network Optimization

Daniel Morais dos Reis^{a,1,4}, Thiago F. Noronha^{b,3,4},
Sérgio R. de Souza^{a,2,4}

^a Departamento de Computação
Centro Federal de Educação Tecnológica de Minas Gerais
Belo Horizonte, Brazil

^b Departamento de Ciência da Computação
Universidade Federal de Minas Gerais
Belo Horizonte, Brazil

Abstract

The problem of Fiber Installation in Optical Network Optimization consists in routing a set of lightpaths (all-optical connections), such that the cost of the optical components necessary to operate the network is minimized. We propose a novel Iterated Local Search heuristic. Computational results showed that the new heuristic is better than the best heuristic in the literature.

Keywords: Optical Network Optimization, WDM, and Iterated Local Search.

¹ Email: daniel.morais@gmail.com

² Email: sergio@dppg.cefetmg.br

³ Email: tfn@dcc.ufmg.br

⁴ This work was partially supported by the Brazilian National Council for Scientific and Technological Development (CNPq), the Foundation for Support of Research of the State of Minas Gerais (FAPEMIG), and Coordination for the Improvement of Higher Education Personnel (CAPES).

1 Introduction

The *Wavelength Division Multiplexing* (WDM) technology allows a more efficient use of the huge capacity of optical fibers. It permits the simultaneous transmission of different optical signals simultaneously along the same fiber, provided they are multiplexed with different wavelengths. An all-optical point-to-point connection between two nodes is called a *lightpath*. Two lightpaths may be multiplexed with same wavelength, provided they do not share any common fiber.

Given an optical network and a set of lightpaths to be established, the problem of *routing and wavelength assignment* (RWA) in WDM optical networks consists in routing the set of lightpaths and assigning a wavelength to each lightpath, such that lightpaths whose routes share a common fiber are assigned different wavelengths. Variants of RWA are characterized by different optimization criteria and traffic patterns, see e.g. [2,7].

We consider a variant in which all lightpath demands are known beforehand. Let $N = (X, A)$ be an undirected graph representing the network topology, where X is the set of nodes (central offices) and A is the set of edges, which represent the optical links between offices. Let also $T = \{t_1, t_2, \dots, t_{|T|}\}$ be the set of demands of lightpaths, where each $t_i \in T$ is defined by a pair of nodes in X and the number of wavelengths necessary to carry the traffic between these nodes.

The problem of Fiber Installation in Optical Networks (FIP) consists in routing all demands in T while minimizing the cost of the optical devices necessary to operate the network. These equipments worth millions of dollars and any percentage of the network costs is not negligible. This problem was proposed in [1] as a part of an effort at Bell Labs to develop optimization methods for optical networks [1]. In this paper, we consider the three most expensive devices used in optical networks.

The Reconfigurable Optical Add/Drop Multiplexer (ROADM) device allows the optical signal to travel through a central office from any incoming fiber to any outgoing fiber with the same wavelength. This device reduces drastically the signal overhead, because it eliminates the optical-electrical and than the electrical-optical conversions at intermediate nodes of the lightpath. One ROADM can have many arms, each one supporting a WDM fiber. The cost of a ROADM is linearly proportional to its number of arms. For each WDM fiber used, we need two ROADM arms placed at the extremes of the fiber. As one ROADM has to be installed at every central office, the optimization model only accounts for the number of ROADM arms. Therefore,

for each WDM fiber used, it is charged $c_1 = 2 \cdot C_{ROADM}$, where C_{ROADM} is the cost of a ROADM arm. The cost of the fiber itself is not considered, because it is assumed that the fibers are already deployed and are inactive or operating with an outdated technology.

The Optical Transponder (OT) device converts electrical signals coming from outside the network to optical signals and converts optical signals leaving the network into electrical signals. A lightpath originates in an OT, passes through one or more ROADM and ends in another OT. However there is a limit L_{OT} on the length of a lightpath. If a lightpath is longer than L_{OT} kilometers, it must be split into two lightpaths, and another pair of OT is needed. The cost of OT was modeled in [1] as follows. For each wavelength used in each WDM fiber, it is charged $c_3 = 2 \cdot \frac{C_{OT}}{L_{OT}}$ per kilometer of fiber, where C_{OT} is the cost of an OT (typically 5%-10% of the cost of a ROADM arm).

The Optical Amplifier (OA) device compensates the chromatic dispersion. The latter is a phenomenon that causes the decrease in power of the optical signals, which may prevent it to arrive at its final destination. According to [1], the cost of OA can be approximated to the first order by distance-based rules as the following. For each WDM fiber used, it is charged $c_2 = \frac{C_{OA}}{L_{OA}}$ per kilometer of fiber, where L_{OA} is the reach of an OA.

The total network cost can be calculated by summing the device cost of each individual network link. The device cost of transmitting ω wavelengths through a WDM link of length l is given by the function $F_{WDM}(\omega, l)$ (Equation 1), where μ is the number of wavelengths supported by a WDM fiber (typically $\mu = 100$), and $\left\lceil \frac{\omega}{\mu} \right\rceil$ is the number of WDM fibers needed to carry the ω wavelengths.

$$(1) \quad F_{WDM}(\omega, l) = c_1 \cdot \left\lceil \frac{\omega}{\mu} \right\rceil + c_2 \cdot l \cdot \left\lceil \frac{\omega}{\mu} \right\rceil + c_3 \cdot \omega \cdot l$$

In [1], the authors argued that the routing and the wavelength assignment subproblems should be tackled separately. They suggest that solution approaches for FIP should focus only on the routing subproblem since the wavelength assignment subproblem can be solved by any algorithm for the well studied Graph Coloring Problem in a conflict graph $G = (V, E)$, where the vertices correspond to the lightpaths in T , i.e. $V = \{v_i : t_i \in T\}$, and there is an edge $e \in E$ between each pair of vertices whose associated routes share a common fiber. The lightpaths whose corresponding nodes have the same color are assigned to the same wavelength.

The state-of-art heuristics for FIP are presented in Section 2. In Section 3,

we propose a new heuristic for FIP based on the iterated local search meta-heuristic. Computational experiments and concluding remarks are showed in Section 4.

2 Related Works

There are only heuristic algorithms in the literature of FIP. The two greedy heuristics and local search proposed in [1] are presented in Section 2.1, and the genetic algorithm of [3] is presented in Section 2.2.

2.1 Greedy and PSC Heuristics

The first heuristic proposed in [1], **Greedy**, is a greedy heuristic followed by a local search procedure, while the second, **PSC**, is similar to **Greedy** but uses an alternative piecewise strongly concave function instead of F_{WDM} to route the lightpaths. First, they build a permutation of the lightpath demands. Next, they iterate over this permutation. At each iteration, they calculate the route to the current demand that results in the least increasing in their respective greedy cost functions. Then, they stop when all lightpaths are routed.

Once all the demands are routed, it is possible that cheaper routes for each lightpath may exist as a result of the additional devices installed to satisfy the subsequent demands. Therefore, these heuristics perform a local search on a *1-opt* neighborhood. Let s be a feasible solution for FIP, a solution s' is a neighbor of s if and only if it differs from s by the route of at most one lightpath. The local search procedure subsequently passes over the demand permutation removing the existing route to the current demand and finding the least expansive route under the present circumstances (possibly the current one). It stops when a local optimum is reached, i.e., when no better solution can be obtained by re-routing a single lightpath.

While **Greedy** uses F_{WDM} (Equation 1) as its greedy function, **PSC** uses a more sophisticated approach based on a piecewise strongly concave greedy function. Let (i) μ be the number of wavelengths supported by an optical fiber, (ii) p be the current pass of the heuristic over the permutation of lightpath demands, (iii) P be an integer parameter to be tuned, and (iv) $f(x) = \lfloor x \rfloor + \sqrt{x - \lfloor x \rfloor}$. The greedy function used by PSC is given in Equation (2).

$$(2) \quad F'_{WDM}(\omega, l, p, P) = (c_1 + c_2 \cdot l) \cdot \left[\frac{p}{P} \cdot \left\lceil \frac{\omega}{\mu} \right\rceil + \frac{P-p}{P} \cdot f\left(\frac{\omega}{\mu}\right) \right] + c_3 \cdot \omega \cdot l$$

In the first pass of the local search procedure of **PSC** over the demand

permutation, p is equal to zero. Therefore, $F'_{WDM} = (c_1 + c_2 \cdot l) \cdot f\left(\frac{\omega}{\mu}\right) + c_3 \cdot \omega \cdot l$. At every pass of PSC over the demand permutation, the value of p is incremented, and F'_{WDM} gradually converges to F_{WDM} . The speed of the convergence is defined by the parameter P , which was set to 5 in [1].

2.2 GA-FIP Heuristic

The genetic algorithm for Fiber Installation in Optical Networks **GA-FIP** was proposed in [3]. It evolves a population of chromosomes that consists of vectors of real numbers (called keys). Solutions are represented as following. There is one real number in the range $[0,1]$ for each lightpath demand in T , and the solution represented by a chromosome is decoded by the following two step procedure. First, the lightpath demands are sorted in non-increasing order of their key values. The resulting order is used as the permutation in which the lightpath demands are processed by the heuristic PSC. The cost of the solution returned by PSC is used as the fitness of the chromosome.

GA-FIP implements an elitism mechanism in which a set with the best chromosomes of each generation is copied without change to the next generation. It uses the *parameterized uniform crossover* scheme proposed in [6] to combine two parent chromosomes and produce an offspring. In this scheme, the offspring inherits each of its keys from the best fit of the two parents with probability 0.7 and from the least fit parent with probability 0.3. This genetic algorithm does not make use of the standard mutation operator, where parts of the chromosomes are changed with a small probability. Instead, the concept of *mutants* is used. In each generation, a fixed number of mutant solutions are introduced in the population. They are generated randomly in the same way as the initial population. As with mutation, mutants play the role of helping the procedure to escape from local optima.

3 Solution Approach

The motivation for using the Iterated Local Search (ILS) metaheuristic [4] is based on the fact that recently it was successfully used in [5] for routing and wavelength assignment in WDM optical networks, as well as in solution approaches for many other combinatorial optimization problems.

The **ILS-FIP** heuristic proposed in this work makes use of the same 1-*opt* local search of [1], where every lightpath is removed from the network and reinserted with the least expansive route under the present circumstances (possibly the current one). Given a solution s , the perturbation procedure

```

begin ILS-FIP( $s^i$ )
1.  $s, s^* \leftarrow LocalSearch(1\text{-}opt, s^i);$ 
2. while stopping condition is not met do
3.    $s' \leftarrow Perturbation(k\text{-}opt, s);$ 
4.    $s' \leftarrow LocalSearch(1\text{-}opt, s');$ 
5.   if  $s'$  is better than  $s^*$  then  $s^* \leftarrow s'$ ;
6.   if AcceptanceCriterion( $s, s'$ ) then  $s \leftarrow s'$ ;
7. end-while;
8. return  $s^*$ ;
end

```

Fig. 1. Pseudo-code of the ILS-FIP heuristic.

applied by ILS-FIP consists on randomly selecting a k -*opt* neighbor of s as following. First, a set K with $k = |K|$ lightpaths are selected at random. The probability of a lightpath being selected is proportional to the number of wavelength it carries. Next, the lightpaths in K are removed from s . This may result in a decreasing in the network cost, as the equipments used to carry the traffic of the lightpaths in K are no longer needed. Then, each lightpath in K is iteratively reinserted in s with the least expansive route under the present circumstances.

Figure 1 shows the pseudo-code of ILS-FIP. It starts from an initial solution s^i provided by the heuristic Greedy [1]. First, the 1-*opt* local search is applied to s^i in Line 1, in order to produce the first local optimum s . Next, the loop in lines 2-7 is performed until a stopping condition is met. A new local optimum s' is obtained by applying the k -*opt* perturbation to s in Line 3, followed by the 1-*opt* local search to the perturbed solution in Line 4. Then, if the resulting solution s' is better than the best known solution s^* , the latter is updated in Line 5. Following, the current local optimum s is replaced by s' accordingly to a defined acceptance criterion in Line 6. Solution s' is accepted if its cost is smaller than the cost of s . Finally, the best solution found by this heuristic is returned in Line 8.

4 Experiments and Concluding Remarks

Heuristics GA-FIP and ILS-FIP [3] were implemented in C++ and compiled with the GNU gcc compiler version 4.4.3. The experiments were performed on an Intel Core 2 Quad 2.66 GHZ with 4Gb of RAM memory. As in [3], the size of sets TOP , $REST$, and BOT of GA-FIP was set to $0.25 \cdot |X|$, $0.75 \cdot |X|$ and

$0.05 \cdot |X|$ respectively. Preliminary experiments showed that the best value for the parameter k of ILS-FIP was $0.3 \cdot |T|$.

Table 1 shows the comparison on the expected cost of ILS-FIP and GA-FIP. The instances were downloaded from the Survivable Network Design Library - SNDlib (<http://sndlib.zib.de/>). Column 1 displays the instance name. Columns 2 to 4 show the number of nodes, the number of links, and the number of lightpath demands on each instance, respectively. The average cost of five runs of GA-FIP on 10 minutes of processing time is given in Column 5. The same data is given for ILS-FIP on Columns 6. The last column displays the relative improvement of ILS-FIP over GA-FIP.

It can be observed that the new ILS-FIP heuristic proposed in this work outperformed the best heuristic in the literature GA-FIP on all instances tested. The improvement of ILS-FIP over GA-FIP was up to 18.07% on instance Abilene, and 7.47% on average.

References

- [1] S. Antonakopoulos and L. Zhang. Heuristics for fiber installation in optical network optimization. In *Proceedings of 2007 IEEE Global Telecommunications Conference*, Washington, 2007.
- [2] J. S. Choi, N. Golmie, F. Lapeyrere, F. Mouveaux, and D. Su. A functional classification of routing and wavelength assignment schemes in DWDM networks: Static case. In *Proceedings of the 7th International Conference on Optical Communication and Networks*, pages 1109–1115, Paris, 2000.
- [3] N. Goulart, L. G. S. Dias, S. R. Souza, and T. F. Noronha. Biased random-key genetic algorithm for fiber installation in optical network optimization. In *2011 IEEE Congress on Evolutionary Computation*, pages 2267–2271, New Orleans, 2011. IEEE.
- [4] Helena R. Loureno, Olivier C. Martin, and Thomas Sttzle. Iterated local search. In *Handbook of Metaheuristics, volume 57 of International Series in Operations Research and Management Science*, pages 321–353. Kluwer Academic Publishers, 2002.
- [5] A. X. Martins, C. Duhamel, P. Mahey, R. R. Saldanha, and M. C. de Souza. Variable neighborhood descent with iterated local search for routing and wavelength assignment. *Comput. Oper. Res.*, 39(9):2133–2141, 2012.
- [6] W. Spears and K. deJong. On the virtues of parameterized uniform crossover. In *Proceedings of the Fourth International Conference on Genetic Algorithms*, pages 230–236, San Mateo, 1991.

Table 1
Comparison of ILS-FIP with GA-FIP.

Name	$ X $	$ A $	$ T $	GA-FIP	ILS-FIP	$\frac{\text{GA} - \text{ILS}}{\text{GA}}$
Abilene	12	15	66	2590862.43	2122658.06	18.07%
Atlanta	15	22	105	35357466.00	30467837.00	13.83%
Cost266	37	57	666	7586708.00	7222385.50	4.80%
France	25	45	300	83468964.00	79487045.50	4.77%
Geant	22	36	231	9183407.50	8210787.00	10.59%
Germany50	50	88	662	3565288.88	3340440.50	6.31%
Janos-US-CA	39	122	741	9752470.25	9518100.75	2.40%
Nobel-EU	28	41	378	5074613.00	4894050.13	3.56%
Nobel-Germany	14	21	121	827165.97	733111.09	11.37%
Nobel-US	14	21	91	5007265.25	4465302.50	10.82%
Normay	27	51	351	155783096.00	149634760.50	3.95%
Polska	27	18	66	757454.92	731700.05	3.40%
Ta2	27	108	807	171030828.00	162910628.00	4.75%
Zib54	27	81	626	150975312.00	142060596.50	5.90%
Average:						7.47%

- [7] H. Zang, J. P. Jue, and B. Mukherjee. A review of routing and wavelength assignment approaches for wavelength-routed optical WDM networks. *Optical Networks Magazine*, 1:47–60, 2000.

Team Orienteering Problem with Decreasing Profits

H. Murat Afsar*¹ Nacima Labadie*²

*ICD-LOSI, STMR UMR 6279 CNRS

University of Technologie of Troyes

12, rue Marie Curie, CS 42060, 10004, Troyes, France

Abstract

Team Orienteering with Decreasing Profits (DP-TOP) extends the classical Team Orienteering problem (TOP) by considering the profit of each client as a decreasing function of time. It consists of maximizing the sum of collected profit by a fixed number K of vehicles, visiting each client at most once. In this work, we present lower bounds based on a Dantzig-Wolfe decomposition and column generation as well as upper bounds obtained by an evolutionary local search approach (ELS).

Keywords: Team Orienteering problem, variable profits, column generation, ELS

1 Introduction

Orienteering problems are initially inspired by a sport game, where each competitor has to build a path between the starting and end points. They obtain a score (or *profit*) every time they visit a check point. The objective is to maximize the scores collected. This problem was defined by Tsiligrides [10]

¹ Email: murat.afsar@utt.fr

² Email: nacima.labadie@utt.fr

and consists in building a single tour so that the total profit collected is maximized, subject to time restriction on the tour length. When more than one tours are to be built ($K > 1$), the problem is called Team Orienteering problem and was described the first time by Chao [1]. For a survey on routing problems with profits see Vansteenwegen et al. [11] and Feillet et al. [6].

This paper is dedicated to an extension of the classical Team Orienteering problem where the profit associated to each node is not constant but is a decreasing function of time. To our knowledge, this problem is not addressed before despite its importance in real life applications. As an example, in the field of repairing-maintenance, a client can be ready to pay more to be serviced earlier and can loose his interest as the time passes by. Another application can be found in search and rescue operations after an earthquake for example. For the success of such operations a quick and reliable situation assessment is crucial. There may be several positions where the survivors can be found with different probabilities decreasing with time. The aim is to visit as many positions as possible with higher probabilities.

The paper is organized as follows: a formal problem definition is given in section 2. A column generation approach and a metaheuristic method based on Evolutionary Local Search (ELS) are proposed in sections 3 and 4, respectively. The results obtained by both approaches are presented in section 5. Finally, section 6 is dedicated to concluding remarks and some perspectives.

2 Problem Definition

The Team Orienteering Problem with Decreasing Profits can be described on an undirected graph $G = (V, E)$ where $V = \{0, 1, \dots, n+1\}$ is the set of nodes. Nodes $1, \dots, n$ are potential customers to visit, whereas nodes 0 and $n+1$ correspond respectively to beginning and end points of the paths to build. To each customer $i \in \{1, 2, \dots, n\}$, a variable profit (m_i) and a fixed profit (n_i) are associated. The profit collected at node i if it is visited at the moment t is equal to $m_i * t + n_i$. As it is mentioned, the profits are decreasing by time, so $m_i < 0 \ \forall i \in \{1, \dots, n\}$. E is the set of undirected edges, each edge $[i, j]$ defines a connection between nodes i and j and is weighted by w_{ij} , the time required to traverse it in any direction. From now on, for the sake of simplicity, each edge is replaced by two arcs of opposite directions with the same travelling time $w_{ij} = w_{ji}$. A fleet of K identical vehicles is available at node 0 and the total travel time of each vehicle is limited by a constant T_{max} .

The objective is to maximize the collected profit subject to the fleet size and time limitation of tours, by visiting at most once each potential customer.

Let x_{ij} be a binary variable taking the value 1 if the arc (i, j) is traversed and 0 otherwise. The visiting time of the node i is noted by the real variable t_i .

$$z^* = \max \sum_{i \in V} m_i t_i + \sum_{i \in V} \sum_{j \in V} x_{ij} n_j \quad (1)$$

subject to

$$\sum_{j \in V \setminus \{0, n+1\}} x_{0j} \leq K \quad (2)$$

$$\sum_{j \in V \setminus \{0\}} x_{ij} \leq 1 \quad \forall i \in V \setminus \{n+1\} \quad (3)$$

$$\sum_{j \in V \setminus \{0\}} x_{ij} - \sum_{j \in V \setminus \{n+1\}} x_{ji} = 0 \quad \forall i \in V \setminus \{0, n+1\} \quad (4)$$

$$t_j + w_{j,n+1} - \sum_{i \in V \setminus \{n+1\}} x_{ij} T_{max} \leq 0 \quad \forall j \in V \setminus \{n+1\} \quad (5)$$

$$t_i + x_{ij}(w_{ij} + T_{max}) - t_j \leq T_{max} \quad \forall i \in V \setminus \{n+1\} \quad \forall j \in V \setminus \{0\} \quad (6)$$

$$x_{ij} \in \{0, 1\} \quad \forall i \in V \setminus \{n+1\} \quad \forall j \in V \setminus \{0\} \quad (7)$$

$$t_i \geq 0 \quad \forall i \in V \quad (8)$$

The objective function 1 maximizes the sum of collected profits. The constraint 2 limits the fleet size. Each node is visited at most once because of constraints 3. Constraints 4 are flow constraints. The respect of time limitation on the tours is guaranteed by the constraints 5. Constraints 6 are used to define the visiting time of each node. Finally constraints 7 and 8 fix the nature of variables x_{ij} (binary) and t_i (non negative).

3 Column Generation

A Dantzig-Wolfe decomposition is proposed to reformulate the model 1-8: the master problem keeps the constraints 2 and 3, the rest of the constraints are pushed to the sub-problem. This method proved its value in several works such as in [3,4,5,8].

3.1 Restricted Master Problem

In the path reformulation of the DP-TOP, λ_r is the binary variable taking the value 1 if the route r is in the optimal solution. The total profit collected by the route r is noted as p_r . As the number of variables in the path reformulation is very large, the master problem is solved over a restricted subset of variables ($\Omega' \subset \Omega$) and the variables (*columns*) are generated dynamically by a *column*

generation procedure. The candidate columns are found by solving the subproblem. In the following model, γ_r^i is a parameter showing either the node i is serviced by the route r or not.

$$z_{LP}^* = \max \sum_{r \in \Omega'} p_r \lambda_r \quad (9)$$

subject to

$$\Pi_0 \longrightarrow \sum_{r \in \Omega'} \lambda_r \leq K \quad (10)$$

$$\Pi_i \longrightarrow \sum_{r \in \Omega'} \gamma_r^i \lambda_r \leq 1 \quad \forall i \in V \setminus \{0, n+1\} \quad (11)$$

$$\lambda_r \geq 0 \quad \forall r \in \Omega' \quad (12)$$

In the linear relaxation of the restricted problem (RMLP), Π_0 and Π_i are the dual variables associated with the fleet size restriction (10) and covering (11) constraints.

3.2 Sub-problem

The sub-problem seeks feasible routes (or *columns*) subject to the constraints (4-8) which are supposed to improve the objective function (9). A route is feasible if and only if:

- It starts at 0 and ends at $n+1$,
- Its total duration does not exceed T_{max} ,
- It does not visit a node more than once.

The route r *may* improve the objective function if its reduced cost is positive ($\bar{p}_r = p_r - \sum_{i \in V} \gamma_p^i \Pi_i - \Pi_0$). This is an elementary longest path problem with decreasing prices. We can reformulate it to have a classical elementary shortest path problem by:

$$\bar{p}_r = \sum_{i \in V} \gamma_r^i (\Pi_i - m_i t_i - n_i) + \Pi_0$$

We use a dynamic programming approach which is a modified version of the method developed in [7]. A route r weakly dominates r' if and only if:

- p and p' have the same last node i ,
- $\bar{p}_r \leq \bar{p}_{r'}$,

- Total duration of r is smaller than r' ($t_i^r \leq t_i^{r'}$ where t_i^r is the visiting time of node i by the route r),
- All accessible nodes for p' should also be accessible also for p .

4 Evolutionary Local Search

Evolutionary Local Search (ELS) metaheuristic was proposed by Merz and Wolf [12] for a peer-to-peer problem in telecommunications. The general structure of the implemented ELS (see Algorithm 1) starts by generating two initial solutions (with two constructive heuristics H_1 and H_2) which are then improved by a local search procedure (LS). The best incumbent solution is considered as the starting one in the ELS. Each iteration of the algorithm considers the best current solution which is perturbed several times to generate NBC child solutions, these resulting child solutions are then improved by Local Search (LS). For the next iterations, the best child solution replaces the best solution every time this later is improved, and the parameter p used to control the perturbation rate is reset to its minimal value p_{min} .

Algorithm 1 : General structure of the ELS

```

Compute  $s_1$  and  $s_2$  with  $H_1$  and  $H_2$  resp.
 $s_1 := LS(s_1); s_2 := LS(s_2); BestSol := argmax(z^*(s_1), z^*(s_2));$ 
 $p := p_{min}$ 
for  $Iter := 1$  to  $MaxIter$  do
     $z^*(BestChild) := 0$ 
    for  $k := 1$  to  $NBC$  do
         $ChildSoln := BestSoln$ 
         $Perturb(ChildSoln, p)$ 
         $LS(ChildSoln)$ 
        if  $z^*(ChildSoln) > z^*(BestChild)$  then
             $BestChild := ChildSoln$ 
        end if
    end for
    if  $z^*(BestChild) > z^*(BestSoln)$  then
         $BestSoln := BestChild$ 
         $p := p_{min}$ 
    else
         $p := min(p_{max}, p + 1)$ 
    end if
end for

```

4.1 Initial solutions

Two constructive heuristics (denoted H_1 and H_2) have been developed to build the initial solutions used in the ELS. They are both based on insertion mechanism. In the first heuristic (H_1), K tours are initialized each with a customer

for which $\tau_i = \frac{n_i}{w_{0,i} + w_{i,n+1}} - m_i$ is maximal. The tours are then extended by inserting at each iteration a node j in the best possible position after node i for which $\frac{PVar(i,j)}{\delta_{i,j}}$ is maximal. $PVar(i,j)$ and $\delta_{i,j}$ stand respectively for the variation of tour profit and tour duration, if j is inserted after i . As for $\delta_{i,j}$, the computation of $PVar(i,j)$ can be done in $O(1)$ since for each customer i visited in a given tour at position k , we keep in memory the sum of variable profits m_j of all customers j visited in previous positions $1, 2, \dots, k$ in the same tour. Because of the decreasing profits, some customers may become non profitable from a particular moment $t^* \leq T_{max}$ (for a customer i , this moment is such that $m_i + t^*n_i = 0$). Therefore, as for vehicle routing problems with time windows, the maximum allowable delay keeping the customer profitable is also recorded. Thanks to these precautions, the variation of both profit and duration of a tour before an insertion can be made in $O(1)$.

In the second heuristic H_2 , the K tours are initialized as in H_1 . The tours are extended in the next iterations by considering only the insertion positions after the last customer of each tour. The criterion used is hence different: for each potential customer i and each tour $k = 1, \dots, K$ ending at customer j_k , the value $\frac{n_i}{w_{j_k,i} + w_{i,n+1}} - m_i$ is computed and the customer giving its maximum to this criterion is inserted at the end of the corresponding tour. Both algorithms stop when there is no further possible insertion.

4.2 Perturbation procedure

The perturbation procedure has the role of diversification in the ELS. It is a crucial component since if the perturbation introduces strong mutation on a solution, the quality of this last could be deteriorated considerably and when the solution is not sufficiently perturbed, the algorithm would be trapped quickly in a local optima. For this reason, we proposed a perturbation mechanism which remove some customers from the current solution and replace them by other customers not visited. It consists in considering each tour of the solution and removing p customers randomly chosen from the tour. Each emptied space is fulfilled by adding non visited customers which are selected such as to increase the tour profit as much as possible. The perturbation rate is controlled thanks to a parameter p which is controlled dynamically between two values p_{min} and p_{max} as proposed in [9].

4.3 Local Search

The local search contains two classical moves usually used in routing problems (*Or-Opt*, *Exchange*) and two new moves more suitable to problems with profits

(*AddNew*, *ReplaceSequence*). These moves are tested in the order given by the list below and the search stops when no improving move can be found.

- *AddNew*: adds a customer not yet visited to the current solution
- *Or-opt*: relocates one customer already visited in the solution
- *Exchange*: swaps the positions of two customers
- *ReplaceSequence*: deletes a sequence of at most β consecutive customers and replaces it with a new sequence of non visited customers.

5 Experimental Results

The computational experiments were driven on TOP instances proposed by Chao et al. [2] for which we only added the variable profits. There are 7 data sets with the number of vertices $n+1 = 21, 32, 33, 64, 66, 100$ and 102. The starting and the ending points are assumed to be distinct in these instances. The problems within each data set differ in the maximal duration of the tour and in the number of tours. The number of tours ranges from 2 to 4. Due to the lack of rooms and the important number of files (a total of 387), only the results of the Set 5 (with $n+1 = 100$) are presented. This set contains 60 files (20 files for each value of $K = 2, 3, 4$).

The ELS was executed on a Dell Latitude E6420 equipped with 2.4 GHz intel core i7-2760 processor and 8 GB of RAM. It was coded in Delphi, a Pascal-like programming environment. The column generation approach was written in C++ and the commercial solver CPLEX was used to solve the mathematical models in an Intel Core i5-3570 processor clocked at 3.40GHz \times 4.

After several tests, the ELS parameters were set to the following values: $MaxIter = 100$, $NBC = 50$, $p_{min} = 2$, $p_{max} = 4$ and a maximum of $\beta = 4$ consecutive nodes are removed in the last neighborhood of the local search. Tables 1 and 2 summarize respectively the results obtained on Set 5 for the case where m_i are randomly generated in the intervals $[-2.50, 0[$ (Tab. 1) and $] -1.1, -0.1[$ (Tab.2). Both tables follow the same structure: columns 2 and 3 give respectively the average profit for each group of instances with $K = 2, 3, 4$, when the best heuristic (H_1) and the ELS are executed. Column 4 indicates the average running time achieved by the ELS. The three following columns are devoted to the column generation approach (CG) and indicate respectively for each group of files the average values of the upper bound (CG-UB) and lower bound (CG-LB) as well as running time (Time-CG). The last line of the tables give the mean over the three groups of instances of Set 5.

For the Set 5 and for $m_i \in [-2.50, 0[$, almost all of the 60 instances except eight are solved to optimality by CG, while the ELS is less competitive in termes of computational time and the quality of results. Furthermore, the

column generation procedure is very fast. Its average computational time is 0.11 seconds with a maximal running time of 0.36 seconds. For the case where $m_i \in] -1.1, -0.1[$ (Tab. 2) the column generation becomes more time consuming and the results shown in the three last columns are those obtained when the running time is limited to 600 seconds for each instance. CG still however more efficient than ELS in termes of obtained results. It is interesting to mention that execution time are very uneven for instances in the same group for $m_i \in] -1.1, -0.1[$. For example in Set 5, there are 13 files for $K = 2$, 9 for $K = 3$ and only 4 files for $K = 4$ for which the execution time is more than 200 seconds.

	HEUR	ELS	Time-ELS(s)	CG-UB	CG-LB	Time-CG (s)
$K = 2$	123.747	130.601	15.429	159.188	159.156	0.11
$K = 3$	141.824	150.966	17.582	185.408	185.160	0.13
$K = 4$	160.452	171.320	25.765	196.423	196.406	0.10
Aver.	141.023	149.872	19.297	179.387	179.286	0.11

Table 1
Summary of results on Set 5, $n + 1 = 100$, $m_i \in] -2.5, 0[$

	HEUR	ELS	Time-ELS(s)	CG-UB	CG-LB	Time-CG (s)
$K = 2$	254.119	277.400	32.497	320.233	319.523	349.558
$K = 3$	287.192	313.790	35.122	370.892	368.872	264.338
$K = 4$	311.552	353.036	37.959	407.116	405.821	148.498
Aver.	282.775	312.707	35.045	363.796	362.464	259.600

Table 2
Summary of results on Set 5, $n + 1 = 100$, $m_i \in] -1.1, -0.1[$

6 Conclusion and Perspectives

This work presents the first study dealing with the Team Orienteering with variable profits. This new variant is motivated by real applications in service sector for example. A mathematical model, a lower bound and upper bound based on column generation approach and Evolutionary Local search Metaheuristic are developed. Further research must be conducted in order to improve the performance of the ELS especially its local search. An interesting issue of this work is to understand why the methods behave differently for slightly higher values of negative variable profits. Future work would also be dedicated to other variants (other kinds of profit functions, multi-criteria case).

References

- [1] I-Ming Chao, Bruce L. Golden, and Edward A. Wasil. A fast and effective heuristic for the orienteering problem. *European Journal of Operational Research*, 88(3):475 – 489, 1996.
- [2] I-Ming Chao, Bruce L. Golden, and Edward A. Wasil. The team orienteering problem. *European Journal of Operational Research*, 88(3):464 – 474, 1996.
- [3] Frank H. Cullen, John J. Jarvis, and H. Donald Ratliff. Set partitioning based heuristics for interactive routing. *Networks*, 11(2):125–143, 1981.
- [4] J. Desrosiers, F. Soumis, and M. Desrochers. Routing with time windows by column generation. *Networks*, 14(4):545–565, 1984.
- [5] M. Dror and A. Langevin. Transformations and exact node routing solutions by column generation. In M. Dror, editor, *Arc routing: Theory, solutions and applications*, pages 277–326. Kluwer, 2000.
- [6] D. Feillet, P. Dejax, and M. Gendreau. Traveling salesman problems with profits. *Transportation Science*, 39(2):188 – 205, 2005.
- [7] D. Feillet, P. Dejax, M. Gendreau, and C. Guéguen. An exact algorithm for the elementary shortest path problem with resource constraints: Application to some vehicle routing problems. *Networks*, 44(3):216–229, 2004.
- [8] G. Gutirrez-Jarpa, G. Desaulniers, G. Laporte, and V. Marianov. A branch-and-price algorithm for the vehicle routing problem with deliveries, selective pickups and time windows. *European Journal of Operational Research*, 206(2):341 – 349, 2010.
- [9] C. Prins. A grasp x evolutionary local search hybrid for the vehicle routing problem. In F. B. Pereira and J. Tavares, editors, *Bio-inspired algorithms for the vehicle routing problem*, volume 161 of *Studies in Computational Intelligence*, pages 35 – 53. Springer, 2009.
- [10] T. Tsiligirides. Heuristic methods applied to orienteering. *The Journal of the Operational Research Society*, 35(9):p. 797–809, 1984.
- [11] P. Vansteenwegen, W. Souffriau, and D. Van Oudheusden. The orienteering problem: A survey. *European Journal of Operational Research*, 209(1):1 – 10, 2011.
- [12] S. Wolf and P. Merz. Evolutionary local search for the super-peer selection problem and the p-hub median problem. In *Hybrid Metaheuristics*, volume 4771 of *Lecture Notes in Computer Science*, pages 1–15. Springer Berlin Heidelberg, 2007.

A Network Approach to a Geometric Packing Problem

Bradley J. Paynter¹

*Department of Mathematics and Statistics
University of Central Oklahoma
Edmond, OK, U.S.A.*

Douglas R. Shier²

*Department of Mathematical Sciences
Clemson University
Clemson, SC, U.S.A.*

Abstract

We investigate a geometric packing problem (derived from an industrial setting) that involves fitting patterns of regularly spaced disks without overlap. We first derive conditions for achieving a feasible placement of a given set of patterns and then construct a network formulation that facilitates the calculation of such a placement. A heuristic utilizing this network representation is also outlined. Additionally, we show a connection to the well-known Periodic Scheduling Problem.

Keywords: Shortest Paths, Periodic Scheduling, Geometric Packing

¹ Email: bpaynter@uco.edu

² Email: shierd@clemson.edu

1 Introduction

We are concerned with the problem of fitting together objects that occur periodically. These objects can be intervals on the number line or shapes in the plane. This research was motivated by a situation arising from an industrial setting. Namely, a machine is used to drill holes in wheels to attach the wheels to vehicles with lug nuts. The holes are arranged in patterns. Each pattern has a specific number of holes, all with the same inner diameter, whose centers lie, evenly spaced, around the outer circumference of a larger circle. The factory uses templates to guide the machine in drilling patterns of holes for multiple wheel types. A template can contain several nonoverlapping patterns whose outer circles are concentric. The company wishes to design the fewest number of templates to accommodate a given set of patterns.

Example 1.1 Suppose we are given a set \mathcal{P} containing two patterns, the first (P_1) comprising three holes, each of radius 1, evenly spaced around a circle of radius 11.5, and the second (P_2) comprising four holes, each of radius 2, evenly spaced around a circle of radius 13. We can arrange these patterns without overlap on a single template by placing the “first” hole of P_2 immediately after (in a clockwise sense) the “first” hole of P_1 , as shown in Figure 1(a). In fact there is a range of relative displacements between these two patterns such that they can be feasibly arranged. Figure 1(b) shows a more complicated feasible arrangement of 14 patterns.

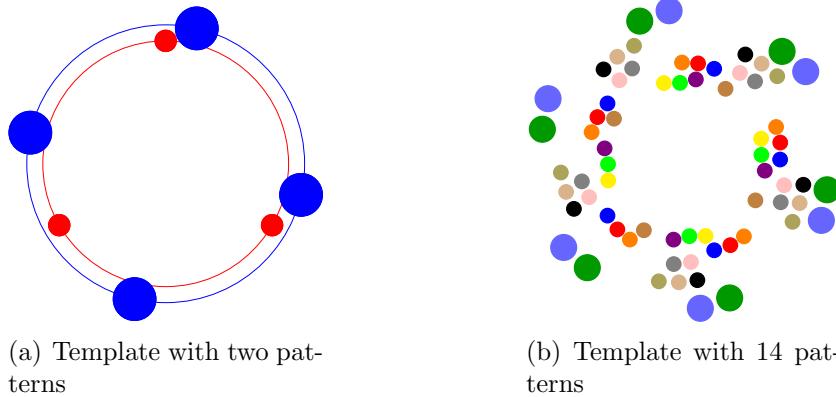


Fig. 1. Examples of feasible templates

More generally we define a *pattern* to be a set of open disks on the plane. Each pattern P consists of n disks of radius ρ , arranged so that their centers are evenly spaced around the circumference of a circle of radius R . The first

disk of the pattern will be indexed by zero and the angle of rotation of the center of that first disk will be denoted φ and measured in radians.

Definition 1.2 The pattern $P(\varphi) = \bigcup_{k \in \mathbb{Z}} P(\varphi, k)$ is a set of disks in the plane defined by the positive parameters R , n , ρ , and starting angle φ . Each disk $P(\varphi, k)$ is centered at $(R, \varphi + k2\pi/n)$, has radius ρ and forms the open set $P(\varphi, k) = \{(r, \theta) | r^2 - 2rR \cos(\theta - \varphi - k2\pi/n) + R^2 < \rho^2\}$. For simplicity, we may use the notation $P = (R, n, \rho)$.

Definition 1.3 A template is a set of patterns $\mathcal{P} = \{P_i | i \in I\}$ with fixed starting angles $\Phi = \{\varphi_i | i \in I\}$, denoted $\mathcal{T} = \mathcal{P}(\Phi) = \{P_i(\varphi_i) | i \in I\}$. Furthermore, if $P_i(\varphi_i) \cap P_j(\varphi_j) = \emptyset, \forall i \neq j$ then the template \mathcal{T} is a feasible template and the starting positions Φ are valid for \mathcal{P} .

A fundamental problem of interest in this paper is that of partitioning a set of patterns $\mathcal{P} = \{P_i | i \in I\}$ into the fewest number of feasible templates. To achieve this, we partition the index set I into a collection of disjoint sets $\mathcal{I} = \{I_u | u \in U\}$; accordingly the set of patterns \mathcal{P} is partitioned into sets $\mathcal{P}_u = \{P_i | i \in I_u\}$ so that the templates $\mathcal{T}_u = \{P_i(\varphi_i) | i \in I_u\}$ are feasible. This problem can be stated as the Minimum Templates Problem (MinTemp):

MinTemp(\mathcal{P}):

Given a set of patterns $\mathcal{P} = \{P_i | i \in I\}$, determine a partition of I , $\mathcal{I} = \{I_u | u \in U\}$, with minimum $|U|$, such that valid starting positions exist for each $\mathcal{P}_u = \{P_i | i \in I_u\}$.

It is easy to see a connection between this problem and the well-known Bin Packing Problem. However, whereas it is relatively simple to determine whether or not an item fits in a bin, it is more complicated to determine whether a set of patterns can form a feasible template. As a result, we focus now on this subproblem.

2 Fit Conditions

It is useful to first define $\beta_{ij} = 2\pi / \text{lcm}(n_i, n_j)$ and

$$\delta_{ij} = \begin{cases} 2 \arcsin \left(\sqrt{\frac{(\rho_i + \rho_j)^2 - (R_i - R_j)^2}{4R_i R_j}} \right), & |R_i - R_j| < \rho_i + \rho_j \\ 0, & |R_i - R_j| \geq \rho_i + \rho_j. \end{cases}$$

Theorem 2.1 [5] Suppose $\mathcal{P} = \{P_i \mid i \in I\}$ where $P_i = (R_i, n_i, \rho_i)$, and let \preceq be a total order on I . Then $\{P_i(\varphi_i) \mid i \in I\}$ is a feasible template iff there exist $k_{ij} \in \mathbb{Z}$, $i \prec j$, such that

$$k_{ij}\beta_{ij} + \delta_{ij} \leq \varphi_j - \varphi_i \leq (k_{ij} + 1)\beta_{ij} - \delta_{ij}, \quad \forall i \prec j. \quad (1)$$

Given a set of patterns $\mathcal{P} = \{P_i \mid i \in I\}$, inequality (1) for patterns P_i and P_j , $i \prec j$ can be rewritten as

$$\begin{cases} \varphi_j - \varphi_i \leq (k_{ij} + 1)\beta_{ij} - \delta_{ij} \\ \varphi_i - \varphi_j \leq -k_{ij}\beta_{ij} - \delta_{ij}. \end{cases}$$

Thus, if we define for $i \neq j$

$$w_{ij} = \begin{cases} (k_{ij} + 1)\beta_{ij} - \delta_{ij}, & i \prec j \\ -k_{ji}\beta_{ji} - \delta_{ji}, & j \prec i \end{cases} \quad (2)$$

then the above inequalities become (for $i \neq j$) $\varphi_j - \varphi_i \leq w_{ij}$. These, however, are simply the optimality conditions for the single-source shortest path problem [1] in an appropriately defined network with arc lengths (weights) w_{ij} . Additionally, if we specify a root node r (i.e., $\varphi_r = 0$), then the shortest path distances from the root node will be valid starting positions for the set \mathcal{P} . Consequently, we are led to the construction of the following network.

Definition 2.2 Let $\mathcal{P} = \{P_i \mid i \in I\}$ be a set of patterns $P_i = (R_i, n_i, \rho_i)$, let \preceq be a total order on I , and let $K = \{k_{ij} \in \mathbb{Z} \mid i \prec j\}$. Define network $G(\mathcal{P}, \preceq, K) = (N, A, w)$ with $N = I$, $A = \{(i, j) \mid i \neq j \in I\}$, and w as in (2).

Example 2.3 Suppose the set \mathcal{P} contains patterns $P_1 = (12, 3, 1)$, $P_2 = (11.5, 4, 1)$, $P_3 = (12.5, 5, 1)$. If we use the ordering $1 \prec 2 \prec 3$ and propose $K = \{k_{12} = 0, k_{13} = 1, k_{23} = 1\}$ then $G(\mathcal{P}, \preceq, K)$ is shown in Figure 2(a). Finding shortest paths from node 1 gives the starting positions $(0, 0.221, 0.680) = (\varphi_1, \varphi_2, \varphi_3)$. This solution is shown in Figure 2(b).

If we can find shortest paths in network $G(\mathcal{P}, \preceq, K)$, then the corresponding shortest path distances provide valid starting positions for \mathcal{P} . However, shortest path distances exist in a network precisely when the network contains no negative weight cycles. Thus, if we can eliminate negative weight cycles from $G(\mathcal{P}, \preceq, K)$ by adjusting the k_{ij} values, then valid starting positions will be obtained. Such a heuristic (see [5]) is briefly described in the next section.

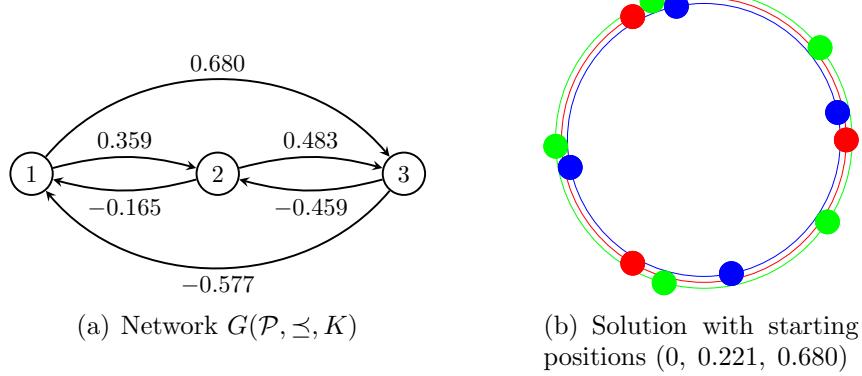


Fig. 2. Network solution for Example 2.3

3 Cycle Heuristic

Since MinTemp is fundamentally a Bin Packing Problem, well-known bin packing heuristics [2] can be applied here. However, additional routines need to be developed to determine whether a pattern can be fit in an existing feasible template (or bin). To achieve this, we exploit the fact that the current feasible template $\mathcal{T} = \mathcal{P}(\Phi)$ has a total order \preceq and a set K such that $G(\mathcal{P}, \preceq, K)$ has no negative cycles. We then expand $(\mathcal{P}, \preceq, K)$ to include a new pattern P_t forming $(\mathcal{P}', \preceq', K')$ where t is last with respect to \preceq' . Since the original $G(\mathcal{P}, \preceq, K)$ contains no negative cycles, then any negative cycle in $G(\mathcal{P}', \preceq', K')$ must pass through the node t corresponding to the new pattern P_t .

The heuristic proceeds by first finding a negative cycle C in $G(\mathcal{P}', \preceq', K')$ using a standard shortest path algorithm [1]. If no such cycle exists then the new, expanded template is feasible. Otherwise, we identify the arcs on the negative cycle incident with the new node t , namely (i, t) and (t, j) which have weights $w_{it} = (k_{it} + 1)\beta_{it} - \delta_{it}$ and $w_{jt} = -k_{jt}\beta_{jt} - \delta_{jt}$. We then either add $[-w[C]/\beta_{ij}]$ to k_{it} or subtract the same amount from k_{jt} to make the cycle non-negative. By restricting all changes in the set K' to items in $K' \setminus K$, we maintain the property that $G(\mathcal{P}, \preceq, K)$ has no negative cycles. It should be noted that adjusting k_{it} or k_{jt} to eliminate a negative cycle may in fact cause another cycle to become negative. As a result this process is repeated until no negative cycles remain, or some termination criteria are met. Care is also taken to prevent the heuristic from cycling.

This heuristic was applied to the original industrial data set which contained 63 patterns with varying parameters. A solution with five templates

was obtained, one of which is shown in Figure 1(b). A commercial integer program solver, running on the same hardware, found a comparable solution but consumed approximately 2500 times the amount of processing time. Additionally, the commercial solver was unable to improve on the heuristic solution or prove its optimality in any reasonable amount of time.

In addition, the heuristic was tested on a collection of randomly generated sets of patterns. Table 1 summarizes the results and provides a comparison to the commercial solver³. Although the commercial solver can produce marginally better results (average number of templates used), it takes significantly longer to do so and increasingly is unable to solve instances within the given 1 hour time limit.

Patterns per set	25	50	75	100	125	150	175
Number of sets	10	10	10	10	10	10	10
Average heuristic solution	6.4	9.6	13.4	16.8	19.7	23.5	26.7
Average heuristic time (s)	0.02	0.01	0.02	0.03	0.05	0.05	0.03
# Solver feasible solutions	8	9	7	5	2	1	0
Average solver solution	6.6	8.6	12.3	14.8	19	26	-
Average solver time (s)	2.55	1549	3266	3604	3638	3663	3660

Table 1
Summary of heuristic performance

4 Linearization

We now briefly study a simplified version of the two-dimensional problem in which the objects to be packed are now intervals on the real line. In the process, we make a connection to the problem of scheduling periodic events.

The structure of a pattern in \mathbb{R}^2 can be approximated by unfolding its outer circumference along the number line and duplicating the pattern infinitely in both directions. The disks of the pattern now become intervals of the outer circumference. The pattern can now be defined by the length of the outer circumference B , the number of intervals n , and the length of each interval d .

³ All computations were performed on a personal computer with dual AMD Opteron 4284 3.0GHz 8 core CPUs with 64GiB of RAM. The commercial solver used was Gurobi 5.0.1.

The rotation of the pattern now becomes a shift s , which measures the distance from a designated point on the outer circumference to the left endpoint of the first interval. Templates and feasibility are defined as before.

Definition 4.1 The linear pattern $\tilde{P}(s) = \bigcup_{k \in \mathbb{Z}} (s + kB/n, s + kB/n + d)$ is an infinite set of open intervals defined by the strictly positive constants $B, d \in \mathbb{R}$ and $n \in \mathbb{Z}$. For simplicity, we use the notation $\tilde{P} = (B, n, d)$.

We now examine the special case in which all patterns share the same outer circumference B . Fitting all such patterns on the fewest templates is denoted the Linearized Minimum Templates Problem (LinMinTemp).

In the Periodic Scheduling Problem (PSP) [3] we are given a set T of n periodic tasks, where each task $i \in T$ has period $p(i) \in \mathbb{N}$ and execution time $e(i) \in \mathbb{N}$ with $e(i) \leq p(i)$. Also given are a set of identical processors. Tasks are assigned to processors and a starting time $s(i)$ is given for each task i with the requirement that no processor can run two tasks at the same time. The problem is to find such an assignment that uses the fewest processors. Note that the periodic tasks from the PSP correspond to the linear patterns of Definition 4.1 with $p(i) = B/n_i$ and $e(i) = d_i$. The processors of the PSP are analogous to templates in LinMinTemp and a task's starting time $s(i)$ corresponds to a pattern's starting position s_i . Also $\gcd(p(i), p(j)) = \gcd(B/n_i, B/n_j) = B/\text{lcm}(n_i, n_j) \equiv B_{ij}$. Thus, LinMinTemp is equivalent to PSP and this allows us to state the following theorem.

Theorem 4.2 (Korst, Aarts, Lenstra & Wessels [4])

Given a set of linear patterns $\tilde{\mathcal{P}} = \{\tilde{P}_i \mid i \in I\}$ where $\tilde{P}_i = (B, n_i, d_i)$, then $\{\tilde{P}_i(s_i) \mid i \in I\}$ is a feasible template iff there exist $k_{ij} \in \mathbb{Z}$ such that

$$k_{ij}B_{ij} + d_i \leq s_j - s_i \leq (k_{ij} + 1)B_{ij} - d_j, \quad \forall i \prec j. \quad (3)$$

The similarities between (3) and (1) result from the fact that linear patterns approximate general patterns with $d_i \approx 2\rho_i$, $B = 2\pi R$, $s_i = R\varphi_i$, and

$$\delta_{ij} \approx 2\sqrt{\frac{(\rho_i + \rho_j)^2 - (R_i - R_j)^2}{4R_i R_j}}.$$

Using this approximation our previous network formulation applies with B_{ij} and d_j replacing β_{ij} and δ_{ij} respectively in the definition of w_{ij} . Moreover, the heuristics developed for MinTemp can therefore be applied to LinMinTemp as well. Computational results [5] show that such heuristics can be effective in solving the linearized problem.

This linearized formulation provides some advantages over the generalized formulation. For example, the parameters of (3) are much easier to calculate than those in (1) and are less prone to numerical roundoff errors. However, the linearized formulation does lose the ability to have patterns with differing outer circumferences. It should be noted that, as the cycle heuristic is essentially a heuristic for eliminating negative cycles from a network, it can be applied directly to linearized problems.

5 Conclusion

In this paper we examined a geometric packing problem (motivated by an industrial situation) and derived feasibility conditions for the problem. We then converted these feasibility constraints into a network problem. By eliminating negative cycles from this parameterized network and finding shortest paths from a specified root node, we obtain feasible packings of the patterns.

We then considered an approximation to the geometric packing problem and transferred most of the previous results, including the network formulation, to this approximation. We further showed that this approximate packing problem is equivalent to the Periodic Scheduling Problem, so that heuristics developed for the former can be applied to the latter.

References

- [1] Ahuja, R., T. Magnanti and J. Orlin, “Network Flows: Theory, Algorithms, and Applications,” Prentice-Hall, Upper Saddle River, NJ, 1993.
- [2] Coffman, E. G., Jr., M. R. Garey and D. S. Johnson, *Approximation algorithms for bin packing: A survey*, in: D. Hochbaum, editor, *Approximation Algorithms for NP-hard Problems*, PWS Publishing Co., Boston, MA, USA, 1997 pp. 46–93.
- [3] Korst, J., E. Aarts and J. K. Lenstra, *Scheduling periodic tasks*, INFORMS Journal on Computing **8** (1996), pp. 428–435.
- [4] Korst, J., E. Aarts, J. K. Lenstra and J. Wessels, *Periodic multiprocessor scheduling*, in: *Proceedings of the Conference on Parallel Architectures and Languages Europe, PARLE ’91*, Lecture Notes in Computer Science **505**, Springer, Berlin, 1991 pp. 166–178.
- [5] Paynter, B., “An Optimization Approach to a Geometric Packing Problem,” PhD dissertation, Department of Mathematical Sciences, Clemson University, Clemson, SC (2012).

The cut condition for robust network design

Sara Mattia¹

*Istituto di Analisi dei Sistemi ed Informatica (IASI)
Consiglio Nazionale delle Ricerche (CNR)
Rome, Italy*

Abstract

We consider the Robust Network Design problem, i.e. the problem of dimensioning the capacities on the edges of a graph to serve point-to-point traffic demands subject to uncertainty. The scope of the paper is to investigate when the cut condition is sufficient to guarantee the feasibility for such problem.

Keywords: Robust Network Design, Cut Inequalities

1 Introduction

Let $G(V, E)$ be an undirected graph with capacity installation costs for the edges. Let \mathcal{D} be the set of traffic matrices and let $D \in \mathcal{D}$ be a feasible traffic matrix to be served, where entry d_{ij} is the amount to be routed from source node i to destination node j . Let K be the set of commodities, where each commodity is a triple (s_k, t_k, d_k) , where (s_k, t_k) is a node pair with $s_k \neq t_k$ and $d_k = d_{s_k t_k}$ is the (uncertain) amount to be sent from s_k to t_k . The Robust Network Design (RND) problem consists of finding minimum cost edge capacities such that all $D \in \mathcal{D}$ can be routed non simultaneously on the network. The capacities are not restricted to be integer numbers. If each commodity

¹ Email: sara.mattia@iasi.cnr.it

must use a single path, then the flows are *unsplittable* (or *non-bifurcated*). If the flow for a commodity can be splitted along different paths, the flows are *splittable* (or *bifurcated*). The routing scheme is called *dynamic* if it depends on the traffic matrix (each traffic matrix can be routed in a different way), it is called *static* (or *oblivious*) if the routing is independent of the matrix (it must be the same for all the matrices). Unlike what happens for static RND, in general dynamic RND cannot be formulated in a compact way and it is not easy to solve [8,14]. The separation version of the problem is solved by bilevel and mixed integer programs [20].

For approximation and complexity results for the problem see [8,13,14]. An exact algorithm for single commodity RND is given in [7]. In [12] a special case of the problem where the dominant extreme points of the demand polyhedron have a disjoint support is studied. For the RND problem with integer capacities and a single traffic matrix (Network Loading, NL) see [4,6,22,27]. For the problem with integer capacities and demand uncertainty (Robust Network Loading, RNL) see [2,3,17,20]. Additional references can be found in the above mentioned papers.

Different models to represent the demand uncertainty have been investigated in the literature. In this paper we focus our attention on two uncertainty models: the hose model [9,10] and the cardinality constrained model [5]. In the hose model, upper bounds on the traffic generated by each node are set and \mathcal{D} includes all the matrices respecting the bounds. The polyhedron is called *asymmetric* if, for every node i , we have one bound for the incoming traffic $b_i^{in} \geq 0$ and another one for the outgoing traffic $b_i^{out} \geq 0$. If we have a single bound $b_i \geq 0$ on the sum of the incoming and outgoing traffic it is called *symmetric*. We denote by $D(A)$ (resp. $D(S)$) the asymmetric (resp. symmetric) hose polyhedron. In the cardinality constrained model each demand d_{ij} is supposed to be an independent symmetric and bounded random variable with nominal value $\bar{d}_{ij} \geq 0$ taking values in $[\bar{d}_{ij} - \delta_{ij}, \bar{d}_{ij} + \delta_{ij}]$, where $0 \leq \delta_{ij} \leq \bar{d}_{ij}$ for all $i, j \in V$ [1]. At most Γ demands can deviate from the nominal value at the same time (for the sake of simplicity we assume that Γ is integer). We denote by \bar{D} the traffic matrix where each demand is at its nominal value and by $D(\Gamma)$ the set of feasible traffic matrices when the cardinality uncertainty model is used (Γ polyhedron). The results we present for the Γ polyhedron also hold for the multiple interval model [21].

Let $\{S : V - S\}$ be a partition of the nodes (cut), let $\delta(S)$ be the set of edges

having endpoints in different sets of the partition and let $K(S)$ be the set of commodities having sources and destinations separated by the cut. Cut inequalities (1) provide a necessary condition for a solution x to be feasible.

$$\sum_{e \in \delta(S)} x_e \geq \sum_{k \in K(S)} d_k \quad S \subseteq V, D \in \mathcal{D} \quad (1)$$

It is well-known that, in general, even when $|\mathcal{D}| = 1$ (single traffic matrix), the cut condition (1) is not sufficient to ensure the feasibility and more general conditions (*metric inequalities* [16,25]) must be satisfied. However, for the problem with a single traffic matrix, there exist several cases in which the cut condition is indeed also sufficient [15,18,23,24,26,28,30,31]. See [29] for additional references. The scope of this paper is to investigate what happens in (some of) such cases to the robust problem with dynamic routing (Section 2). Comments on the problem with static routing (Section 2.1) and on what happens if the graph and the demands are directed (Section 2.2) are also made. We also note that, when cut inequalities are enough for feasibility for RND, then cut inequalities plus integrality requirements are a formulation for RNL.

2 The results

The proofs of the results in this section can be found in [19]. We refer to condition (1) for a single traffic matrix as the *single matrix cut condition*, and for $|\mathcal{D}| > 1$ as the *robust cut condition*. Given a traffic matrix D , it is possible to build a graph $H_D(T, R)$ having an edge for every commodity and $T \subseteq V$. G is the *supply graph* and H_D is the *demand graph*. A graph is called *planar* if it can be drawn in such a way that no edges cross each other. The regions bounded by the edges are called *faces*, including the outer (unbounded) one.

For the problem with a single traffic matrix, the single matrix cut condition is sufficient for feasibility when:

- (i) G is a tree;
- (ii) all the commodities share the same source or the same destination (this includes the single commodity case [11]);
- (iii) we have only two commodities [15];
- (iv) $H_D = K_4$ (complete graph on four nodes) or $H_D = C_5$ (cycle of five nodes) or H_D is the union of two stars (all the nodes but one connected only to the remaining node) [18,26,30];

- (v) $G + H_D$ is planar [31];
- (vi) G is planar and s_k, t_k are on the boundary of the infinite face for every $k \in K$ [24];
- (vii) G is planar and there exists a node $r \in V$ such that for every $k \in K$ either s_k and t_k are on the outer boundary or $r \in \{s_k, t_k\}$ [23];
- (viii) G is planar and has two bounded faces F_1 and F_2 such that $s_1, \dots, s_{|K|}$ and $t_1, \dots, t_{|K|}$ occur clockwise around the boundary of F_1 and F_2 respectively [28].

Condition **i** is independent of the demand graph, conditions **ii-iv** are independent of the supply graph, whereas the others rely on the structure of both the supply and the demand graph. It is easy to see that when condition **i** holds, cut inequalities are enough to guarantee the feasibility also for the robust problem, independently of the definition of the uncertainty set \mathcal{D} . Our aim it is provide conditions under which the pair (G, H_D) respects one of the conditions above for all $D \in \mathcal{D}$. If so, robust cut inequalities are enough for feasibility for dynamic RND due to the following theorem.

Theorem 2.1 *If, for all $D \in \mathcal{D}$, the single matrix cut condition is enough for feasibility, then the robust cut condition is enough for feasibility for dynamic RND.*

Let us consider now dynamic RND under hose uncertainty. Conditions **iii** and **v** can be used as below.

Lemma 2.2 *Let $D = D(A)$ (resp. $D(S)$). Let $I = \{i \in V : b_i^{in} + b_i^{out} > 0\}$ (resp. $I = \{i \in V : b_i > 0\}$). If $|I| \leq 2$, then the robust cut condition is sufficient for feasibility for dynamic RND.*

Lemma 2.3 *Let $\mathcal{D} = D(A)$ (resp. $D(S)$). Let H be the graph obtained connecting any source to any destination. If $G + H$ is planar, then the robust cut condition is sufficient for feasibility for dynamic RND.*

For $D(A)$, conditions **ii** and **viii** can be generalized as follows.

Lemma 2.4 *Let $\mathcal{D} = D(A)$. If either there is a unique node i having $b_i^{in} > 0$, or there exists a unique node i having $b_i^{out} > 0$, then the robust cut condition is sufficient for feasibility for RND under dynamic routing.*

Lemma 2.5 *Let $\mathcal{D} = D(A)$. If G is planar and has two bounded faces F_1 and F_2 such that nodes s_1, \dots, s_p with positive outgoing bounds and all the nodes t_1, \dots, t_q with positive incoming bounds occur clockwise around the boundary of*

F_1 and F_2 respectively, then the robust cut condition is sufficient for feasibility for dynamic RND.

For a generalization of conditions [vi](#) and [vii](#) to dynamic RND with $D(A)$ see [\[19\]](#). When the Γ polyhedron is used, all the above mentioned conditions for the single matrix case can be generalized to dynamic RND, independently of Γ , i.e. on the number of demands that are allowed to change with respect to the nominal value. In fact, the assumption $\delta_{ij} \leq \bar{d}_{ij}$ made on $D(\Gamma)$ implies that, when $\bar{d}_{ij} = 0$, the corresponding demand does not change. Hence, the amount of each demand can vary, but since the amount does not affect the structure of the demand graph (only the origin-destination pairs matter) and no new pairs can have a positive demand, H_D has the same structure of $H_{\bar{D}}$, for every $D \in \mathcal{D}$.

Lemma 2.6 *Let $\mathcal{D} = D(\Gamma)$. If the pair $(G, H_{\bar{D}})$ satisfies one of conditions [i-viii](#), then the robust cut condition is also sufficient for feasibility for RND under dynamic routing.*

2.1 Static routing

It is easy to see that if G is a tree, then the cut condition is sufficient also for the static case. The same happens if $\mathcal{D} = D(\Gamma)$ and \bar{D} includes only one commodity (s, t, d_{st}) . The rest of the conditions above cannot be generalized to static RND as done for dynamic RND. In fact, for dynamic RND the existence of a feasible routing for each matrix implies the existence of a routing for the robust problem. It is not so for static RND, because we also have to ensure that the routing is the same for all the matrices.

An example showing that the robust cut condition is not sufficient for feasibility for $\mathcal{D} = D(S)$ can be found in [\[20\]](#). As for the Γ polyhedron, even for a two commodity problem when the two commodities share the same source, the robust cut condition does not imply the feasibility for static RND. Let G be the complete undirected graph on three nodes and let the nominal value demands be $\bar{d}_{12} = \bar{d}_{13} = 1$, $\bar{d}_{23} = 0$ and suppose that each demand can vary by at most $\delta_{ij} = \bar{d}_{ij}$ and that $\Gamma = 1$. Vector $x_{12} = 2$, $x_{13} = x_{23} = 1$ satisfies the robust cut condition but it does not allow a static routing. Let the demands be ordered lexicographically and consider realizations $D_A = [1, 2, 0]$ and $D_B = [2, 1, 0]$. There is only one feasible routing for D_A : d_{12} routed 100% on path 1-2, d_{13} routed 50% on 1-3 and 50% on 1-2-3. This routing is not feasible for D_B (capacity on (1, 2) is exceeded), hence no static routing exists.

2.2 Directed graphs

For directed graph, very few conditions in which cuts are sufficient are known for the single matrix problem. If G is a directed tree (a directed acyclic graph that remains acyclic when the orientation of the edges is not considered), then either no feasible solution exists or cut inequalities are sufficient for feasibility. The single matrix cut condition is sufficient for feasibility independently of the supply graph, if and only if all the demands have a common source or a common destination (this includes the single commodity case) [29]. The first condition can be easily generalized to the robust problem, independently of \mathcal{D} . As for the second condition, for the hose polyhedron the following holds.

Lemma 2.7 *Let $\mathcal{D} = D(A)$. If either there is a unique node i having $b_i^{in} > 0$, or there exists a unique node i having $b_i^{out} > 0$, then the robust cut condition is sufficient for feasibility for RND under dynamic routing.*

For the Γ polyhedron the following holds.

Lemma 2.8 *Let $\mathcal{D} = D(\Gamma)$. If the demands in \bar{D} have a unique source or a unique destination, then the robust cut condition is also sufficient for feasibility for RND under dynamic routing.*

3 Conclusions

We investigated when cut inequalities are sufficient for feasibility for dynamic RND. The conditions for the single matrix problem can still be used when the Γ polyhedron is considered, while additional assumptions may be needed if the uncertainty is modeled using the hose polyhedron. When static routing is used, very few results can be proved, because the feasibility of each single matrix problem does not imply the feasibility of the robust problem, if it is not proved that the routing is the same for all the matrices.

References

- [1] A. Altin, E. Amaldi, P. Belotti, and M.Ç. Pinar. Provisioning virtual private networks under traffic uncertainty. *Networks*, 49(1):100–115, 2007.
- [2] A. Altin, H. Yaman, and M.Ç. Pinar. The robust network loading problem under hose demand uncertainty: formulation, polyhedral analysis, and computations. *INFORMS J. Computing*, 23(1):75–89, 2010.

- [3] A. Atamturk and M. Zhang. Two-stage robust network flow and design under demand uncertainty. *Oper. Res.*, 55:662–673, 2007.
- [4] P. Avella, S. Mattia, and A. Sassano. Metric inequalities and the network loading problem. *Disc. Opt.*, 4:103–114, 2007.
- [5] D. Bertsimas and M. Sim. The price of robustness. *Oper. Res.*, 52(1):35–53, 2004.
- [6] D. Bienstock, S. Chopra, O. Günlük, and C.Y. Tsai. Minimum cost capacity installation for multicommodity flows. *Math. Prog.*, 81:177–199, 1998.
- [7] C. Buchheim, F. Liers, and L. Sanità. An exact algorithm for robust network design. In *Proc. of INOC 2011, LNCS 6701*, pages 7–11, 2011.
- [8] C. Chekuri, G. Oriolo, M.G. Scutellà, and F.B. Shepherd. Hardness of robust network design. *Networks*, 50(1):50–154, 2007.
- [9] N.G. Duffield, P. Goyal, A. Greenberg, P. Mishra, K.K. Ramakrishnan, and J.E. van der Merwe. A flexible model for resource management in virtual private networks. In *Proc. of the ACM SIGCOMM Computer Communication Review*, volume 29, pages 95–109, 1999.
- [10] J.A. Fingerhut, S. Suri, and J. Turner. Designing least-cost nonblocking broadband networks. *J. Algorithms*, 24:287–309, 1997.
- [11] L.R. Ford and D.R. Fulkerson. Maximal flow through a network. *Can. J. Math.*, 8:399–404, 1956.
- [12] A. Frangioni, F. Pascali, and M.G. Scutellà. Static and dynamic routing under disjoint dominant extreme demands. *Oper. Res. Lett.*, 39(1), 2011.
- [13] N. Goyal, N. Olver, and B. Shepherd. Dynamic vs oblivious routing in network design. In *17th Annual European Symposium on Algorithms (ESA 2009)*, 2009.
- [14] A. Gupta, J. Kleinberg, A. Kumar, R. Rastogi, and B. Yener. Provisioning a virtual private network: a network design problem for multicommodity flows. In *Proc. of ACMSTOC 2001*, pages 389–398, 2001.
- [15] T.C. Hu. Multi-commodity network flows. *Oper. Res.*, 11:344–360, 1963.
- [16] M. Iri. On an extension of the max-flow min-cut theorem to multicommodity flows. *Journal of the Operations Research Society of Japan*, 13:129–135, 1971.
- [17] A. Koster, M. Kutschka, and C. Raack. Robust network design: Formulations, valid inequalities, and computations. *Networks*, 2013. doi 10.1002/net.21497.

- [18] M. Lomonosov. Combinatorial approaches to multiflow problems. *Disc. Appl. Math.*, 11:1–93, 1985.
- [19] S. Mattia. The cut condition for robust network design. Technical Report 12-23, IASI-CNR, 2012.
- [20] S. Mattia. The robust network loading problem with dynamic routing. *Comput. Optim. Appl.*, 2012. doi 10.1007/s10589-012-9500-0.
- [21] S. Mattia. Robust optimization with multiple intervals. Technical Report 12-07, IASI-CNR, 2012.
- [22] S. Mattia. Separating tight metric inequalities by bilevel programming. *Oper. Res. Lett.*, 40:568–572, 2012.
- [23] H. Okamura. Multicommodity flows in graphs. *Disc. Appl. Math.*, 61:55–62, 1983.
- [24] H. Okamura and P.D. Seymour. Multicommodity flows in planar graphs. *J. Comb. Theory Ser. B*, 31:71–81, 1981.
- [25] K. Onaga and O. Kakusho. On feasibility conditions of multicommodity flows in network. *IEEE Trans. Circuit Theory*, 18(4):425–429, 1971.
- [26] B.A. Papernov. Feasibility of multicommodity flows. In A. Friedman, editor, *Studies in Discrete Optimization*, pages 230–261. 1976. (in Russian).
- [27] C. Raack, A. Koster, S. Orlowski, and R. Wessäly. On cut-based inequalities for capacitated network design polyhedra. *Networks*, 57(2):141–156, 2011.
- [28] A. Schrijver. The klein bottle and multicommodityflows. *Combinatorica*, 9:375–384, 1989.
- [29] A. Schrijver. *Combinatorial Optimization. Polyhedra and Efficiency*. Springer, 2003.
- [30] P.D. Seymour. Four-terminus flows. *Networks*, 10:79–86, 1980.
- [31] P.D. Seymour. Matroids and multicommodity flows. *Eur. J. Comb.*, 2:257–290, 1981.

A new model for multicommodity flow problems, and a strongly polynomial algorithm for single-source Maximum Concurrent Flow

Bauguion Pierre-Olivier¹

*Orange Labs
38 rue du général Leclerc 92130 Issy-Les-Moulineaux France*

Ben-Ameur Walid²

*Samovar, CNRS UMR 5157
Télécom SudParis
Evry France*

Gourdin Eric³

*Orange Labs
38 rue du général Leclerc 92130 Issy-Les-Moulineaux France*

Abstract

In this paper, a new decomposition approach is proposed to solve large size instances of Multicommodity Flow problems. Instead of generating paths, we generate trees in a convenient way. Numerical results show that the new approach is much more efficient than the classical paths generation approach. Moreover, we propose a combinatorial polynomial-time algorithm to solve the maximum concurrent flow problem (MCF) in the single-source case.

Keywords: Maximum concurrent flow, column generation, tree packing

1 Multicommodity flow problems

Consider a capacitated directed graph $G = (V, A)$ where V ($|V| = n$) is the set of nodes or vertices and A ($|A| = m$) is the set of arcs. The maximum amount of flow that each arc $a \in A$ can accommodate is called capacity and denoted by $c_a \geq 0$. A commodity k is defined by a source node $s^k \in V$, a destination node $t^k \in V$ and an amount of flow $d^k > 0$ (also called demand) to carry from s^k to t^k . We denote by K the set of all commodities. Let P^k be the set of all simple paths (without cycles) between s^k and t^k . Each commodity k can hence be routed over a set of paths in P^k . For a commodity defined between the nodes i and j , we will sometimes denote by d^{ij} the demand and P^{ij} the available path between the two nodes s and t , and its $s-t$ flow, which can be defined as a vector $(f_a^{st})_{a \in A}$ such that:

$$(1) \quad f \{ \quad {}^{st}(\delta^-(v)) = f^{st}(\delta^+(v)), \text{ for all } v \in V \setminus \{s, t\}.$$

where $\delta^-(v)$ (resp. $\delta^+(v)$) denotes the set of incoming (resp. outgoing) arcs at node v . Constraint (1) is the so-called "flow conservation" constraint and f_a^{st} represents the amount of flow on arc a . An $s-t$ flow is feasible if it is non-negative and it satisfies all capacity constraints: $0 \leq f_a^{st} \leq c_a$, $a \in A$. It is well known that an $s-t$ flow can be equivalently described by flows on paths between s and t . More precisely, a flow between s and t is equivalent to a set of flows on paths, plus flows on circuits. However, for typical objective functions, there are optimal solutions without flow on circuits. This is indeed the case in our paper.

There are many problems, generally called *Multicommodity Flow* problems, where a set of commodities has to be routed in the graph G , such that the resulting flow is feasible. The model we propose in this paper can be applied to any Multicommodity Flow problem. However, to illustrate its efficiency, we consider mainly one of the "hardest" of these problems, the *Maximum Concurrent Flow* problem (MCF). Approximation algorithms have been proposed for this problem (see, .e.g., [4]). Multicommodity Flow problems are generally solved using column generation. Many techniques are known to improve the efficiency of column generation (see, .e.g., [1]).

We first recall one of the standard models for this problem, using path variables λ_p^k representing, for each commodity k , the proportion devoted to

¹ Email:pierre.bauguion@orange.com

² Email: walid.benameur@cit-sudparis.eu

³ Email: eric.gourdin@orange.com

each path $p \in P^k$ and denoting by γ_{path} the objective function:

$$(2) \quad \max \gamma_{path},$$

$$(3) \quad (MCF_{path}) \left\{ \begin{array}{ll} \sum_{k \in K} \sum_{p \in P^k, p \ni a} \lambda_p^k d^k \leq c_a, & \forall a \in A, \\ \sum_{p \in P^k} \lambda_p^k \geq \gamma_{path}, & \forall k \in K, \\ \lambda_p^k \geq 0, & \forall k \in K, p \in P^k. \end{array} \right.$$

$$(4)$$

$$(5)$$

2 The tree-flow formulation for Maximum Concurrent Flow

In the following text, we use the word tree even if it should be properly called arborescence. Let $S = \bigcup_{k \in K} \{s^k\}$ be the subset of V consisting of commodity source nodes. Given a source node $i \in S$, we denote by $T^i = \bigcup_{k \in K, s^k=i} \{t^k\}$ the set of nodes that are destinations for commodities having their source in i , and by \mathcal{T}^i the set of trees rooted at i and spanning T^i . Given a rooted tree $\tau \in \mathcal{T}^i$ and an arc $a \in A$, we denote by V_a^τ the set of vertices such that the unique path, in τ , connecting i to t^k , for some $k \in T^i$, contains the arc a : $V_a^\tau = \{j \in T^i, \exists p \in P^{ij} \text{ such that } p \subset \tau, p \ni a\}$. In the model we propose, a common variable λ_τ^i is assigned to each tree $\tau \in \mathcal{T}^i$ and, for each commodity $k \in K$ having its source in i ($s^k = i$), $\lambda_\tau^i d^k$ units of flows are sent to t^k along the unique path in the tree τ :

$$(6) \quad \max \gamma_{tree},$$

$$(7) \quad (MCF_{tree}) \left\{ \begin{array}{ll} \sum_{i \in S} \sum_{\tau \in \mathcal{T}^i: \tau \ni a} \sum_{j \in V_a^\tau} \lambda_\tau^i d^{ij} \leq c_a & \forall a \in A, \\ \sum_{\tau \in \mathcal{T}^i} \lambda_\tau^i \geq \gamma_{tree}, & \forall i \in S, \\ \lambda_\tau^i \geq 0, & \forall i \in S, \tau \in \mathcal{T}^i. \end{array} \right.$$

$$(8)$$

$$(9)$$

In constraints (7), the left-hand-side sums over all trees containing the arc a , the traffic from the root node (common source of some commodities) towards the destinations of these commodities, which hence represents the total traffic on arc a . In constraints (8), we require that the sum of λ_τ^i over the trees having a common source i should be at least γ_{tree} . Denoting by γ^* the value of the optimal solution (which exists and is finite) we prove that the path and the

tree formulations are equivalent.

Theorem 2.1 $\gamma_{tree}^* = \gamma_{path}^*$.

Proof. Consider the dual of MCF_{tree} :

$$(10) \quad \min \sum_{a \in A} c_a u_a,$$

$$(11) \quad \gamma_{tree}^* = \begin{cases} \sum_{s \in S} v^s \geq 1, \\ \sum_{t \in T^s} d^{st} \text{len}_{\tau}^{st}(u) \geq v^s \quad \forall s \in S, \tau \in \mathcal{T}^s, \\ u_a, v^s \geq 0, \quad \forall a \in A, s \in S, \end{cases}$$

where $\text{len}_{\tau}^{st}(u)$ denotes the length of the path between s and t , in the tree τ , with weights u_a on the arcs. Note that $(u_a)_{a \in A}$ and $(v^s)_{s \in S}$ are the sets of dual variables associated respectively with the constraints (7) and (8). From (12), we see that:

$$v^s \leq \min_{\tau \in \mathcal{T}^s} \sum_{t \in T^s} d^{st} \text{len}_{\tau}^{st}(u), \quad \forall s \in S.$$

Since the sum of minimum length is achieved along the shortest-path tree, we have:

$$(14) \quad v^s \leq \sum_{t \in T^s} \min_{p \in P^{st}} d^{st} \text{len}_p^{st}(u), \quad \forall s \in S,$$

where $\text{len}_p^{st}(u)$ is the length of the path p with weights u_a . Letting $v^{st}(= v^k) = \min_{p \in P^{st}} d^{st} \text{len}_p^{st}(u)$, we have:

$$(15) \quad v^k \leq d^{st} \text{len}_p^{st}(u) = d^k \sum_{a \in p} u_a, \quad \forall k \in K, p \in P^k.$$

Moreover, by (14) and (11), we have:

$$(16) \quad \sum_{k \in K} v^k \geq \sum_{s \in S} v^s \geq 1.$$

Hence, the solution of the dual of MCF_{tree} is also feasible for the dual of

MCF_{path} :

$$(17) \quad \min \sum_{a \in A} c_a u_a,$$

$$(18) \quad \gamma_{path}^* = \left\{ \begin{array}{l} \sum_{k \in K} v^k \geq 1, \\ d^k \sum_{a \in p} u_a \geq v^k \end{array} \right.$$

$$(19) \quad \left. \begin{array}{ll} \forall k \in K, p \in P^k, \\ u_a, v^k \geq 0, \end{array} \right. \quad \forall a \in A, k \in K,$$

It follows that $\gamma_{path}^* \leq \gamma_{tree}^*$. Since we already know that $\gamma_{tree}^* \leq \gamma_{path}^*$, we get that $\gamma_{path}^* = \gamma_{tree}^*$. \square

Remark 2.2 The problem $(MCF)_{tree}$ can be solved by column generation. The column generation oracle is based on the computation of a shortest-path tree with respect to the dual variables corresponding with constraints (7).

3 Single-source MCF

When all commodities have the same source, we propose the combinatorial polynomial-time algorithm 1 to solve MCF. The notation used to describe the algorithm is a standard one: the residual graph is the graph usually defined in the case of the maximum-flow algorithms.

Figures 1, 2 and 3 illustrate the algorithm on a simple instance.

Algorithm 1 Single-source MCF

Require: a directed graph $G = (V, A)$ with capacities $(c_a)_{a \in A}$ and a set of commodities $(s, t^k, d^k)_{k \in K}$.

Ensure: the value γ^* of the maximum concurrent flow between s and T .

- 1: initialize $\gamma^* \leftarrow 0$.
 - 2: **while** there exists a shortest-path tree (in terms of hops) τ spanning T rooted at s in the residual graph **do**
 - 3: find the maximum γ such that flows γd^k can be concurrently routed in τ between s and t^k .
 - 4: augment the flow and update the residual graph
 - 5: $\gamma^* \leftarrow \gamma^* + \gamma$
 - 6: **end while**
-

The algorithm can be seen as an extension of the Edmonds-Karp algorithm to solve the maximum flow problem [3].

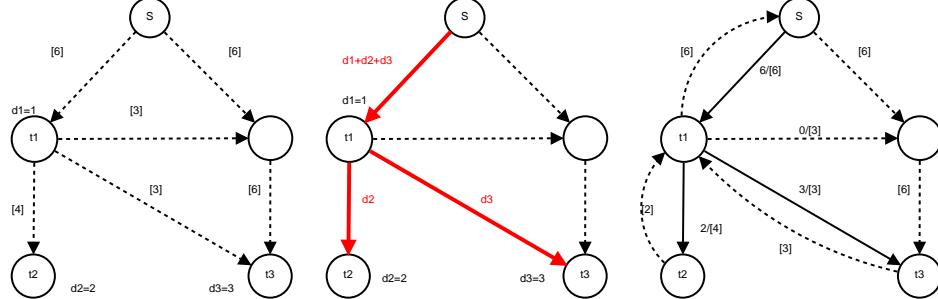


Fig. 1. Starting with an initial network, and a set of demands (d^k , written next the associated terminal) we build a terminal-spanning tree rooted at S . Then we sum the demands that will go through the corresponding arcs on the tree. Then we calculate the maximum γ we can send through this tree τ . It will be the minimum ratio $\frac{c_a}{\sum_{k \in V_d^\tau} d^k}$. In our example it is 1. We update the residual graph with backward arcs. Our current solution has a $\gamma^* = 1$

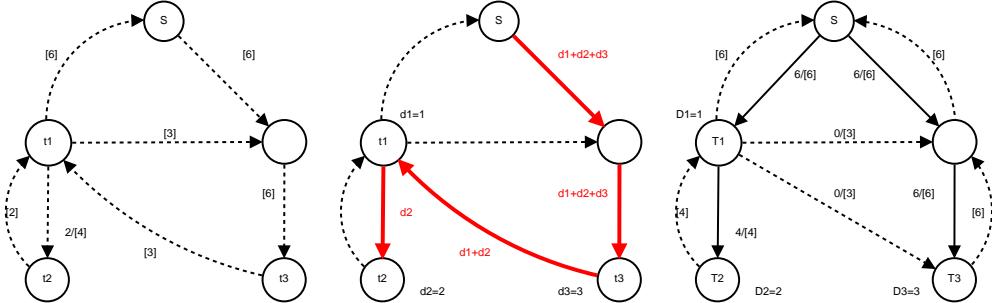


Fig. 2. On the first graph we have removed the unusable arcs (saturated). We build a new terminal-spanning tree, calculate the maximum $\gamma' = 1$, and update the residual graph. The new solution has now a $\gamma^* = 2$.

Theorem 3.1 The single-source maximum concurrent flow problem can be solved by Algorithm 1 in $O(m^2n)$ time.

The proof is skipped due to page limitation.

4 Some numerical results

Some numerical results are presented to compare the efficiency of the path and tree formulation. The tree and the path formulations are compared. MCF instances are generated using the RMF Generator [2]. The characteristics of the six sets of instances are described in details in Table 1. For each instance, we

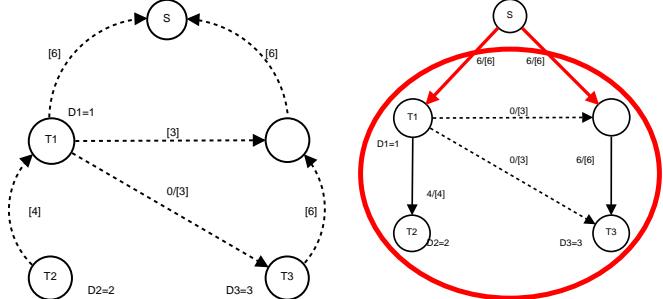


Fig. 3. It is now impossible to build a terminal-spanning tree in the residual graph. Then the algorithm stops, finding the sparsest cut. The solution is optimal.

reported an average value of several instances, run with XPRESS on a personal computer. In each table, we present the number of generated variables (columns) denoted by “Gen. Vars” when either the path formulation or the tree formulation is used. The number of variables used in the optimal solution is also presented and denoted by “Opt. Vars”. The number of iterations is reported. Notice that an iteration of the path flow problem corresponds to a computation of a shortest path for each commodity, while it corresponds to a computation of a shortest path tree for each source when the tree formulation is considered. The computing times are also presented. To facilitate the comparison between the two formulations, the ratios ($100 \times \text{tree}/\text{path}$) are given in the column percentage of Table 2.

The obtained results clearly show that the tree approach is much more efficient than the path approach.

Set	$ V $	$ S $	$ T^i $	a	b	$ E $
1	125	125	31	5	5	600
2	125	63	62	5	5	600
3	125	31	124	5	5	600
4	125	125	124	5	5	600
5	252	20	251	5	5	600
6	1000	1	999	10	10	5400

Table 1
Characteristics of the six sets of instances

Set	CPU time (s)			Gen. Vars			Opt. Vars			Iterations	
	path	tree	%	path	tree	%	path	tree	%	path	tree
1	810	203	25	33365	5375	16	4016	246	6	116	89
2	752	135	18	39818	3588	9	4047	188	5	113	95
3	692	80	12	40010	2031	5	3988	142	4	104	106
4	6327	502	8	116243	8655	7	15747	254	2	99	103
5	7930	502	6	104158	2564	2	5254	222	4	153	180

Table 2
Results related to sets 1-5

In Table 3, the combinatorial algorithm is compared with the tree-based approach in the single-source case. One can see that the combinatorial algorithm is really very fast.

Set	CPU time (s)			Iterations		
	tree	alg	%	tree	alg	%
6	461	0.6	0.13	395	538	137

Table 3
Set 6: $|V| = 1000$, $|S| = 1$, $|K| = 999$, $a = 10$, $b = 10$, and $|E| = 5400$

References

- [1] Ben-Ameur, W., and J. Neto: Acceleration of cutting planes and column generation algorithms: Application to network design; Networks, Vol 49 (2007) 3–17.
- [2] Derigs, U., and W. Meier, Implementing Goldberg’s Max-Flow-Algorithm A Computational Investigation (1989), ZOR - Methods & Models of OR volume 33, 383–403
- [3] Edmonds, J., and R.M. Karp, Theoretical improvements in algorithmic efficiency for network flow problem, Journal of the ACM, volume 19, 2 (1971), 248–264
- [4] Shahrokhi, F., and D.W. Matula, On Solving Large Maximum Concurrent Flow Problems, Journal of the ACM volume 37 (1991), 318–334

Modeling Earthquake Vulnerability of Highway Networks

İdil Arşık¹

*Department of Industrial Engineering
Koç University
Istanbul, Turkey*

F. Sibel Salman²

*Department of Industrial Engineering
Koç University
Istanbul, Turkey*

Abstract

In this study, we investigate the earthquake vulnerability of highway networks whose links are subject to failure. We propose a model called α -conservative failure model that aims to capture the dependency among link failures in the event of an earthquake. According to this model, we calculate a path-based accessibility measure to assess the expected weighted average shortest distance to serve a unit demand after the earthquake. We test the proposed link failure model on a case study of the earthquake vulnerability in Istanbul.

Keywords: Network Vulnerability, Reliability, Dependent Link Failures, Disaster Risk of Highways

¹ Email: iarsik@ku.edu.tr

² Email: ssalman@ku.edu.tr

1 Introduction

The assessment of the earthquake vulnerability of highway networks is essential for both structural strengthening decisions by highway administrators in the pre-disaster stage and efficiently carrying out post-disaster emergency activities by means of proper preparedness plans. In the event of an earthquake, network components such as bridges and viaducts are subject to failure. In this study, we assume that the nodes are reliable and postulate that both the radiation of seismic waves and the structural properties of network components create a dependency among network link failures. To represent this dependency structure, we suggest a link failure model which we call *α -conservative dependency model*. According to this dependency model, given the marginal survival probability of each link, we identify a joint probability distribution of the surviving links which characterize the surviving network. We calculate a path-based accessibility measure to evaluate the earthquake service level of the network. We test the model and calculate the measure for the Istanbul highway network which is subject to earthquake hazard.

2 Literature Review

To maintain societal welfare in general and to manage post-disaster emergency response activities effectively, the vulnerability of infrastructure networks has been evaluated in various studies. Sohn [7] evaluates the importance of highway network links under flood damage by introducing an accessibility index to embody the decreasing effect of distance and the volume of traffic influence on a highway network. Chang and Nojima [1] assess network performance in terms of network coverage and transportation accessibility in the post-disaster stage. Nagurney and Qiang [5] define a unified network performance measure which integrates the flow information and is applicable to different types of networks with either fixed or elastic demands on them. In order to evaluate the flow network reliability, Gertsbakh and Shpungin [3] estimate a topological characteristic of the network called *destruction spectrum (D-spectrum)* with an efficient Monte Carlo simulation.

In the event of a disaster, the dependency among network components that are exposed to the same amount of impact has been taken into consideration in a limited number of studies on network reliability. Taylor et al. [8] introduce a vulnerability analysis for the independent link failure case. Taking into account the node failures which would cause the failure of links attached to them, they claim that their analysis would work in dependent

link failure cases. Selçuk and Yücenem [6] define a spatial correlation between two elements to calculate the seismic capacity of a single component in the network. Günneç and Salman [4] define dependent link failures for highway networks under seismic hazard. Given the marginal link failure probabilities, they propose a link failure dependency model in which they define vulnerability-based (*VB*) and set-based (*SB*) dependency. According to this model, the links are divided into vulnerability sets based on deterministic *peak ground acceleration (PGA)* values. Links within the sets depend on each other with *VB-dependency*, while links in different sets fail independently with *SB-dependency*. In a vulnerability set, if a stronger (lower failure probability) link fails due to an earthquake, they assume that the weaker links fail with a probability equal to 1. *VB-dependency* yields $m+1$ surviving network realizations for a network composed of one vulnerability set with m links.

Our contribution to the literature is that we propose a new dependency model for link failures that captures earthquake vulnerability of highways. The model generates a family of joint probability distributions for the random network subject to link failures. The input parameter for the proposed model determines the degree of dependency. Furthermore, all possible realizations may have a positive occurrence probability, but for many realizations this probability gets very small. Therefore, our proposed model provides computational advantage in sampling based approaches (see Sections 4 and 6).

3 Notation

We are given an undirected graph $G = (N, E)$, where E is the set of edges/links and $N = \{N_1 \cup N_2 \cup N_3\}$ is the set of nodes consisting of three disjoint sets. N_1 represents the set of demand points, N_2 the set of emergency response facility (supply) points, and N_3 the set of junction points. Let $|E| = m$. A network realization $x = (x_1, x_2, \dots, x_m)$ is a binary vector representing the state of the links. If $x_i = 1$, link i is operational, and if $x_i = 0$, link i is non-operational. For each link $i \in E$, its marginal survival probability $p(i)$ is given. If link j is in the dependency set of link i , $n_{ij} = 1$, and 0 otherwise. Additionally, the casualty demand at node $v \in N_1$ are denoted by w_v . Π^{vy} is the set of paths between node $v \in N_1$ and node $y \in N_2$, and π_k^{vy} is the length of the k^{th} path between node $v \in N_1$ and node $y \in N_2$.

We further define the following. Let S represent the set of all possible network realizations (outcomes) defined by x . The probability of network realization $s \in S$ is defined as $p(s)$. Finally, $a_{sk}^{vy} = 1$, if the k^{th} path between nodes v and y survives in network realization s and 0, otherwise.

4 Proposed Conservative Dependency Model

For the failure model we propose, we partition the links into mutually exclusive sets. The links in different sets are spatially separated according to their *PGA* levels and the earthquake vulnerability of the structures on them. Each spatially and structurally defined set consists of links that have dependency among them, that is a dependency set according to the definition in [4], so that links in different sets fail independently. Different from [4], in a dependency set, the failure of a strong structure makes the failure of weaker structures of the same kind more likely by decreasing the survival probabilities of weaker links by the factor $100(1-\alpha)\%$, as defined below.

Definition 4.1 Given two links i and j in the same dependency set with survival probabilities $p(i)$ and $p(j)$, we say links i and j have α -conservative failure dependency, if $p(i) \leq p(j)$ implies $P(i \text{ survives} \mid j \text{ fails}) = (1-\alpha)p(i)$.

According to 4.1, links i and j within a dependency set do not fail independently. Their covariance, $\alpha[p(i)p(j)][1 - p(i)]$ which is greater than 0 confirms the dependency. *Pearson Product-Moment Correlation Coefficient* is equal to $\alpha\sqrt{p(i)p(j)[1 - p(i)][1 - p(j)]^{-1}}$. Thus, the correlation between two links in the same dependency set is proportional to α . As α increases, positive correlation between the links increases. According to the choice of α , the dependency model avoids unlikely network realizations and captures the likely realizations where the link failures are correlated by shared structural characteristics of structures on the links. For $\alpha=0$, the problem reduces to the independent link failure case and $\alpha=1$ is the totally dependent case.

Our model is conservative in the sense that we only focus on the decrease in probability of weaker links with respect to the failure of stronger links but not vice versa. Unlike the *VB-dependency* [4], our model does not limit the number of realizations but it changes the probabilities of network realizations according to a control parameter, α . The probability of each distinct network realization is the product of the updated marginal probabilities. However, we note that it is computationally intractable to generate the joint probability distribution for large m . Therefore, we propose a sampling algorithm to generate a sample of network realizations and then utilize sample average approximation to estimate any probabilistic measure of network vulnerability. We give the pseudo-code, Algorithm 1 below where the number of replications is specified as R .

Algorithm 1

```

Step 1: Order network links such that  $p(i) \geq p(i + 1)$ .
For  $r = 1$  to  $R$ 
  For  $i = 1$  to  $m$ 
    Step 2: Generate a network realization array  $x$  with size  $m$ .
    Step 3: Set  $x_i \leftarrow 0$ ,  $\xi_i \leftarrow 1$   $p'_\alpha(i) \leftarrow 0$ ,  $\forall i = 1, \dots, m$ .
    Step 4: Update the survival probability of link  $i$  as
      
$$p'_\alpha(i) = p(i) - p(i) * \alpha \sum_{j=1}^{i-1} \xi_j n_{ij}.$$

    Step 5: Generate a random number  $\varphi$  between 0 and 1.
    If  $p'_\alpha(i) \geq \varphi$ , then  $x_i \leftarrow 1$  and  $\xi_i \leftarrow 0$ .
    EndIf
  EndFor
EndFor

```

In Step 1, we sort the links with respect to their marginal survival probabilities in descending order. In Step 2, in each replication, we generate an array of all zeros with length m . In Step 4, we update the marginal survival probabilities of each link according to the state of the stronger links in its dependency set. Finally, in Step 5, we update the state of each link.

5 Performance Measure

As we mention in Section 2, the network vulnerability and reliability studies can be conducted on flow networks with edge capacities. In this study, we propose a path-based accessibility measure which considers the edge distances instead of edge capacities. Since the number of possible paths between any two nodes is usually high for large networks, it is not tractable to enumerate and check them all. Therefore, we generate a set of k loopless paths between each demand point (D) and emergency facility (F), where $D \in N_1, F \in N_2$, by a k -shortest path algorithm. When all the generated paths between a $D - F$ pair fail, we use a *Penalty Cost* which can either be interpreted as a proxy to costs of the paths that are not taken into consideration, or the cost of an alternative and reliable mode of transportation. For each network realization $s \in S$ and for each demand point $v \in N_1$, we represent the surviving shortest path to the closest facility, $\min_{y,k} \{\pi_k^{vy} a_{sk}^{vy}\}$, with d_{sv} . Finally, we calculate the expected weighted average shortest distance to send one unit of demand in

the network, $EWAD(N, E)$.

$$EWAD(N, E) = \frac{\sum_{s \in S} \sum_{v \in N_1} p(s) d_{sv} w_v}{\sum_{v \in N_1} w_v} \quad (1)$$

This accessibility measure incorporates disaster risk in terms of demand figures as well as the condition of highway network. The following lemma [2] shows that this measure satisfies three desirable properties.

Lemma 5.1 *Let $G = (N, E)$ be a random undirected graph with link failures according to α conservative link failure mode. The following properties hold.*

- (1) *Scale Invariance: The measure, $EWAD$ is invariant with respect to scale changes in demand values, w_v , at demand points.*
- (2) *Monotonicity: Let $E^* = E \cup \{e^*\}$. Then, $EWAD(N, E^*) \leq EWAD(N, E)$.*
- (3) *Membership in [Deterministic Upper Bound (DUB), Deterministic Lower Bound (DLB)]: $DLB \leq EWAD(N, E) \leq DUB$.*

Proof.

(1) Assume that the amount of demand at node $v \in N_1$ varies by a factor β , where $\beta \geq 0$ so that the new demand is $w_v^* = \beta w_v$, $\forall v \in N_1$. Denote by $EWAD^*$, the measure with the new demand vector w^* is equal to $EWAD^*(N, E) = \beta EWAD(N, E)$.

(2) Clearly, adding a new link to the network may decrease d_{sv} $\forall s \in S$ and $\forall v \in N_1$, but will not increase it.

(3) In the network realization s , where all links are operational, $a_{sk}^{vy} = 1$ $\forall v \in N_1$, $\forall y \in N_2$ and $\forall k \in \Pi^{vy}$. Then, $d_{sv} \forall v \in N_1$ is equal to the shortest path to the closest emergency response facility $y \in N_2$ and $EWAD$ is equal to DLB . In the network realization s , where all links fail, $d_{sv} \forall v \in N_1$ is equal to *Penalty Cost* which is also equal to DUB . Thus, $EWAD$ can take values between DLB and DUB . \square

Even checking k paths between each $D - F$ pair increases computational burden. Therefore, for each demand point, we select and work with the shortest l paths to any emergency facility points where $l \leq k * |N_2|$. This approximation reduces the run time significantly.

6 Computational Tests on the Case of Istanbul

Our network, which is based on the Istanbul highway network, consists of 83 edges and 60 nodes. Out of all nodes, 34 of them represent districts and 26

of them are the junction points in the highway. Among districts, we select 8 districts where the hospitals are densely populated as the emergency response facility points and identify the remaining districts as the demand points. We estimate the marginal survival probability of each link from the uniform distribution by mapping the earthquake vulnerability scores on the links and site dependent *PGA* levels. We estimate each demand value based on casualty rates and the population values given in the Istanbul Earthquake Masterplan, 2004.

For path generation, we take $k = 30$ and $l = 30$. We divide the edges into 9 dependency sets. We identify the longest path from any demand point as 75.95 km. We set the penalty cost to 76 km which is *DUB* on *EWAD*. We calculate the deterministic lower bound as 20.47 km for the case in which all edges survive. We then calculate *EWAD* for $R = 1,000,000$. We code Algorithm 1 and *VB-dependency* in MATLAB and estimate the proposed measures by sample average approximation. We report the elapsed time (*ET*) in seconds, estimated *EWAD* in kilometers, *Pearson Variation Index (PVI)* for five α values, the independent link failure case (*Indep.*) and the *VB-dependency(VB)*. We also report the lower bound ($LB_{95\%}$) and upper bound ($UB_{95\%}$) of 95% confidence interval on *EWAD* in the Table 1.

	$\alpha = 1$	$\alpha = 0.75$	$\alpha = 0.50$	$\alpha = 0.25$	$\alpha = 0.15$	<i>Indep.</i>	<i>VB</i>
<i>ET(sec)</i>	2382	2556	2737	2525	2053	1258	2538
<i>EWAD</i>	62.17	61.26	59.79	54.51	48.46	33.91	62.15
$LB_{95\%}$	62.15	61.24	59.77	54.48	48.44	33.90	62.13
$UB_{95\%}$	62.19	61.28	59.81	54.53	48.48	33.92	62.17
<i>PVI</i>	0.17	0.18	0.18	0.21	0.23	0.16	0.17

Table 1
Comparative Computational Results for Istanbul Highway

We note that the replication number R is sufficient because the *PVI* comes out to be small for the proposed probability distributions. We observe that as α decreases from 1 to 0.15, *EWAD* decreases because the level of dependency decreases. In the case of $\alpha = 0$, we see that *EWAD* is less than any other values with different α . *VB-dependency* yields similar *EWAD* to the $\alpha = 1$ case, which is expected. Finally, *EWAD* = 33.91 km is a reasonable distance. However, we include the effect of dependency in link failures, which increases the *EWAD* up to 62.17 km according to α .

7 Conclusion and Future Work

In this study, we proposed a new link failure dependency model and a new performance measure for the accessibility of demand points from supply points on a vulnerable highway under earthquake risk. For future work, dissimilar path generation methods in the literature can be tested instead of using the k -shortest path algorithm.

References

- [1] Chang, S.E., and N. Nojima, *Reliability of lifeline networks with multiple sources under seismic hazard*, Natural Hazards. **21** (2000), 1–18.
- [2] De-Los-Santos, A., and G. Laporte, and J.A. Mesa, and F. Perea, *Evaluating passenger robustness in a rail transit network*, Transportation Research Part C: Emerging Technologies. **20** (2000), 34–46.
- [3] Gertsbakh, I., and Y. Shpungin, “Spectral Approach to Reliability Evaluation of Flow Networks,” Proceedings of the European Modeling and Simulation Symposium. (2012), 68–73.
- [4] Günneç, D., and F.S. Salman, *Assessing the reliability and the expected performance of a network under disaster risk*, OR Spectrum. **33** (2011), 499–523.
- [5] Nagurney, A., and Q. Qiang, *Fragile networks: identifying vulnerabilities and synergies in an uncertain age*, International Transactions in Operational Research. **19** (2012), 123–160.
- [6] Selçuk, A.S., and M.S. Yücen, *Measuring post-disaster transportation system performance: the 1995 Kobe earthquake in comparative perspective*, Transportation Research Part A: Policy and Practice. **35** (2001), 475–494.
- [7] Sohn, J., *Evaluating the significance of highway network links under the flood damage: An accessibility approach*, Transportation Research Part A: Policy and Practice. **40** (2006), 491–506.
- [8] Taylor, M.A.P., and M. D’Este, and S.V.C Sekhar, *Application of Accessibility Based Methods for Vulnerability Analysis of Strategic Road Networks*, Networks and Spatial Economics. **6** (2006), 267–291.

Intersecting a simple mixed integer set with a vertex packing set

Agostinho Agra¹

*Department of Mathematics
University of Aveiro
Aveiro, Portugal*

Mahdi Doostmohammadi²

*Department of Mathematics
University of Aveiro
Aveiro, Portugal*

Cid Carvalho de Souza³

*Institute of Computing
University of Campinas
Campinas, Brazil*

Abstract

We consider a mixed integer set that results from the intersection of a simple mixed integer set with a vertex packing set from a conflict graph. This set arises as a relaxation of the feasible set of mixed integer problems such as inventory routing problems. We derive families of strong valid inequalities that consider the structures of the simple mixed integer set and the vertex packing set simultaneously.

Keywords: mixed integer set, conflict graph, valid inequalities, inventory routing

¹ Email: aagra@ua.pt

² Email: mahdi@ua.pt

³ Email: cid@ic.unicamp.br

1 Introduction

We consider a mixed integer set that results from the intersection of two well-known sets: a simple mixed integer set and the vertex packing set. This set arises as a relaxation of more general sets occurring in mixed integer problems such as inventory routing problems. Valid inequalities for this set can be used to strengthen the usual formulations for those models.

Let X be the set of points $(s, x) \in \mathbb{R} \times \mathbb{Z}^n$ satisfying

$$s + C \sum_{i \in N_1} x_i \geq D, \quad (1)$$

$$x_i + x_j \leq 1, \quad (i, j) \in E, \quad (2)$$

$$x_i \in \{0, 1\}, \quad i \in N, \quad (3)$$

$$s \geq 0, \quad (4)$$

where $N = \{1, \dots, n\}$ is the index set of binary variables, $E \subset N \times N$ is a set of index pairs, and $N_1 \subseteq N$. We assume $D > C \geq 0$ and D does not divide C . The graph $G = (N, E)$ is known as the conflict graph of pairwise conflicts between binary variables (see [3]). We denote by P the convex hull of X .

Set X arises from the relaxation of Inventory Routing Problems (IRP), and in particular in maritime inventory routing problems, see [1,2,10]. IRP combine the inventory management at each node with the routing of vehicles. Constraint (1) results from the relaxation of inventory constraints, where s is the stock level at a given node, D is the aggregated demand at that node during a set of periods, C is the vehicle capacity (when several vehicles are considered we may assume this capacity to be constant for all vehicles, otherwise we can take C as the maximum of these capacities) and x_i represents an arc traveled by a vehicle. N_1 is the index set of arcs entering to that particular node. Constraints (2) represent incompatible arcs, that is, arcs that cannot belong to the same route, for instance, due to time constraints. For such IRP few inequalities are known that combine the information from the routing with the information from the inventory. By studying set X we intend to derive new inequalities that can be used to improve the integrality gaps of such problems. Consider a simple example of a maritime IRP, with two ports: A and B. Constraints (1) can be obtained as a relaxation of the inventory constraints at port A, $x_i \in N_1$ may represent arcs entering into node A in different periods, and $x_i, i \in N \setminus N_1$ may represent arcs entering into node B. Valid inequalities for X including simultaneously nonnegative coefficients on $s, x_i, i \in N_1$ and $x_j, j \in N \setminus N_1$ relate visits to node B to the inventory at node A.

Set X can be regarded as the intersection of two well-known sets: $X = X_{VP} \cap X_{SMI}$, where X_{VP} is the vertex packing set defined by (2) - (3) (defined here in the space of x and s variables) that results from the conflict graph $G = (N, E)$, and X_{SMI} is the simple mixed set defined by (1), (3), (4).

The set X_{SMI} has been intensively used as a relaxation of several mixed integer sets, see [9] for examples. It is well-known that in order to describe the convex hull of X_{SMI} , denoted by P_{SMI} , if $|N_1| \geq \lceil D/C \rceil$, then it suffices to add to the trivial inequalities the following Mixed Integer Rounding (MIR) inequality

$$s + r \sum_{i \in N_1} x_i \geq r \lceil D/C \rceil, \quad (5)$$

where $r = D - C(\lceil D/C \rceil - 1)$.

On the contrary, a complete description of the convex hull of X_{VP} is not known and since optimizing a linear function over X_{VP} is a NP-hard problem, there is no much hope in finding such a description. Nevertheless, families of valid inequalities are known, see [5,6,7,8].

To the best of our knowledge set X has never been studied before. The most related mixed integer set considered before is the mixed vertex packing set studied by Atamturk et al. [4].

From the theoretical point of view, valid inequalities for X_{VP} and valid inequalities for X_{SMI} are valid for X . As, in general, P is strictly included in $P_{VP} \cap P_{SMI}$, there are fractional solutions that cannot be cut off by valid inequalities derived either for P_{VP} or for P_{SMI} . Hence, here we focus on valid inequalities derived for P that take into account properties from the two sets, simultaneously.

2 Basic Polyhedral Results

In this section we establish some basic results without proof.

Proposition 2.1 *Polyhedron P is full dimensional.*

Proposition 2.2 *Polyhedron P is unbounded with extreme ray $v = (1, \mathbf{0})$ where $\mathbf{0}$ is the null vector of dimension n .*

Notation. For every $i \in N$, $N(i) = \{j \in N \mid (i, j) \in E\}$ is the set of neighbors of i . Moreover, $N_1(i) = \{j \in N_1 \mid (i, j) \in E\}$, $N_0(i) = \{j \in N \setminus N_1 \mid (i, j) \in E\}$, and for $S \subseteq N$, we define $N_1(S) = \bigcup_{j \in S} N_1(j)$, $\tilde{N}_1(S) = \bigcap_{j \in S} N_1(j)$, and $N_0(S) = \bigcup_{i \in S} N_0(i)$.

For a graph $G(V, E)$ and a set $S \subseteq V$, $G[S]$ denotes the subgraph induced by S , and $\alpha(G[S])$ denotes its independence number.

Now we analyze the defining inequalities, called trivial inequalities.

Proposition 2.3 *Inequality (1) defines a facet of P .*

Proposition 2.4 *(i) $x_i \geq 0, i \in N$ and $s \geq 0$ are facet-defining for P . (ii) $x_i \leq 1, i \in N$ defines a facet of P if and only if $N(i) = \emptyset$. (iii) $x_i + x_j \leq 1$ defines a facet of P if and only if $N(i) \cap N(j) = \emptyset$.*

Similar results to the ones presented in Proposition 2.4 have been established in [4].

Proposition 2.5 *The MIR inequality defines a facet of P if and only if the following conditions hold: (i) $\alpha(G[N_1]) \geq \lceil \frac{D}{C} \rceil$; (ii) $\bigcap_{S \in \mathcal{I}_1(\lceil \frac{D}{C} \rceil)} N_0(S) = \emptyset$; (iii) $\bigcap_{S \in \mathcal{I}_1(\lceil \frac{D}{C} \rceil)} N_1(S) = \emptyset$; where $\mathcal{I}_1(\lceil \frac{D}{C} \rceil)$ is the collection of all independent sets of N_1 with cardinality $\lceil \frac{D}{C} \rceil$.*

Proposition 2.6 *Every facet-defining inequality for P_{VP} is a facet-defining inequality for P . Conversely, every facet-defining inequality $\sum_{j \in N} \alpha_j x_j + \beta s \geq \delta$, for P with $\beta = 0$, is a facet-defining inequality of P_{VP} .*

As a consequence of Proposition 2.6 we conclude that all the inequalities we are interested in, that is, those that combine the structure of the vertex packing set with the simple mixed integer set, must include the continuous variable.

Next we establish that if $\alpha(G[N_1]) \leq \lfloor \frac{D}{C} \rfloor$ then all the nontrivial facet-defining inequalities are those from the vertex packing polytope.

Proposition 2.7 *If $\alpha(G[N_1]) \leq \lfloor \frac{D}{C} \rfloor$, then the projection of P into the space of x variables coincides with the projection of P_{VP} into the same space.*

3 Nontrivial valid inequalities

In this section we present some of the valid inequalities derived for X . All these inequalities define facets of P under certain conditions. For brevity we focus only on the validity.

To derive the first set of inequalities, notice that if $x_j = 1$ for some $j \in N \setminus N_1$, then $x_i = 0, \forall i \in N_1(j)$. Hence it follows that $s + C \sum_{i \in N_1 \setminus N_1(j)} x_i \geq D x_j$ is valid for X . The following proposition extends this inequality to a general clique S in $N \setminus N_1$.

Proposition 3.1 *Let $S \subseteq N \setminus N_1$ be a clique. Then the following inequality is valid for X .*

$$s + C \sum_{i \in N_1 \setminus \tilde{N}_1(S)} x_i \geq D \sum_{i \in S} x_i. \quad (6)$$

Observe that by considering a clique S we need to remove from N_1 , in the left hand side of the inequality, all the variables with index in $\tilde{N}_1(S)$ which is a subset of $N_1(j)$. Therefore, the inequality is stronger than the inequality derived for $S = \{j\}$ if and only if $\tilde{N}_1(S) = N(j)$.

Now consider a $j \in N \setminus N_1$, and a subset $\tilde{S} \subset N_1 \setminus N_1(j)$ that cannot cover D in (1), that is, $\alpha(G[\tilde{S}]) \leq p \leq \lfloor \frac{D}{C} \rfloor$. Then, if $x_j = 1$ the amount that is not covered by \tilde{S} , $D - pC$, must be covered either from s or from $C \sum_{i \in N_1 \setminus (N_1(j) \cup \tilde{S})} x_i$. Hence, the inequality $s \geq (D - pC)(x_j - \sum_{i \in S} x_i)$ is valid for X . Again, this inequality can be extended to any clique of $N \setminus N_1$.

Proposition 3.2 *Let $S \subseteq N \setminus N_1$ be a clique and let $\tilde{S} \subseteq N_1 \setminus N_1(S)$ such that $\alpha(G[\tilde{S}]) \leq p \leq \lfloor \frac{D}{C} \rfloor$. Define $T = N_1 \setminus (\tilde{N}_1(S) \cup \tilde{S})$. Then the following inequality is valid for X .*

$$s \geq (D - pC) \left(\sum_{i \in S} x_i - \sum_{i \in T} x_i \right). \quad (7)$$

Next we establish several related families of inequalities. The general idea behind these inequalities is to impose a lower bound on the value of s when a subset of variables in $\bar{S} \subseteq N_1$ is set to one. Obviously, \bar{S} must be an independent set. In this case the variables corresponding to the neighbors of the nodes in \bar{S} are zero. Hence, if the remaining variables in N_1 are not enough to cover D , that is, if $C \sum_{j \in N_1 \setminus \bar{S}} x_j + C \sum_{j \in \bar{S}} x_j < D$, we can impose a positive bound on s .

We use the notation $(a)^+ = \max\{0, a\}$.

Proposition 3.3 *Let $S \subseteq N \setminus N_1$ such that the condition*

$$\alpha(G[N_1 \setminus N_1(\bar{S})]) \leq \left(\lfloor \frac{D}{C} \rfloor - |\bar{S}| + 1 \right)^+, \quad (8)$$

holds for all non-empty independent set $\bar{S} \subseteq S$. Then the following inequality is valid for X .

$$s + (C - r) \geq C \sum_{i \in S} x_i. \quad (9)$$

When, for a given S , condition (8) does not hold new valid inequalities can be derived under a weaker condition.

Proposition 3.4 *Let $S \subseteq N \setminus N_1$ such that the condition*

$$\alpha(G[N_1(S) \setminus N_1(\bar{S})]) \leq \left(\lfloor \frac{D}{C} \rfloor - |\bar{S}| + 1 \right)^+, \quad (10)$$

holds for all non-empty independent set $\bar{S} \subseteq S$. Then the following inequality is valid for X .

$$s + (C - r) \geq C \sum_{i \in S} x_i - C \sum_{i \in N_1 \setminus N_1(S)} x_i. \quad (11)$$

Observe that condition (8) coincides with condition (10) when $N_1(S) = N_1$ and, in that case, the two valid inequalities coincide.

The following proposition is based on another way to weaken condition (8). Basically, we remove from N_1 a subset S_1 and apply condition (8) to the reduced subset.

Proposition 3.5 *Let $S \subseteq N \setminus N_1$ and assume that condition (8) is violated for some independent set $A \subseteq S$. Define*

$$T = \left\{ A \subseteq S \mid A \text{ is independent set} \wedge \alpha(G[N_1 \setminus N_1(A)]) > \left(\lfloor \frac{D}{C} \rfloor - |A| + 1 \right)^+ \right\}.$$

Let $S_1 \subseteq \bigcup_{A \in T} (N_1 \setminus N_1(A))$ such that

$$\alpha(G[N_1 \setminus (N_1(A) \cup S_1)]) \leq \left(\lfloor \frac{D}{C} \rfloor - |A| + 1 \right)^+, \forall A \in T.$$

Then the following inequality is valid for X .

$$s + (C - r) \geq C \sum_{i \in S} x_i - C \sum_{i \in S_1} x_i. \quad (12)$$

Under more restrictive conditions Constraints (12) can be strengthened. Assume that in addition to the conditions of Proposition 3.5 we have that for all $\bar{S} = \{j, k\} \subseteq S$ such that \bar{S} is an independent set and $x_j = x_k = 1$ implies $x_i = 0, \forall i \in S_1$. Then

$$s + (C - r) \geq C \sum_{i \in S} x_i - r \sum_{i \in S_1} x_i, \quad (13)$$

is valid for X .

Combining Proposition 3.4 with Proposition 3.5 we have the following inequalities valid under more weaker conditions.

Proposition 3.6 *Let $S \subseteq N \setminus N_1$ and assume that condition (10) of Proposition 3.4 is violated for some $A \subset S$. Define*

$$T = \left\{ A \subset S \mid A \text{ is independent set} \wedge \alpha(G[N_1(S) \setminus N_1(A)]) > \lfloor \frac{D}{C} \rfloor - |A| + 1 \right\}.$$

Let $S_1 \subseteq \bigcup_{A \in T} (N_1(S) \setminus N_1(A))$ such that

$$\alpha(G[(N_1(S) \setminus (N_1(A)) \cup S_1)]) = \left(\lfloor \frac{D}{C} \rfloor - |A| + 1 \right)^+, \forall A \in T.$$

Then the following inequality is valid for X .

$$s + (C - r) \geq C \sum_{i \in S} x_i - C \sum_{i \in N_1 \setminus N_1(S)} x_i - C \sum_{i \in S_1} x_i. \quad (14)$$

Again Constraints (14) can be strengthened in a similar way as Constraints (12).

4 Conclusions and current research

We have identified many families of facet-defining inequalities for P , some of them were omitted for brevity. Also, some of those facet-defining inequalities can be obtained by lifting weaker variants of the inequalities discussed in the previous section.

From the computational point of view it is important to address the separation problem associated to the several families of inequalities presented. Since, in general, the separation problems are NP-hard since they involve the computation of the independence number of a subgraph, finding good and fast heuristic separation algorithms is fundamental in order to use efficiently the inequalities derived.

Given the large number of families of facet-defining inequalities identified, we are investigating, from the computational point of view, which families lead to a greater reduction on integrality gaps. As a future research we aim to test the effectiveness of the inclusion of these most relevant families of inequalities in a branch and cut scheme to solve maritime inventory routing instances.

Acknowledgement

The work of the first two authors was supported by *FEDER* funds through *COMPETE* and by Portuguese funds through the *Center for Research and Development in Mathematics and Applications* and the Portuguese Foundation for Science and Technology, within project PEst-C/MAT/UI4106/2011 with COMPETE number FCOMP-01-0124-FEDER-022690. The third author is supported by grants from CNPq, CAPES and FAEPEX-UNICAMP.

References

- [1] Agra, A., H. Andersson, M. Christiansen, and L. Wolsey, *A Maritime Inventory Routing Problem: Discrete Time Formulations and Valid Inequalities*, submitted to journal.
- [2] Agra, A., M. Christiansen, and A. Delgado, *Mixed Integer Formulations for a Short Sea Fuel Oil Distribution Problem*, *Transportation Science* **47** (2013), 108-124.
- [3] Atamturk, A., G. L. Nemhauser, and M. W. P. Savelsbegh, *Conflicts Graph in Integer Programming*, *European Journal of Operations Research* **121** (1998), 40-55.
- [4] Atamturk, A., G. L. Nemhauser, and M. W. P. Savelsbegh, *The Mixed Vertex Packing Problem*, *Mathematical Programming* **89** (2000), 35-53.
- [5] Easton, T., K. Hooker, and E. K. Lee, *Facets of the Independent Set Polytope*, *Mathematical Programming* **98** (2003), 177-199.
- [6] Fouilhoux, P., M. Labbe, A. R. Mahjoub, and H. Yaman, *Generating Facets for the Independence System Polytope*, *SIAM Journal on Discrete Mathematics* **23** (2009), 1484-1506.
- [7] Nemhauser, G. L., and L. E. Trotter, *Properties of Vertex Packing and Independence System Polyhedra*, *Mathematical Programming* **6** (1974), 48-61.
- [8] Padberg, M. W., *On the Facial Structure of Set Packing Polyhedra*, *Mathematical Programming* **5** (1973), 199-215.
- [9] Pochet, Y., and L. Wolsey, “Production Planning by Mixed Integer Programming,” Springer, New York, 2005.
- [10] Song, J-H., and K. C. Furman, *A Maritime Inventory Routing Problem: Practical Approach*, *Computers and Operations Research* **40** (2013), 657-665.

Solving the Two-Stage Robust FTTH network design Problem under Demand Uncertainty

Cedric Hervet ¹

Orange Labs, ENSTA-CEDRIC, France

Alain Faye, Marie-Christine Costa ²

ENSIIE/ESNTA-CEDRIC, France

Matthieu Chardy, Stanislas Francfort

Orange Labs, France

Abstract

For the past few years, the increase in high bandwidth requiring services forced telecommunication operators like France Telecom - Orange to engage the deployment of optical networks, the Fiber To The Home Gigabit Passive Optical Network (FTTH GPON) technology, leading to new design problems. Such problems have already been studied. However, to the best of our knowledge, without taking into account the future demand uncertainty. In this paper, we propose a model for a two-stage robust optimization FTTH network design problem tackling the demand uncertainty. We propose an exact algorithm, based on column and constraint generation algorithms, and we show some preliminary results.

Keywords: Robust Optimization, Network Design, Mathematical Programming.

¹ Email:cedric.hervet@orange.com

² Email:alain.faye@ensiie.fr

1 Introduction

Optimizing the fixed optical access network deployment is a priority for telecommunication operators. Let us present first the global scheme for the FTTH GPON technology which is mainly composed of optical fibers and splitters. A fiber entering a splitter is split into a given number of fibers of a higher level; this number is the splitter capacity. The level 1 fibers are originated from the source node, called Optical Line Terminal (OLT) and, for quality of service requirements, only the last level of fibers can supply the client demands. The problem is to locate the splitters and to determine the route of the fibers in order to serve given demands, while minimizing the overall cost of both installed splitters and fibers. One can find complete descriptions and models for this problem in [11] or in [13]. In [12], the impact of the solution in terms of future maintenance cost is also examined.

However, the demand is uncertain and considering the low precision one can have on it, we decided to tackle the problem with robust optimization, and more precisely, with two-stage robust optimization [5,6,14]: decisions are taken in two stages. First stage variables correspond to decisions taken before discovering the actual values of uncertain data, second stage variables are determined once uncertainty has been revealed, this is the recourse problem. Moreover, robust optimization allows one to handle problems without making a lot of assumptions on random data distribution, contrary to stochastic optimization. It tries to provide acceptable solutions, no matter what happens in the future, *i.e.* whatever the scenario that occurs.

This approach is very conservative because the cost one pays to protect himself against very unlikely scenarios may be seen as too prohibitive. But in fact, the marketing service at France-Telecom Orange can "*expect a given percentage of the total household number in an area to purchase our offer for optical fiber*". That allows us, following the ideas of [8,17], to bound the total scale deviation of the uncertain data.

Demand uncertainty leads to problem formulations with right-hand-side uncertainty, a specific and hard version of robust optimization problems: see [2,9,16] for general models and [1,3,4,10,15] for network design problems. All these papers consider continuous recourse variables, but there is very few works available on problems like ours, with integral recourse variables [7,18].

In this paper, we solve a two-stage robust optical network design problem with integral recourse variables. We first introduce a model for this problem, then we propose an algorithm inspired from [18] to solve it to optimality and we finally examine some experimental results.

2 Model for the two-stage robust problem

2.1 The deterministic problem

We briefly describe the deterministic version of the problem. The given infrastructure is represented by an undirected graph $G = (V, E)$, a_i^{max} denotes the demand on each node $i \in V$ and \mathbf{a}^{max} the corresponding vector (the vectors are written in bold). There are K levels of splitters ($K+1$ levels of fibers), m^k denotes the capacity of a level k splitter and C^k its cost. The cost of a level k fiber along an edge $(i, j) \in E$ is denoted by d_{ij}^k . Note that in reality, splitter cost is much more important than fiber cost. The set of solutions that satisfy both a given demand vector \mathbf{a} and the architecture requirements is denoted by $P_G(\mathbf{a})$. The variables are the number of level k splitters to install on node $i \in V$, denoted by z_i^k , and the number of level k fibers routed along an edge $(i, j) \in E$ from i to j , denoted by f_{ij}^k . The fibers have to be installed in existing ducts that are considered large enough whatever the routing. Then, the deterministic problem PON^{det} , without capacity constraints, may be written as such:

$$PON^{det} = \min_{(\mathbf{f}, \mathbf{z}) \in P_G(\mathbf{a})} \sum_{i \in V} \sum_{k=1}^K C^k z_i^k + \sum_{(i,j) \in E} \sum_{k=1}^{K+1} d_{ij} (f_{ij}^k + f_{ji}^k) \text{ with}$$

$$(\mathbf{f}, \mathbf{z}) \in P_G(\mathbf{a}) \Leftrightarrow \begin{cases} \sum_{j|(i,j) \in E} (f_{ji}^1 - f_{ij}^1) \geq z_i^1, \forall i \in V \setminus \{0\} \quad (0 = OLT) \\ \sum_{j|(i,j) \in E} (f_{ji}^k - f_{ij}^k) + m^{k-1} z_i^{k-1} \geq z_i^k, \forall i \in V, \forall k = 1..K \\ \sum_{j|(i,j) \in E} (f_{ji}^{K+1} - f_{ij}^{K+1}) + m^K z_i^K \geq a_i, \forall i \in V \\ z_i^k \in \mathbb{N}, \forall i \in V, \forall k = 1..K \\ f_{ij}^k \in \mathbb{R}^+, \forall (i, j) \in E, \forall k = 1..K \end{cases}$$

These constraints enable the fiber flow to be coherent with installed splitters in the network and with the client demands (see [11] for a complete description).

2.2 The two-stage robust problem

In reality, the future demand is greatly uncertain and it is highly expectable that not every household will purchase the fiber offer of a single telecommunication operator. Therefore, considering the very high amount of necessary investments, it was decided that the initial optical network would be designed to serve only a part of households. But, once the network is deployed, it is not acceptable for operational units to add or move fibers in order to supply

clients that were not connected to the network. However, adding or removing splitters is much more acceptable and the splitter cost is the major part of the investment [11]. That is why it was decided to route the fibers in order to serve each household but to install the splitters and light the fibers only *once the actual demand is known*.

The question is then: "how should one design the fiber network so that its worst future splitter cost may be the lowest possible ?"

Knowing the number of potential subscribers at each node i , denoted by a_i^{max} , it is rather simple for marketing services to estimate what could be the maximum number of clients in the whole studied area. Let us denote it by \bar{A} ($0 \leq \bar{A} \leq \sum_{i \in V} a_i^{max}$). Then, the uncertainty set containing the random demand vector \mathbf{a} , denoted by Λ , is defined as such :

$$\Lambda = \left\{ \mathbf{a} \in \mathbb{N}^{|V|} \left| \begin{array}{l} a_i \leq a_i^{max}, \forall i \in V \\ \sum_{i \in V} a_i \leq \bar{A} \end{array} \right. \right\} = \{\mathbf{a}^s, s = 1, \dots, S\}$$

where S is the total number of scenarios and \mathbf{a}^s denotes a scenario. As in section 2.1, let us denote by f_{ij}^k the number of level k fibers routed along the edge (i, j) before the demand is revealed (*i.e.* so that every household is connected to the OLT) and by z_i^k the maximum number of level k splitters that node i can receive (*i.e.* the number of splitters one would install if all f_{ij}^k were used): f_{ij}^k and z_i^k are the first stage variables. The variables ζ_i^k concern the number of level k splitters that will be installed on node i , and φ_{ij}^k the number of level k fibers that will be lighted along the edge (i, j) , once the demand is revealed. Then, the two-stage robust problem PON^{rob} may be written as such :

$$PON^{rob} = \min_{(\mathbf{f}, \mathbf{z}) \in P_G(\mathbf{a}^{max})} \left(\sum_{(i,j) \in E} \sum_{k=1}^{K+1} d_{ij} (f_{ij}^k + f_{ji}^k) + \max_{\mathbf{a} \in \Lambda} \min_{(\boldsymbol{\varphi}, \boldsymbol{\zeta}) \in P_G(\mathbf{a})} \sum_{i \in V} \sum_{k=1}^K C^k \zeta_i^k \right) \quad \boldsymbol{\zeta} \leq \mathbf{z}, \boldsymbol{\varphi} \leq \mathbf{f}$$

3 Column-and-Constraint generation algorithm

We solve the two-stage robust problem PON^{rob} using the column-and-constraint generation algorithm of Zhao and Zeng [18] which is a method dedicated to robust problem with mixed integer recourse variables. We adapt this method

in the context of our problem. The problem PON^{rob} has some characteristics. First it satisfies the full recourse property that is the recourse problem (*i.e.* the internal minimization problem) has a feasible solution for any value of the higher level variables \mathbf{z} , \mathbf{f} and the demand vectors \mathbf{a} . Next the continuous fiber variables $\boldsymbol{\varphi}$ have a null cost in the objective function of the recourse problem. These characteristics are fully used to solve PON^{rob} . Now we shortly describe the main ideas of our method.

The uncertainty set contains a finite number S of elements denoted by $\Lambda = \{\mathbf{a}^s\}_{s=1}^S$. To each \mathbf{a}^s is associated a splitter recourse variable $\boldsymbol{\zeta}^s$ and a fiber recourse variable $\boldsymbol{\varphi}^s$. For shortening, we denote $\mathbf{C}\boldsymbol{\zeta} = \sum_{i \in V} \sum_{k=1}^K C^k \zeta_i^k$.

By enumerating all possible assignments of \mathbf{a} , *i.e.* all scenarios of demands, the problem is written :

$$\min_{\eta, (\mathbf{f}, \mathbf{z}), (\boldsymbol{\zeta}^s, \boldsymbol{\varphi}^s)} \sum_{(i,j) \in E} \sum_{k=1}^{K+1} d_{ij} (f_{ij}^k + f_{ji}^k) + \eta$$

with

$$\begin{cases} \eta \geq \mathbf{C}\boldsymbol{\zeta}^s, s = 1, \dots, S \\ (\boldsymbol{\zeta}^s, \boldsymbol{\varphi}^s) \in P_G(\mathbf{a}^s), s = 1, \dots, S \\ \boldsymbol{\zeta}^s \leq \mathbf{z}, \boldsymbol{\varphi}^s \leq \mathbf{f}, s = 1, \dots, S \\ (\mathbf{f}, \mathbf{z}) \in P_G(\mathbf{a}^{max}) \end{cases}$$

As S may be huge, only a subset of constraints is used through a constraint generation algorithm. The sub-problem for finding a cut is:

$$Q(\mathbf{f}, \mathbf{z}) = \max_{\mathbf{a} \in \Lambda} \min_{(\boldsymbol{\zeta}, \boldsymbol{\varphi}) \in P_G(\mathbf{a}), \boldsymbol{\zeta} \leq \mathbf{z}, \boldsymbol{\varphi} \leq \mathbf{f}} \mathbf{C}\boldsymbol{\zeta}$$

The solution of this problem gives a new scenario of demand that must be added as a constraint if $\eta < Q(\mathbf{f}, \mathbf{z})$ otherwise the algorithm stops and PON^{rob} is solved.

$Q(\mathbf{f}, \mathbf{z})$ is written by enumerating all possible assignments of the integer splitter variables $\boldsymbol{\zeta}$ in the internal minimization problem that will be generated through another column and constraint generation algorithm. Because of the article size limitations, we do not give details of the solving method for the subproblem $Q(\mathbf{f}, \mathbf{z})$, since it is much more complex in its refinements yet very similar to the method used to solve the master problem. We advise the reader to refer to [18].

4 Experimental results

The method proposed to solve the robust problem involves a huge number of sub-problems to solve since there are two interlinked constraints generation algorithms. Then we can solve only small instances of the problem. Nevertheless, the interest of our tests is to emphasize the economic interest of a robust approach. Graphs were first randomly generated as tree-graphs to which we added a very few number of random edges between sons in order to obtain graphs topology close to our real-world ones. Tests were performed on more than 100 instances of various sizes (from 5 to 10 nodes) and a mean node degree of 3 (which is compliant with real world instances). Behaviours presented here were always qualitatively observed in our tests.

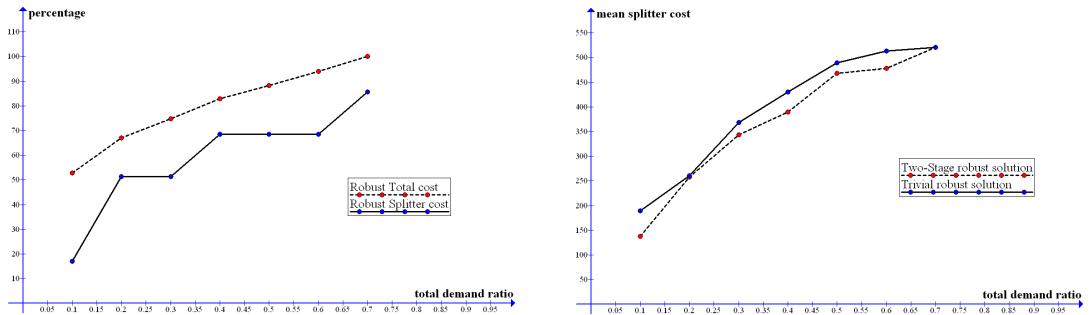


Fig. 1. Cost of the robust solution, compared to a 100% demand solution (the lowest curve is for the worst splitter cost).

Fig. 2. Comparison of the mean splitter cost over 2000 random scenarios between the trivial solution and the two-stage robust solution.

Figures 1 and 2 are extracted from results computed on a small instance of 5 nodes. Without any algorithmic improvement, we were not able to solve instances to optimality with 10 nodes or more. However, the behaviour shown in this paper has always been observed. Figure 1 assesses the cost improvement of the robust solution in function of the size of the uncertainty set (we consider various proportions of the total demand for the maximum demand possible in the area). The cost is expressed as a percentage of the One-Stage robust solution cost, which is computed by solving PON^{det} for the demand vector α^{max} . The step-like aspect of the lowest curve is due to the fact that splitters are much more costly than fibers, and adding one splitter in the worst-case scenario implies a huge cost step. Of course, as the uncertainty set size increases, costs of both solutions tend to be equal.

Robust optimization deals with the worst case, but it is quite unlikely that every scenario will be a bad one. Figure 2 shows how effective is the two-stage

robust solution on any scenario compared to a trivial solution where no two-stage robust optimization has been made (here, we take the PON^{det} problem solution with \mathbf{a}^{max}). We randomly draw 2000 scenarios in the uncertainty set, and we compute the best splitter cost, given \mathbf{f} and \mathbf{z} , for those scenarios. The figure shows that the mean cost of the robust solution is always better than the one of the trivial solution. This was not granted since we only optimize the worst-case splitter cost.

5 Conclusion

This paper introduced a model for the Passive Optical Network design with demand uncertainty, which was tackled with robust optimization techniques. The problem is solved to optimality through a complex column-and-constraints generation algorithm, whose complexity makes real-life instances too big to be solved. However, obtaining optimal robust solutions for this problem enabled us to show the interest of having a robust approach for solving this problem, especially compared to stochastic programming approaches since our tests showed that we tend to optimize the cost of all scenarios, not only the worst-case one. Moreover, as we do not need to consider probability laws for the demand uncertainty that we do not have in practice, robust optimization seems really suited for this particular problem. Future research steps are to draw conclusions from the analysis of robust optimal solutions in order to design heuristics, or local moves that may improve a solution's robustness.

References

- [1] Atamturk, A. and M. Zhang, *Two-stage robust network flow and design under demand uncertainty*, Operation Research **55** (2007), 662–673.
- [2] Babonneau, F., Vial, J-P. and R. Apparigliato, *Handbook on "Uncertainty and Environmental Decision Making"*, chapter Robust Optimization for environmental and energy planning, International Series in Operations Research and Management Science, Springer Verlag (2009).
- [3] Babonneau, F., Klopfenstein, O., Ouorou, A. and J-P. Vial, *Robust capacity expansion solutions for telecommunications networks with uncertain demands*, Technical paper, Orange Labs R&D,(2009).
- [4] Ben-Ameur, W. and H. Kerivin, *Routing of Uncertain Traffic Demands*, Optimization and Engineering **6** (2005), 283–313.

- [5] Ben-Tal, A., El-Ghaoui, L. and A. Nemirovski, *Robust Optimization* (2006), Princeton Series in Applied Mathematics.
- [6] Bertsimas, D., Brown, D. and C. Caramanis, *Theory and Applications of Robust Optimization*, SIAM Review **53** (2010), 464–501.
- [7] Bertsimas, D. and C. Caramanis, *Finite adaptability for linear optimization*, IEEE Trans. Automat. Control **55** (2010), 2751–2766.
- [8] Bertsimas, D., and M. Sim, *The Price of Robustness*, Operations Research **52** (2004), 35–53.
- [9] Billionnet, A., Costa, M.-C. and P.-L. Poirion, *2-Stage Robust MILP with continuous recourse variables*, CEDRIC report N12-2683 (2012), 13 p.
- [10] Bertsimas, D., and M. Sim, *Robust Discrete Optimization and Network Flows*, Mathematical Programming Series B **53** (2003) , 49–71.
- [11] Chardy, M., Costa, M-C., Faye, A., and M. Trampont, *Optimizing splitter and fiber location in a multilevel optical FTTH network*, European Journal of Operational Research **222** (2012), 430–440.
- [12] Hervet, C., and M. Chardy, *Passive optical network design under operations administration and maintenance considerations*, Journal of Applied Operational Research **4** (2012), 152–172.
- [13] Kim, Y., Lee, Y., and J. Han, *A splitter location-allocation problem in designing fiber optic access networks*, European Journal of Operational Research **210** (2011), 425–435.
- [14] Kouvelis, P. and G. Yu, *Robust Discrete Optimization and Its Applications*, Kluwer Academic Publishers (1997).
- [15] M. Minoux, *Robust network optimization under polyhedral demand uncertainty is NP-hard*, Discrete Mathematics **158** (2010), 597–603.
- [16] M. Minoux, *On 2-stage robust LP with RHS uncertainty : complexity results and applications*, Journal of Global Optimization **49** (2011), 521–537.
- [17] Thiele, A., Terry, T., and M. Epelman, *Robust Linear Optimization With Recourse* submitted in Naval Research Logistics (2009).
- [18] Zhao, L. and B. Zeng, *An Exact Algorithm for Two-stage Robust Optimization with Mixed Integer Recourse Problems*, submitted, available on Optimization-Online.org, 2012.

Solving the Passive Optical Network with Fiber Duct Sharing Planning Problem Using Discrete Techniques[★]

S.P. van Loggerenberg^{a,1}, M.J. Grobler^{a,1} and
S.E. Terblanche^{b,1}

^a *School for Electric, Electronic and Computer Engineering, North-West University, Potchefstroom, South Africa*

^b *Centre for Business Mathematics and Informatics, North-West University, Potchefstroom, South Africa*

Abstract

Similar to the constrained facility location problem, the passive optical network (PON) planning problem necessitates the search for a subset of deployed facilities (*splitters*) and their allocated demand points (*optical network units*) to minimize the overall deployment cost. In this paper we use a mixed integer linear programming formulation stemming from network flow optimization to construct a heuristic based on limiting the total number of interconnecting paths when implementing fiber duct sharing. Then, a disintegration heuristic involving the construction of valid clusters from the output of a k means algorithm, reduce the time complexity while ensuring close to optimal results. The proposed heuristics are then evaluated using a real-world dataset, showing favourable performance.

Keywords: MILP, Network Modelling, Optimization, PON, FTTH

[★] Research partially sponsored by the Telkom CoE, South Africa

¹ Email: {20289278, leenta.grobler, fanie.terblanche}@nwu.ac.za

1 Introduction

In accordance with the current exponential growth in telecommunication network bandwidth requirements, service providers are opting for optical fiber-based solutions in the last mile deployment. With fiber interconnects moving from the backbone to the access networks and the accompanying large capital expenditure, optimization of these networks become paramount. The main contender for these fiber-based access networks is the Passive Optical Network (PON). A single optic fiber cable runs from a Central Office (CO) to a cabinet housing a passive distribution unit called an optic splitter. In the case of Fiber to the Home (FTTH), the optic signal is then distributed to a number of smaller fibers running directly to termination points known as Optical Network Units (ONUs) at customer premises. Since these fibers are installed in subterranean ducts, expensive trenches need to be dug all the way from the CO to the customer. The problem is then to minimize the cost of connecting customer premises to the CO by choosing appropriate locations for these splitters and deciding which customers to connect to them.

A number of papers deal with this problem using both discrete optimization and meta-heuristic techniques, although most ignore a fundamental part in the deployment phase of real-world networks - fiber duct sharing. Since it would be impractical and expensive to dig a trench for each individual fiber, a single duct usually contains a number of fibers that share a part of their route to customer premises. This paper introduces a way of using concepts from network flow optimization to incorporate fiber duct sharing into a mixed integer formulation of the PON planning problem.

The rest of the paper is organized as follows: In section 2 the PON planning problem is discussed before formulating a mixed integer model in section 3. The path and disintegration heuristics are given in sections 4 and 5. Results from solving the model are given in section 6 before concluding the paper in section 7.

2 Problem definition

The PON planning problem can be seen as an advanced form of the constrained facility location problem, with facilities and demand points substituted with splitters and ONUs respectively. Assume an undirected graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ is given with edge costs $c_e \geq 0, e \in \mathcal{E}$, a disjoint partition $\mathcal{J} = \{S, U\}$ and a CO location $c \in \mathcal{V} \setminus \mathcal{J}$. The set $S \subset \mathcal{V}$ contains the possible splitter locations and $U \subset \mathcal{V}$ is the set of fixed ONU locations. As-

sume deployment costs $c_{\text{co}} \geq 0$, $c_s \geq 0$ and $c_{\text{onu}} \geq 0$ for the CO, splitters and ONUs respectively and a splitter capacity of κ . Furthermore, assume c_t and c_f is a fixed cost per length of trenching and fiber respectively.

Define a commodity pair $k \in K$. The set K consists of all possible pairs of the CO and splitters as well as all possible pairs of splitters and ONUs. For each commodity pair $k = \{i, j\} \in K$, define a set $P(k) \subseteq P$ of all non-cyclic paths between $i \in \mathcal{V}$ and $j \in \mathcal{V}$. Next, define a set $E \subseteq \mathcal{E}$ of all edges traversed in paths $p \in P$ and a set $P(e) \subseteq P$ containing all paths that traverses edge $e \in E$. Conversely, $E(p) \subseteq E$ is the set of all edges contained within path $p \in P$.

Two additional constraints are applicable to PONs: maximum and differential network reach. The total length of fiber connecting the CO with an ONU $j \in U$ through a splitter $i \in S$, i.e. the network reach, may not exceed a threshold $\ell_{\max}^{\text{total}}$ due to optic loss. To avoid synchronization issues between ONUs, the difference between the maximum and minimum network reach for a splitter $i \in S$ may not exceed $\ell_{\max}^{\text{diff}}$ [1].

With paths representing fiber cables and edges representing trenches, the PON planning problem then becomes the search for a subset of deployed splitters such that

- each ONU connects to one and only one splitter via a single path,
- each splitter connects to the CO via a single path,
- a maximum of κ ONUs can connect to a single splitter,
- the maximum and differential network reach constraints are satisfied and
- the sum of the deployment, path and edge costs are minimized.

The non-cyclic paths in set P would typically be formulated as Steiner trees rooted at c and $i \in S$ with Steiner nodes $i \in S$ and $j \in U$ respectively. In our approach, these paths will be calculated beforehand using a heuristic specific to the PON planning problem. Hence, the set P will be treated as a parameter in the formulation.

3 Mixed Integer Formulation

From the above problem, a MILP model can now be formulated. Let y_p indicate the usage of the path $p \in P$ and x_e the usage of edge $e \in E$. Let ψ_i indicate the deployment of splitter $i \in S$. $k_{ij}^{\text{so}} \in K$ denotes the commodity pair of splitter $i \in S$ and ONU $j \in U$ while $k_i^{\text{cs}} \in K$ denotes the commodity pair of the CO and splitter $i \in S$. Parameter ℓ_p denotes the total length of path

$p \in P$ while the variables ℓ_i^{\min} and ℓ_i^{\max} denote the minimum and maximum network reach for splitter $i \in S$ respectively. Let ℓ_e be the length of edge $e \in E$. As done by Li et al. [3], the introduction of a binary *if-then* variable $d_{ij}, i \in S, j \in U$ and a large value, Δ , allows the formulation of the PON problem as follows:

$$\min \quad OBJ = c_{co} + \sum_{i \in S} \psi_i c_s + |U| c_o + \sum_{e \in E} c_t \ell_e x_e + \sum_{p \in P} c_f \ell_p y_p \quad (1)$$

$$\text{s.t.} \quad \sum_{i \in S} \sum_{p \in P(k_{ij}^{\text{so}})} y_p = 1, \quad \forall j \in U \quad (2)$$

$$\sum_{j \in U} \sum_{p \in P(k_{ij}^{\text{so}})} y_p \leq \kappa \psi_i, \quad \forall i \in S \quad (3)$$

$$\sum_{p \in P(e)} y_p \leq \Delta x_e, \quad \forall e \in E \quad (4)$$

$$\ell_i^{\min} - \left(\sum_{p \in P(k_i^{\text{cs}})} y_p \ell_p + \sum_{p \in P(k_{ij}^{\text{so}})} y_p \ell_p \right) \leq \Delta d_{ij}, \quad \forall i \in S, \forall j \in U \quad (5)$$

$$\left(\sum_{p \in P(k_i^{\text{cs}})} y_p \ell_p + \sum_{p \in P(k_{ij}^{\text{so}})} y_p \ell_p \right) - \ell_i^{\max} \leq \Delta d_{ij}, \quad \forall i \in S, \forall j \in U \quad (6)$$

$$\sum_{p \in P(k_{ij}^{\text{so}})} y_p \leq \Delta (1 - d_{ij}), \quad \forall i \in S, \forall j \in U \quad (7)$$

$$\ell_i^{\max} \leq \ell_{\max}^{\text{total}}, \quad \forall i \in S \quad (8)$$

$$\ell_i^{\max} - \ell_i^{\min} \leq \ell_{\max}^{\text{diff}}, \quad \forall i \in S \quad (9)$$

$$y_p \in \{0, 1\}, \quad \forall p \in P \quad (10)$$

$$x_e \in \{0, 1\}, \quad \forall e \in E \quad (11)$$

$$\psi_i \in \{0, 1\}, \quad \forall i \in S \quad (12)$$

$$d_{ij} \in \{0, 1\}, \quad \forall i \in S, j \in U \quad (13)$$

Constraint (2) ensures that each ONU connects to a splitter via a path while (3) limits the maximum number of ONUs per splitter as well as setting the splitter usage variable ψ_i . The inequality in (4) ensures that all edges of used paths are marked used as well while constraints (5) and (6) sets the minimum and maximum network reach parameters for each splitter. Equation (7) activates the previous two constraints only when paths between commodities are used. Finally, the inequalities (8) and (9) implement the global PON fiber distance constraints.

Due to space limitations, the model in question incorporates only the fun-

damental constraints inherent to the PON planning problem. For a more complete model that includes refinements such as economies of scale, network coverage and non-symmetrical fiber costs, refer to [5].

4 Paths heuristic

It is evident that the set P will include an infeasible amount of paths for large graphs. However, as a first step, the number of paths can be reduced using a valid inequality.

Proposition 4.1 *Let $p^* \in P(k)$ be the shortest non-cyclic path between commodity pair $k \in K$ with length ℓ_{p^*} . For the PON planning problem as defined in section 2, the set of paths $Q(k) \subset P(k)$ will not be found in the minimal solution, where $\ell_q > (1 + c_t/c_f)\ell_{p^*}, \forall q \in Q(k)$.*

Proof. The cost of a shortest path $p^* \in P(k)$ with no fiber duct sharing is given by its trenching and fiber components, i.e. $c_{p^*} = \sum_{e \in E(p^*)} c_t \ell_e + c_f \ell_{p^*}$. It follows that any fiber duct sharing will result in a longer path p^+ with length ℓ_{p^+} . Let $E_s \subseteq E(p^*)$ be the edges path p^+ shares with other paths. Therefore, the total cost of path p^+ can be given by $c_{p^+} = \sum_{e \in E(p^+)} c_t \ell_e - \sum_{e \in E_s} c_t \ell_e + c_f \ell_{p^+}$. It is evident that if $c_{p^+} > c_{p^*}$, path p^+ will not be used in the minimal solution. Substituting and simplifying:

$$c_f \ell_{p^+} - \sum_{e \in E_s} c_t \ell_e > c_f \ell_{p^*} \quad (14)$$

The maximum sharing that can occur is if all edges are shared, i.e. $E_s = E(p^*)$. Furthermore, from the definition of a path, it is evident that $\ell_{p^*} = \sum_{e \in E(p^*)} \ell_e$. Substituting into equation (14), we arrive at $c_f \ell_{p^+} - c_t \ell_{p^*} > c_f \ell_{p^*}$, which can be rearranged to $\ell_{p^+} > (1 + c_t/c_f)\ell_{p^*}$. \square

Using this proposition, all paths that are c_t/c_f times longer than the shortest path p^* will not be calculated since they will not result in a minimal solution. In practice however, civil restrictions ensure even greater diminishing returns when deviating from the shortest path. This is due to the fact that trenches are made alongside roads which are usually the only access to customer premises.

To calculate these paths, a k shortest loopless path algorithm is used [2, 6]. These algorithms usually have time complexities that increase linearly with both k and $|\mathcal{V}|$, adding substantial preprocessing time as more paths are generated. It is therefore necessary to minimize the paths generated to

increase computational performance.

As an extreme, we will look at the performance when including only the shortest path ($k = 1$) in the model, effectively using opportunistic fiber duct sharing when shortest paths take a similar route. With $k = 1$, the much faster Dijkstra's algorithm can be used, increasing preprocessing performance.

5 Disintegration heuristic

Since real-world PON data are usually grouped into interconnected neighbourhoods, a disintegration of the input data into clusters should give good computational performance while staying close to the global minimal solution. As the central office is global to all clusters only splitter and ONU nodes are clustered, or the set $D = U \cup S$. The k means algorithm is used since it provides roughly equi-sized clusters which is useful for effective division of computational complexity. The algorithm provides the k output sets L_1, L_2, \dots, L_k , where $D = L_1 \cup L_2 \cup \dots \cup L_k$.

To ensure a feasible solution exists for each cluster, *valid clusters* are built from the output of the k means algorithm. These are clusters containing enough splitters to serve all ONUs contained within the cluster. The minimum split ratio is defined as the capacity of a splitter, κ .

Definition 5.1 (Valid cluster) *Given a set of splitters $S \neq \emptyset$, a set of ONUs $U \neq \emptyset$ and a minimum split ratio κ , a set $L \subseteq U \cup S, L \neq \emptyset$ is said to be a valid cluster only if*

$$\kappa|L \cap S| \geq |L \cap U|. \quad (15)$$

If a cluster L_i is invalid, it is combined with the next cluster and emptied, with $L_{i+1} = L_{i+1} \cup L_i, \forall i, 0 < i \leq k$. This process is repeated iteratively for each i until all clusters are valid.

6 Computational results

The model given above is implemented using C++ and IBM ILOG CPLEX Concert extensions. It is then solved on an Intel Core i7 at 2.67 GHz with 16 GiB main memory running Windows.

A single real-world GIS-mapped dataset containing 6,698 nodes and 7,660 edges, called *CityNet* [5], is solved and compared to a modified version of the *Branch Contracting Algorithm* (BCA) heuristic proposed in [4]. BCA is chosen

since this heuristic implements elements of fiber duct sharing through a tree-based clustering method. The authors claim performance of 10-15 % deviation from optimal with very fast computational speed. Since the original article does not specify values for Q , a grouping factor, the algorithm is run with all practical values for Q , i.e. $0 < Q \leq \kappa$, taking the minimum objective value. Next, since BCA is randomly initialized, it is run 20 times for each Q value, again noting the minimum objective value. This ensures a fair comparison with the proposed path and disintegration heuristics.

It should be noted that the last step of BCA requires a heuristic Steiner tree implementation to connect all splitters to the central office. Since the details of this heuristic are not clear from the original article, the algorithm is modified to connect splitters through shortest path routes to the central office, sharing fibers as possible. BCA without *any* connecting fibers between the central office and splitters (BCANoF) is also tested to ensure the modification does not produce biased results.

The model was solved incorporating the shortest path heuristic (1P) with no disintegration and a 1 hour time limit. Then, the clustering heuristic with 10 and 20 clusters were introduced and solved (1P+CL10 and 1P+CL20 respectively). The optimality gap (GAP) is specified for all scenarios where the branch-and-bound method is used. Peak memory usage in MiB during the tests is given by MEM_{peak} . If the total time to solve, t_{solve} , exceeds the time limit, an asterisk is placed next to the value. Finally, the number of splitters deployed is given in the SPs column.

Table 1
Numerical and computational results for the *CityNet* dataset

Instance	t_{solve} (s)	OBJ (mil R)	MEM_{peak}	GAP (%)	SPs
BCA	2.11	85.566	5,626.71	-	63
BCANoF	1.99	77.968	5,163.72	-	76
1P	3,685.42*	61.166	12,280.31	16.17	92
1P+CL10	212.77	58.746	507.88	0.00	86
1P+CL20	70.45	60.965	320.77	0.00	81

From the numerical results it is clear that the BCA heuristic is much faster than the clustering method, but produces an increase of 45.7 % in objective value. Similar objective values for 1P and 1P+CL indicates that the clustering does not introduce errors of more than the optimality gap of 16 %, although

the actual error margin may be much lower.

7 Conclusion

Given the numerical results, 1P and 1P+CL outperforms BCA by quite a margin, even when BCA is given the best possible chance, producing up to 45 % lower objective values. Time complexity wise, the heuristics dramatically reduce computation time, although BCA is still faster by an order of magnitude.

Overall, the heuristics proved to be very capable at solving the PON planning problem with high accuracy and with fast computation times. Unfortunately, worse than claimed performance for BCA suggests that it may be unsuitable for practical and inherently clustered datasets such as *CityNet*.

Following this research, a more connectivity-aware clustering method can be investigated to take advantage of the nature of the PON planning problem. Also, the estimation of the true distance from optimum would be interesting to determine the effectiveness of the 1P heuristic.

References

- [1] Effenberger, F., D. Clearly, O. Haran, G. Kramer, R. D. Li, M. Oron and T. Pfeiffer, *An introduction to pon technologies [topics in optical communications]*, Communications Magazine, IEEE **45** (2007), pp. S17 –S25.
- [2] Eppstein, D., *Finding the k shortest paths*, in: *Foundations of Computer Science, 1994 Proceedings.*, 35th Annual Symposium on, 1994, pp. 154 –165.
- [3] Li, J. and G. Shen, *Cost minimization planning for greenfield passive optical networks*, Optical Communications and Networking, IEEE/OSA Journal of **1** (2009), pp. 17 –29.
- [4] Mitcsenkov, A., G. Paksy and T. Cinkler, *Geography- and infrastructure-aware topology design methodology for broadband access networks (fttx)*, Photonic Network Communications **21** (2011), pp. 253–266.
- [5] van Loggerenberg, S., “Optimization of Passive Optical Network Planning for Fiber-to-the-Home Applications,” Master’s thesis, North West University Potchefstroom Campus (2012), submitted for examination.
- [6] Yen, J. Y., *Finding the k shortest loopless paths in a network*, Management Science **17** (1971), pp. pp. 712–716.

Load balancing optimization of telecommunication networks with two differentiated services

Amaro de Sousa^{a,b,1} Carlos Lopes^{a,1} Dorabella Santos^{a,1}

^a*Instituto de Telecomunicações, 3810-193 Aveiro, Portugal*

^b*DETI-Universidade de Aveiro, 3810-193 Aveiro, Portugal*

Abstract

The Differentiated Services architecture is a scalable solution to provide differentiated Quality of Service. In this paper, we address the network load balancing optimization of such networks based on bandwidth differentiation between two services. We define the optimization problem as an Integer Programming model and propose a heuristic algorithm based on GRASP with Path Relinking. We present computational results showing that (i) good quality solutions can be computed and (ii) proper load balancing can efficiently obtain service differentiation.

Keywords: Differentiated services, network load balancing, routing

1 Introduction

When the Differentiated Services (DiffServ) architecture was proposed by IETF [3], the aim was to enable network operators to provide services with differentiated Quality of Service (QoS), increasing in this way their revenues

¹ Email: asou@ua.pt, clopes@av.it.pt, dorabella@av.it.pt

with clients willing to pay more for better QoS. Due to the huge increase of bandwidth needs, service differentiation based on packet delay [4] and/or packet drop parameters [2,7] became less relevant, while differentiation based on bandwidth became more important [1].

Consider a network operator offering a Gold subscription (with a higher price) and a Standard subscription (with a lower price). The aim is to route flows through links whose loads are lower for Gold flows than for Standard flows. Consider a routing solution where the highest load of all links supporting either Gold or simultaneously Gold and Standard flows is μ_1 , while the highest load of all links supporting only Standard flows is μ_2 . If all flows grow at the same rate, the Gold traffic can grow up to $(1 - \mu_1)/\mu_1$ before a network link reaches 100% of its capacity, while the Standard traffic can only grow up to $(1 - \mu_2)/\mu_2$. Although both μ_1 and μ_2 should be as low as possible (to accommodate demand growth as much as possible), providing a routing solution where $\mu_1 < \mu_2$ differentiates the two services in the amount of additional bandwidth the network can provide, i.e., the operator guarantees that Gold clients always have more bandwidth than Standard clients and the amount of differentiation is tuned by the difference between μ_1 and μ_2 .

In Section 2, we define the optimization problem as an Integer Programming model, which allows to obtain good mathematical lower bounds. In Section 3, we propose a heuristic algorithm to compute good feasible solutions. In Section 4, we present computational results showing that the proposed algorithm is able to compute good quality solutions and that proper routing planning can efficiently achieve service differentiation.

2 Integer Programming Model

Consider a network given by a graph $G = (N, A)$, with node set N representing the network nodes and edge set A representing the network links. For each link $\{i, j\}$, $c_{\{ij\}}$ represents its demand capacity. Consider $V(i)$ as the set of nodes $j \in N$ such that arc $(i, j) \in A$. The network supports 2 services, indexed by $s = 1, 2$, where $s = 1$ represents the Gold service and $s = 2$ represents the Standard service. For each service s , the network supports a set of symmetric flows K_s , where each flow $k \in K_s$ is defined by its origin node o_{ks} , its destination node d_{ks} , and its average demand b_{ks} on each direction.

The routing optimization problem is defined with the following variables: x_{ij}^{ks} is a binary variable which indicates if arc (i, j) is in the path of flow $k \in K_s$; $z_{\{ij\}}^s$ is a binary variable which indicates if edge $\{i, j\}$ serves flows of service s ; $\mu_{\{ij\}}$ is a real variable indicating the load of edge $\{i, j\} \in A$; and μ_s is an

upper bound of the highest load among all edges that serve flows of service s .

Consider a parameter α ($0.0 < \alpha \leq 1.0$) which gives the maximum allowed difference between the highest link loads of the 2 services. Consider also that each flow is routed on a single routing path (unsplittable routing). Consider the vector $\boldsymbol{\mu} = \{\mu_1, \mu_2\}$. The Integer Programming Model defining the routing optimization problem is given by:

$$\text{lexmin } \boldsymbol{\mu} \quad (1)$$

s.t.

$$\sum_{j \in V(i)} (x_{ij}^{ks} - x_{ji}^{ks}) = \begin{cases} 1, & i = o_{ks} \\ 0, & i \neq o_{ks}, d_{ks} \\ -1, & i = d_{ks} \end{cases} \quad i \in N, s = 1, 2, k \in K_s \quad (2)$$

$$\sum_{s=1}^2 \sum_{k \in K_s} b_{ks} (x_{ij}^{ks} + x_{ji}^{ks}) = c_{\{ij\}} \mu_{\{ij\}} \quad \{i, j\} \in A \quad (3)$$

$$x_{ij}^{ks} + x_{ji}^{ks} \leq z_{\{ij\}}^s \quad \{i, j\} \in A, s = 1, 2, k \in K_s \quad (4)$$

$$\mu_s \geq \mu_{\{ij\}} + (z_{\{ij\}}^s - 1) \quad \{i, j\} \in A, s = 1, 2 \quad (5)$$

$$\mu_2 \leq \mu_1 + \alpha \quad (6)$$

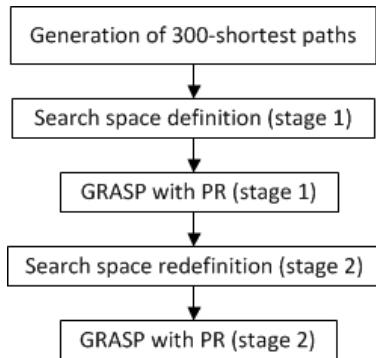
$$x_{ij}^{ks} \in \{0, 1\}, z_{\{ij\}}^s \in \{0, 1\}, \mu_{\{ij\}} \in [0.0, 1.0], \mu_s \in [0.0, 1.0] \quad (7)$$

where **lexmin** in the objective function (1) denotes the lexicographical minimization, i.e., finding the vector $\boldsymbol{\mu}^*$ which is lexicographically minimal among all possible vectors $\boldsymbol{\mu}$. Constraints (2) are the usual path conservation constraints. Constraints (3) guarantee that variables $\mu_{\{ij\}}$ are set with each link load. Constraints (4) guarantee that variable $z_{\{ij\}}^s$ is one if link $\{i, j\}$ supports at least one flow of service s . Constraints (5) impose that the worst link load μ_s of service s must be at least $\mu_{\{ij\}}$ if link $\{i, j\}$ supports service s . Constraint (6) guarantee the maximum allowed difference between μ_1 and μ_2 .

The optimal solution can be obtained by solving two mixed integer linear programming models in sequence: first, minimize μ_1 , subject to constraints (2-7), finding the optimal value μ_1^* ; then, minimize μ_2 , subject to constraints (2-7) and the additional constraint $\mu_1 \leq \mu_1^*$, finding the optimal value μ_2^* . In practice, these models cannot be solved for reasonable sized instances due to exaggerate CPU times. Nevertheless, it is possible to compute good lower bounds by considering the relaxation of variables x_{ij}^{ks} and maintaining the integrality constraints of variables $z_{\{ij\}}^s$. These lower bounds are used in the computational results to evaluate the solutions of the heuristic algorithm.

3 Heuristic Algorithm

The heuristic algorithm is based on Greedy Randomized Adaptive Search Procedure (GRASP) with Path Relinking (PR). GRASP, first introduced in [5], is a multi-start local search method where, at each start, an initial solution is randomly generated (with a greedy randomized procedure) and local search is applied to it to find its closest local optimum. At the end, the best among all local optimum solutions is the result of the algorithm. PR was originally proposed as an intensification method, applied to tabu search [6], that tries to find better solutions by the combination of two initial solutions. The common combination of GRASP with PR is to run Path Relinking at the end of each GRASP iteration between the local optimum solution and a randomly selected solution from a list of elite solutions (for a good survey, see [9]). The straightforward application of GRASP with PR to our problem is not efficient because it is hard to get solutions with good values of μ_1 when both μ_1 and μ_2 are simultaneously optimized. We propose two GRASP with PR stages: stage 1 only concerned with minimizing μ_1 and stage 2 concerned with minimizing both μ_1 and μ_2 . The complete algorithm has the following five steps:



In the first step, the algorithm determines the 300-shortest paths for each flow [8], assuming that the path length is given by its number of hops in G .

In the second step, a search space is defined. This is done by: (i) running B times a greedy randomized procedure using all 300-shortest paths of each flow; (ii) ordering these greedy solutions from the best to the worst; (iii) selecting the smallest number of best solutions, containing a total number of different paths that, divided by the number of flows, is not lower than n ; and (iv) defining the search space with the paths contained in the selected solutions.

In the third step, we run GRASP with PR. In the local search of GRASP, we move to a neighbor solution only if it improves μ_1 and does not violate α .

In the fourth step, we fix the previous selected routing paths for the Gold flows and we compute the routing paths of the Standard flows to be included in the search space in a similar way to the second step with the following difference: we consider that the Gold flows are routed in the paths of the best solution found in the previous step, and the greedy randomized procedure is applied only to the Standard flows.

The fifth step is similar to the third step but now in the local search, we move to a neighbor not only when μ_1 improves but also when μ_1 is the same and μ_2 improves (provided that the solution does not violate α).

The GRASP local search (in both second and fourth steps) considers the neighbor set of a current solution given by all solutions that differ from the current one on the path of a single flow. Concerning the stopping criteria, in the third and fifth steps, we stop the GRASP with PR if the algorithm is T seconds without improving the incumbent. Note that B and n are parameters of the algorithm: B controls the quality of the greedy solutions whose paths are selected to define the search spaces and n controls the size of the search spaces given to the GRASP with PR.

The greedy randomized procedure used in stage 1 (second step and GRASP of third step) is as follows. First, we consider an empty network with no routed flows and select a random order for all flows of both services. Then, for each flow and by the previously selected order, we (i) compute the path that provides the best μ_1 not violating α (ii) route the flow through the selected path and (iii) compute the resulting link loads. The greedy randomized procedure used in GRASP of the fifth step has the following difference: the selected path is the one that provides the best μ_1 and the best μ_2 not violating α .

4 Computational Results

We have considered 6 test instances based on the NSF topology with 26 nodes and 42 links (Fig. 1). The first three instances (NSF100 to NSF102) consider all links with the same capacity. The second three instances (NSF'100 to NSF'102) consider links with different capacities (in Fig. 1, thick lines represent links with ten times more capacity than thin lines). In all instances, we have randomly generated flow demand matrices considering some traffic nodes (white nodes in Fig. 1) while the remaining are transit nodes. Demand values were generated in a way that the percentage of Gold demand decreases from the 102 instances to the 101 instances and from these to the 100 instances. Table 1 shows the lower bounds for $\alpha = 1.0$ and $\alpha = 0.3$ (see Section 2) and their CPU times, obtained with CPLEX 12.1 running on a standard PC.

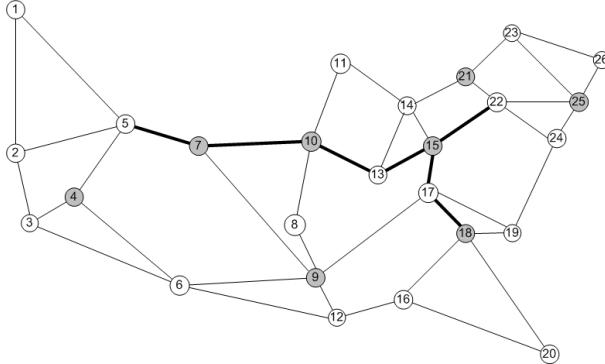


Fig. 1. NSF network used in all test instances.

	$\alpha = 1$			$\alpha = 0.3$		
Instance	μ_1 (%)	μ_2 (%)	Time (sec)	μ_1 (%)	μ_2 (%)	Time (sec)
NSF100	9.94	67.18	6895	13.65	35.99	20253
NSF101	12.70	65.99	31105	14.79	37.03	21388
NSF102	18.21	46.93	28628	18.21	46.93	28628
NSF'100	8.43	99.99	4328	12.20	40.95	13344
NSF'101	8.56	90.79	3594	12.03	42.32	21989
NSF'102	15.05	48.47	5715	15.33	39.45	18352

Table 1
Lower bounds and CPU times

We have run the proposed heuristic for all test instances with $T = 2$ minutes and parameter values $B = 600, 1200, 2000$ and $n = 4, 8, 12$. We have run the heuristic five times for each test instance and each parameter combination. Tables 2 (for $\alpha = 1.0$) and 3 (for $\alpha = 0.3$) show the values of μ_1 and μ_2 of the best solutions found for each instance, together with the CPU time and the parameter values that have computed them. Column 'Dif.' gives the difference between the μ_1 value of the worst solution found among all runs and the best μ_1 value. This difference is at most 3.11% (much lower in most cases) showing that the quality of the solutions between different runs does not vary too much and that the proposed heuristic presents similar efficiency for the different parameter values. Note that the best solutions (Tables 2 and 3) are fairly close to the lower bound values (Table 1) in all cases showing that the heuristic algorithm is able to compute good quality solutions for all tested instances.

Instance	μ_1 (%)	μ_2 (%)	Time (sec)	B	n	Dif.
NSF100	10.02	70.55	579.1	600	8	1.29
NSF101	12.73	66.42	588.8	2000	12	1.02
NSF102	18.21	48.03	364.7	600	8	0.92
NSF'100	8.51	99.79	461.0	2000	8	3.11
NSF'101	8.59	90.77	333.4	2000	12	1.81
NSF'102	16.90	48.63	628.6	1200	4	1.00

Table 2
Heuristic algorithm results ($\alpha = 1.0$)

Instance	μ_1 (%)	μ_2 (%)	Time (sec)	B	n	Dif.
NSF100	13.65	36.21	350.5	2000	12	1.98
NSF101	14.80	37.23	527.2	1200	12	2.04
NSF102	18.48	48.38	601.4	1200	8	0.93
NSF'100	12.28	40.80	394.2	2000	8	2.59
NSF'101	12.15	41.87	651.1	1200	4	2.04
NSF'102	17.01	43.69	440.3	600	8	1.36

Table 3
Heuristic algorithm results ($\alpha = 0.3$)

In order to illustrate the differentiated services concept, let us consider instance NSF100 (it has 73 Gold flows and 153 Standard flows). With the best solutions, we computed for each flow, its worst route load defined as the highest link load among all links of its routing path. Fig. 2 presents the worst route loads of all flows of each service in a decreasing order. In the left solution, all Gold flows have a maximum worst route load of 10.02%, which means that all Gold flows can grow up to $(1 - 0.1002)/0.1002 = 898\%$ before suffering network congestion. On the other end, most of the Standard flows have worst route loads of 70.25%, which means that these flows can only grow up to $(1 - 0.7025)/0.7025 = 42.3\%$ before suffering network congestion. The right solution illustrates how α can be used to tune the service differentiation. Using $\alpha = 0.3$, the worst route load of Gold flows has only slightly increased from 10.02% to 13.65%, while the worst route load of Standard flows has significantly decreased from 70.25% to 36.21% (which enables Standard flows to grow up to 176% before suffering network congestion).

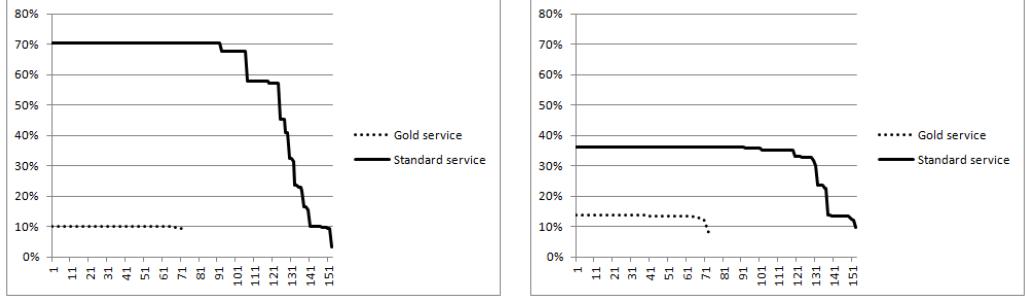


Fig. 2. Worst route loads for NSF100 with $\alpha = 1.0$ (left) and $\alpha = 0.3$ (right).

References

- [1] "MPLS Traffic Engineering - DiffServ Aware (DS-TE)", CISCO Technical Document, 2007.
- [2] Aweya, J., M. Ouellette, and D. Montuno, *Proportional loss rate differentiation in a FIFO queue*, Computer Communications **27**(18) (2004), 1851–1867.
- [3] Blake, S., D. Black, M. Carlson, E. Davies, Z.Wang, and W.Weiss, "An Architecture for Differentiated Services", IETF RFC 2475, 1998.
- [4] Dovrolis, C., D. Stiliadis, and P. Ramanathan, *Proportional Differentiated Services: Delay Differentiation and Packet Scheduling*, IEEE/ACM Trans. On Networking **10**(1) (2002), 12–26.
- [5] Feo, T. and M. Resende, *A probabilistic heuristic for a computationally difficult set covering problem*, Operations Research Letters **8** (1989), 67–71.
- [6] Glover, F., "Tabu search and adaptive memory programming - Advances, applications and challenges", in Interfaces in Computer Science and Operations Research, R.S. Barr, R.V. Helgason, and J.L. Kennington, (Eds.), Kluwer, 1–75, 1996.
- [7] Huang, Y. and R. Guérin, *A simple FIFO-based scheme for differentiated loss guarantees*, Computer Networks **51**(4) (2004), 1133–1150.
- [8] Martins, E., M. Pascoal, J. Santos, "A New Algorithm for Ranking Loopless Paths", Technical Report, CISUC, University of Coimbra, Portugal, 1997.
- [9] Resende, M. and C. Ribeiro, "GRASP with path relinking: recent advances and applications", in Metaheuristics: Progress as Real Problem Solvers, T. Ibaraki, K. Nonobe and M. Yagiura, (Eds.), Springer, 29–63, 2005.

On the Two-Architecture Connected Facility Location Problem

Markus Leitner^{a,1} Ivana Ljubić^{b,2} Markus Sinnl^{b,3}
Axel Werner^{c,4}

^a *Institute of Computer Graphics and Algorithms, Vienna University of Technology, Vienna, Austria*

^b *Department of Statistics and Operations Research, University of Vienna, Vienna, Austria*

^c *Zuse Institute Berlin, Berlin, Germany*

Abstract

We introduce a new variant of the connected facility location problem that allows for modeling mixed deployment strategies (FTTC/FTTB/FTTH) in the design of local access telecommunication networks. Several mixed integer programming models and valid inequalities are presented. Computational studies on realistic instances from three towns in Germany are provided.

Keywords: Connected Facility Location, Branch-and-Cut, FTTx Deployment

¹ Supported by the Austrian Science Fund (FWF) under grant I892-N23. Email: leitner@ads.tuwien.ac.at

² Supported by the APART Fellowship of the Austrian Academy of Sciences. Email: ivana.ljubic@univie.ac.at

³ Email: markus.sinnl@univie.ac.at

⁴ Supported by the German Research Foundation (DFG). Email: werner@zib.de

1 Introduction and Problem Definition

In the design of local access networks three main scenarios (deployment *architectures*) are considered: (i) “fiber-to-the-home” (FTTH), (ii) “fiber-to-the-building” (FTTB), and (iii) “fiber-to-the-curb” (FTTC). From an optimization point of view – abstracting from the more technical details and considering mainly topology decisions – FTTH deployment is modeled using variants of the Steiner tree problem [2,5], and FTTB or FTTC deployments are modeled as connected facility location (ConFL) [1,3,4]. In this paper we consider a new modeling and optimization approach for the *mixed deployment* which is motivated by the fact that in urban areas the lowest investment costs and the best bandwidth rates are achieved with a deployment that mixes FTTH and FTTC/FTTB. The main drawback of existing approaches is that they do not allow for the design of such a combined deployment. To overcome this, we propose to model the mixed deployment as *ConFL with two architectures*, which will be denoted by 2-ArchConFL. We consider two different architectures 1 and 2 (these could be FTTB and FTTC, or two FTTC quality-of-service levels) with associated minimum coverage rates, p_1 and p_2 . The presented model can be easily generalized to more than two architectures, thus incorporating more deployment strategies, such as “fiber-to-the-air” (FTTA), if necessary.

More precisely, we are given a bipartite *assignment graph* between potential *facilities*, representing locations where equipment can be installed, and *customers*. Two types of facilities – one for each architecture – exist and give rise to two types of *assignment arcs* directed from facilities to customers. Each customer can be supplied by at most one facility and each supplying facility has to be opened in order to serve customers. In addition, each open facility must be connected to one of the *central offices*, via a path in the *core graph*. The (undirected) core graph consists of facilities, central offices and potential *Steiner nodes*, and its edges correspond to segments along which fibers can be laid out. See Figure 1(a) for an example.

The goal is to serve certain fractions of customers (determined according to *minimum coverage rates*) by each architecture while minimizing total cost.

Formally, the problem is described by a directed graph $G = (V, A)$ where the node set V is the disjoint union of (i) potential central offices (COs) Q with opening costs $c_q \geq 0, \forall q \in Q$, (ii) customer nodes C with demands $d_c \in \mathbb{N}, \forall c \in C$, (iii) potential facility locations $F = F^1 \cup F^2$ with opening costs $c_i^l \geq 0, \forall i \in F^l, l = 1, 2$, and (iv) potential Steiner nodes S . Hereby, potential facilities in F^l represent locations where equipment can be installed to connect customers using architecture l ; note that F^1 and F^2 need not be

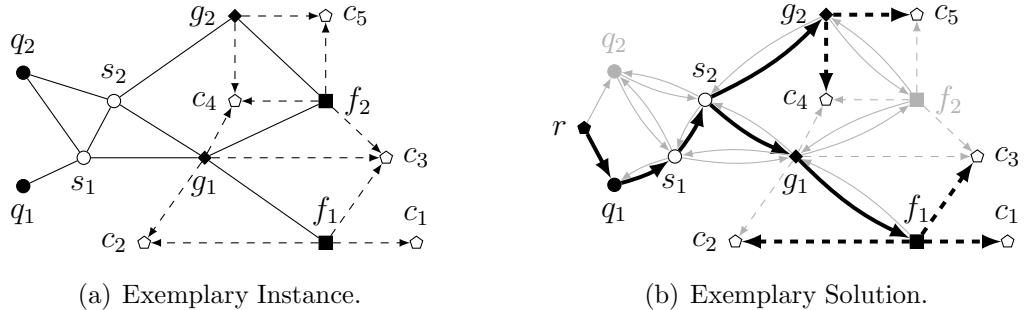


Fig. 1. (a) An exemplary instance with potential central offices q_1, q_2 , type-1 facilities f_1, f_2 , type-2 facilities g_1, g_2 , potential Steiner nodes s_1, s_2 , and customers c_1, \dots, c_5 ; assignment arcs are dashed. (b) A solution, including the root node, with selected CO q_1 supplying c_1, c_2 , and c_3 using technology 1 via facility f_1 , and c_4 and c_5 using technology 2 via facility g_2 ; note that g_1 is used as a Steiner node.

disjoint. The arc set A consists of (i) the core arcs $A_c = \{(i, j) \in A \mid i, j \notin C\}$, corresponding to forward and backward arcs for each edge of the core graph, with trenching costs $c_a \geq 0, \forall a \in A_c$, and (ii) assignment arcs $A^l = \{(i, j) \in A \mid i \in F^l, j \in C\}$, for each architecture $l = 1, 2$. Each potential assignment $(i, j) \in A^l$ is associated with costs $c_{ij}^l \geq 0$ for connecting customer j to facility i using architecture l . Finally, minimum coverage rates p_1 and p_2 are given with $0 \leq p_1 \leq p_2 \leq 1$, specifying the minimal fraction of total demand $D := \sum_{j \in C} d_j$ that must be satisfied by each architecture. Hereby we assume architecture 1 to be preferable to architecture 2, so that a coverage rate of p_2 means that $p_2 \cdot 100\%$ of the total demand needs to be satisfied by either architecture 1 or 2.

The total cost of a solution is the sum of all opening costs of used COs, trenching costs for used core arcs, assignment costs for the realized customer assignments, and opening costs of selected facilities. Note that CO nodes and facility locations can be used as Steiner nodes, in which case no opening costs are paid for passing through them. Furthermore, due to non-negative edge costs, there always exists an optimal solution which is a forest, or even – in case only a single CO is open – a tree. For an example of a feasible solution, see Figure 1(b).

2 Integer Linear Programming Models

For the above stated 2-ArchConFL problem integer linear programs (ILP) can be formulated; we explicitly present cut formulations here, but note that

also other models, comprising flow or subtour elimination constraints, can be devised, as for the classical ConFL problem (cf. [3]).

For modeling purposes, we extend the graph G with an artificial root node $r \notin V$ connected via artificial arcs $A_r = \{(r, q) \mid q \in Q\}$ to all central offices (cf. Figure 1(b)). Their purpose is to select one or more COs to open and to incorporate their costs into the model: for each artificial arc (r, q) , $q \in Q$, we set $c_{rq} := c_q$. Obviously, if $|Q| = 1$, i.e., there is only one potential CO node, creation of the root and artificial arcs can be skipped and the CO itself can act as the root. For abbreviation we use $A_{rc} := A_r \cup A_c$.

In the following subsections, we present ILP models based on various directed cutset constraints. We denote by $F_j^l = \{i \in F^l \mid (i, j) \in A^l\}$ the set of eligible facilities for a customer $j \in C$ for $l = 1, 2$. Then the set of common decision variables for all the models is as follows: (i) core arc variables $x_{ij} \in \{0, 1\}, \forall (i, j) \in A_{rc}$ indicate whether or not core/artificial arc (i, j) is used, (ii) assignment arc variables $x_{ij}^l \in \{0, 1\}, \forall (i, j) \in A^l, l = 1, 2$ indicate if customer j is supplied by facility i using architecture l , (iii) facility variables $y_i^l \in \{0, 1\}, \forall i \in F^l, l = 1, 2$ indicate whether or not facility i is open providing connections using architecture l , and (iv) customer variables $z_j^l \in \{0, 1\}, \forall j \in C, l = 1, 2$ indicate if customer j is connected using architecture l . For a given node set $W \subset V$, let $\delta^-(W) = \{(i, j) \in A \cup A_r \mid i \notin W, j \in W\}$ be the set of ingoing arcs in G . For an arc set $\hat{A} \subseteq A \cup A_r$ we use $x(\hat{A}) := \sum_{(i,j) \in \hat{A} \cap A_{rc}} x_{ij}$, as well as $x^l(\hat{A}) := \sum_{(i,j) \in \hat{A} \cap A^l} x_{ij}^l$ and $(x + x^l)(\hat{A}) := x(\hat{A}) + x^l(\hat{A})$ for $l = 1, 2$.

Basic model

Using the previously described variables, we can formulate 2-ArchConFL as model (yC) given by (1)–(7).

$$\min \quad \sum_{(i,j) \in A_{rc}} c_{ij} x_{ij} + \sum_{l=1}^2 \sum_{(i,j) \in A^l} c_{ij}^l x_{ij}^l + \sum_{l=1}^2 \sum_{i \in F^l} c_i^l y_i^l \quad (1)$$

$$\text{s.t.} \quad \sum_{l=1}^2 z_j^l \leq 1 \quad \forall j \in C \quad (2)$$

$$\sum_{i \in F_j^l} x_{ij}^l = z_j^l \quad \forall j \in C, l = 1, 2 \quad (3)$$

$$x_{ij}^l \leq y_i^l \quad \forall j \in C, i \in F_j^l, l = 1, 2 \quad (4)$$

$$\sum_{\lambda=1}^l \sum_{j \in C} d_j z_j^\lambda \geq \lceil p_l D \rceil \quad l = 1, 2 \quad (5)$$

$$x(\delta^-(W)) \geq y_i^l \quad \forall W \subseteq V \setminus C, i \in F^l \cap W, l = 1, 2 \quad (6)$$

$$(\mathbf{x}, \mathbf{x}^1, \mathbf{x}^2, \mathbf{y}^1, \mathbf{y}^2, \mathbf{z}^1, \mathbf{z}^2) \in \{0, 1\}^{|A| + |A^1| + |A^2| + |F^1| + |F^2| + 2|C|} \quad (7)$$

Constraints (2) and (3) ensure that each connected customer uses a unique architecture and assignment arc; if $p_2 = 1$, Inequality (2) can be replaced by equality. Constraints (4) force a facility to be opened whenever an assignment arc issuing from it is chosen. Demanded coverage rates are satisfied due to Constraints (5). Finally, the connectivity constraints given by (6) (*y-cuts*) ensure that each opened facility is connected to the root node via opened core arcs. Since the root node is adjacent only to the CO nodes, at least one CO is opened in the solution. Hence (yC) is a valid model for 2-ArchConFL.

Note that the left-hand side matrix $M = (a_{ij})_{1 \leq i \leq 2|C| + |A_1 \cup A_2|, 1 \leq j \leq |A_1 \cup A_2|}$ defined by (3) and (4) has the following structure:

$$\left(\begin{array}{cccccc} 1 & 1 & \dots & 1 & & \\ & \ddots & & & & \\ & & & 1 & \dots & 1 \end{array} \right) \quad \left. \begin{array}{c} \\ \\ \end{array} \right\} 2|C|$$

I

$$\left. \begin{array}{c} \\ \\ \end{array} \right\} |A^1 \cup A^2|$$

Here \mathbf{I} denotes the unit matrix of size $|A^1 \cup A^2|$. Observe that each column of this 0/1-matrix contains exactly two nonzero entries; consider the partition (M_1, M_2) of its rows where M_1 contains the first $2|C|$ rows. Then for each column j we have $\sum_{i \in M_1} a_{ij} - \sum_{i \in M_2} a_{ij} = 0$. Hence M is totally unimodular and the integrality of the assignment variables can be relaxed to $x_{ij}^l \in [0, 1]$.

zI-cuts

If some customer j is connected using architecture l , any cut between j and the root node must contain either a core arc or an assignment arc for l . Thus, the model can be strengthened by replacing the y-cuts (6) by z_l -cuts:

$$(x + x^l)(\delta^-(W)) \geq z_j^l \quad \forall W \subseteq V, j \in C \cap W, l = 1, 2 \quad (8)$$

If $|W \cap C| = 1$, we can reformulate (8) using (3) to obtain the following inequalities, which dominate (8):

$$x(\delta^-(W)) \geq \sum_{i \in W \cap F_j^l} x_{ij}^l \quad \forall W \subseteq V, \ C \cap W = \{j\}, \ l = 1, 2 \quad (9)$$

z-cuts

Similarly, if customer j is connected with any architecture then some core or assignment arc must be selected, which gives the *z-cuts*:

$$(x + x^1 + x^2)(\delta^-(W)) \geq z_j^1 + z_j^2 \quad \forall W \subseteq V, j \in C \cap W \quad (10)$$

As for the *zl-cuts*, if $|W \cap C| = 1$, we obtain the dominating inequalities

$$x(\delta^-(W)) \geq \sum_{l=1}^2 \sum_{i \in W \cap F_j^l} x_{ij}^l \quad \forall W \subseteq V, C \cap W = \{j\} \quad (11)$$

In the following, we refer by (zLC) and (zC), to model (yC) with (6) replaced by (9) and (11), respectively. We denote by $v_{LP}(X)$ the optimum objective value of the LP relaxation of MIP model (X). Then the following can be shown (in a similar way as in [3]):

Proposition 2.1 $v_{LP}(zC) \geq v_{LP}(zLC) \geq v_{LP}(yC)$, and there exist instances for which strict inequality holds for both inequalities. Furthermore, the integrality gap of (yC) is in $\Omega(|V|)$.

3 Computational Results

To assess our models, branch-and-cut approaches have been implemented in C++ using IBM CPLEX 12.4 and tested on instances based on realistic networks representing deployment areas of three German towns. Table 1 gives further details on the instances. For each of the three given network topologies, 20 and 40 different instances are generated by varying the allowed sets of facilities and assignment arcs. We applied an absolute time limit of 7200 CPU-seconds to all experiments which have been performed on a single core of an Intel Xeon processor with 2.53 GHz using at most 3GB RAM. We compared the computational performance of (yC), (zLC), and (zC) models, and also considered variant (yzC) where z-cuts are separated if no further violated y-cuts exist. The underlying branch-and-cut implementations follow the main ideas given in [3]. For each instance and cut strategy, nine combinations of (percentage) coverage rates are considered: $(p_1, p_2) \in \{(20, 60), (40, 60), (20, 80), (40, 80), (60, 80), (20, 100), (40, 100), (60, 100), (80, 100)\}$.

Figure 2 shows box plots of the runtimes of all computations for each network w.r.t. the different cut strategies. Each column corresponds to 9 coverage settings for each instance, i.e., we have 180 (for `berlin-tu` and `atlantis`) and

Network	# Instances	$ V \setminus C $	$ C $	$ F $	$ A_{rc} $	$ A^1 \cup A^2 $
berlin-tu	20	384	39	55–109	1124	84–269
atlantis	20	1001	345	361–447	2062	851–2952
vehlefanz	40	895	238	273–407	2197	544–3749

Table 1
Overview of test instances.

360 computations for **vehlefanz**. The numbers on top of each column indicate in how many computations the time limit was hit. In general, the y-cuts appear to be preferable over z- and zl-cuts. For the smaller network the z-cuts show a slightly better performance – this might be due to the fact that these instances significantly differ from the others with respect to the ratio of the number of customer nodes to the total number of nodes.

Figure 3 shows the influence of different coverage rates on the computational performance. Each column contains results of $20+20+40$ computations over all instances, for a fixed coverage pair. Here the (yzC) cut strategy is considered, since this seems to be the best compromise between (yC) and (zC), considering all instance types. As can be seen from the three sections of the plot, increasing p_1 while keeping the values of p_2 fixed, yields a significant reduction in CPU-time. The picture is not that clear if p_1 is kept fixed and p_2 is increased (different greyscale levels): While for $p_1 = 20\%$ CPU-time decreases with higher p_2 , no clear trend can be derived for $p_1 = 40\%$ and $p_1 = 60\%$.

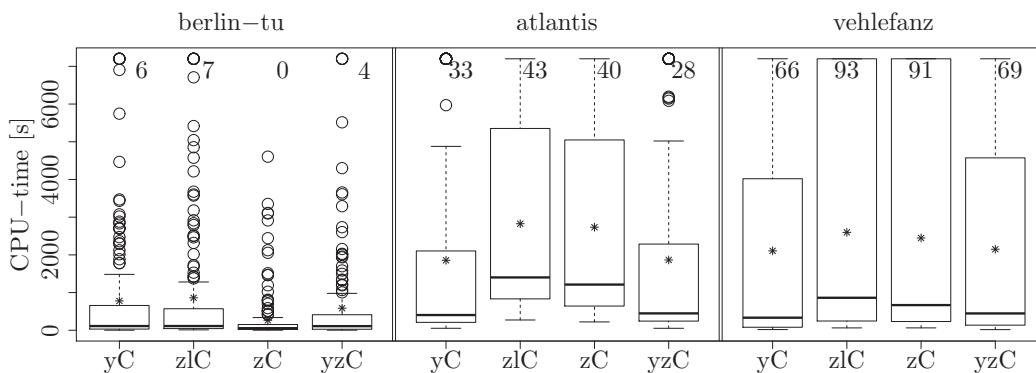


Fig. 2. CPU-time per instance for different cut strategies.

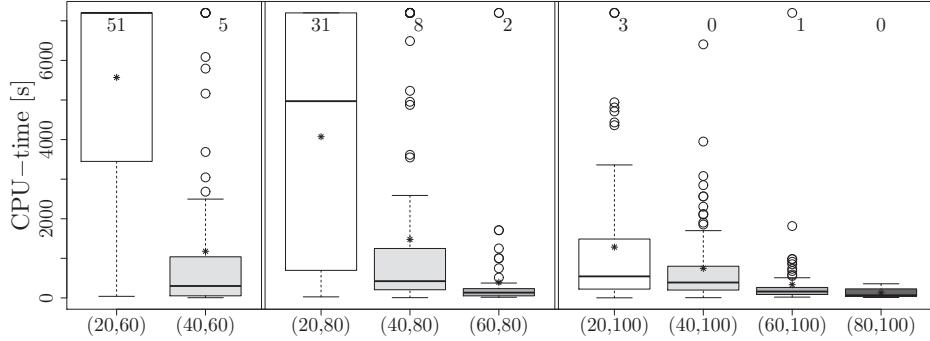


Fig. 3. CPU-time for different coverage rates.

4 Conclusions and Outlook

A new variant of the connected facility location problem has been introduced and a MIP model with cut inequalities has been presented and computationally tested on a set of realistic instances. For future studies, other valid inequalities and formulations for the problem are conceivable, such as variants of cover cuts, and Miller-Tucker-Zemlin or common flow formulations.

References

- [1] Contreras, I. and E. Fernández, *General network design: A unified view of combined location and network design problems*, European Journal of Operational Research **219** (2012), pp. 680–697.
- [2] da Cunha, A. S., A. Lucena, N. Maculan and M. G. C. Resende, *A relax-and-cut algorithm for the prize-collecting Steiner problem in graph*, Discrete Applied Mathematics **157** (2009), pp. 1198–1217.
- [3] Gollowitzer, S. and I. Ljubić, *MIP models for connected facility location: A theoretical and computational study*, Computers & Operations Research **38** (2011), pp. 435–449.
- [4] Leitner, M. and G. R. Raidl, *Branch-and-cut-and-price for capacitated connected facility location*, Journal of Mathematical Modelling and Algorithms **10** (2011), pp. 245–267.
- [5] Ljubić, I., R. Weiskircher, U. Pferschy, G. Klau, P. Mutzel and M. Fischetti, *An algorithmic framework for the exact solution of the prize-collecting Steiner tree problem*, Mathematical Programming, Series B **105** (2006), pp. 427–449.

Benders Decomposition for a Location-Design Problem in Green Wireless Local Area Networks

Bernard Gendron^a Rosario G. Garroppo^b
Gianfranco Nencioni^b Maria Grazia Scutellà^c Luca Tavanti^b

^a *CIRRELT and DIRO, Université de Montréal, Montréal, Canada*

^b *Dipartimento di Ingegneria dell'Informazione, Università degli Studi di Pisa,
Pisa, Italy*

^c *Dipartimento di Informatica, Università degli Studi di Pisa, Pisa, Italy*

Abstract

We consider a problem arising in the design of green (or energy-saving) wireless local area networks (GWLANs). Decisions on both location and capacity dimensioning must be taken simultaneously. We model the problem as an integer program with nonlinear constraints and derive valid inequalities. We handle the nonlinearity of the formulation by developing a Benders decomposition algorithm. We propose various ways to improve the Benders master problem and the feasibility cuts.

Keywords: Green wireless local area network, Facility location, Network design, Benders decomposition.

1 Introduction

We consider a problem arising in the design of green (or energy-saving) wireless local area networks (GWLANs). The network structure is bipartite, with a

set of access points (APs) that must be assigned to user terminals (UTs) to satisfy their demands while minimizing the overall power consumption of the APs. Each UT must be assigned to one AP (single assignment constraints). Each AP can have different power levels (PLs) and be powered off, but at most one PL must be chosen for each AP. The PL assigned to a particular AP affects the capacity of the connections between this AP and the UTs assigned to it. The decisions are therefore to use (open) or not each AP, to assign a PL (capacity) to each open AP and to assign exactly one open AP to each UT. Therefore, the problem can be seen as a discrete location problem, where the capacity to assign to each location also has to be decided (this is the design part of the problem). Hence, we see this problem as a particular case of a broader class of location-design problems, where both the location and capacity dimensioning decisions must be taken.

To fully describe the problem, we use the following notation:

- \mathcal{J} : set of access points (APs);
- \mathcal{I} : set of user terminals (UTs);
- \mathcal{K} : set of power levels (PLs) for each AP;
- p_0 : power consumed when powering on any AP;
- p_k : power consumed when using any AP at PL k ;
- w_i : demand of UT i ;
- $r_{ij}(\pi_j)$: transmission rate (capacity) between AP j and UT i , as a function of π_j , the power consumed at AP j .

We assume that $r_{ij}(\pi_j) \geq 0$ is a nondecreasing function that satisfies the following conditions, for each pair of AP j and UT i :

- There exists a threshold $\gamma_{ij} \geq 0$ such that $r_{ij}(\pi_j) = 0$ if $\pi_j \leq \gamma_{ij}$ and $r_{ij}(\pi_j) > 0$ whenever $\pi_j > \gamma_{ij}$. Thus, if $\gamma_{ij} > 0$, then AP j cannot be assigned to UT i as long as its power consumption π_j remains below γ_{ij} . The particular case $\gamma_{ij} = 0$ represents a situation where transmission can occur between AP j and UT i , whenever j is powered on.
- $r_{ij}(\pi_j) \leq r_{max}$ for any π_j , where r_{max} is typically the maximum rate achievable by any physical link.

Although our approach can be used if $r_{ij}(\pi_j)$ is linear, it is designed to address the inherent nonlinearity of $r_{ij}(\pi_j)$ encountered in practical applications. In

our context, the function has the following form, where $\gamma_{ij} > 0$:

$$(1) \quad r_{ij}(\pi_j) = \begin{cases} 0, & \text{if } \pi_j \leq \gamma_{ij}, \\ \min\{\alpha_{ij}\pi_j, r_{max}\}, & \text{otherwise,} \end{cases}$$

where α_{ij} is a transmission loss factor between AP j and UT i .

One approach to solve the problem would consist in modeling the problem as an integer program (IP) and then linearize $r_{ij}(\pi_j)$. Another approach, which we study in this paper, is to use an IP model within a Benders decomposition framework that handles explicitly the nonlinear function $r_{ij}(\pi_j)$. In Section 2, we present the IP model, as well as valid inequalities. The Benders decomposition method is described in Section 3. Computational results from experiments on randomly generated instances are reported in Section 4.

2 Model and Valid Inequalities

We define the following sets of binary variables:

- x_{ij} : 1 if AP j is assigned to UT i , 0 otherwise;
- y_{jk} , 1 if AP j is assigned PL k , 0 otherwise.

The model can then be written as follows:

$$(2) \quad z = \min \sum_{j \in \mathcal{J}} \sum_{k \in \mathcal{K}} (p_0 + p_k) y_{jk}$$

$$(3) \quad \sum_{j \in \mathcal{J}} x_{ij} = 1, \quad i \in \mathcal{I},$$

$$(4) \quad \sum_{k \in \mathcal{K}} y_{jk} \leq 1, \quad j \in \mathcal{J},$$

$$(5) \quad x_{ij} \leq \sum_{k \in \mathcal{K}} y_{jk}, \quad i \in \mathcal{I}, j \in \mathcal{J},$$

$$(6) \quad \sum_{i \in \mathcal{I}} \frac{w_i x_{ij}}{r_{ij}(\pi_j)} \leq 1, \quad j \in \mathcal{J},$$

$$(7) \quad x_{ij} \in \{0, 1\}, \quad i \in \mathcal{I}, j \in \mathcal{J},$$

$$(8) \quad y_{jk} \in \{0, 1\}, \quad j \in \mathcal{J}, k \in \mathcal{K},$$

where $\pi_j \equiv \sum_{k \in \mathcal{K}} p_k y_{jk}$ for any $j \in \mathcal{J}$.

The objective (2) is to minimize the total power consumption, from powering on each AP and from using the different power levels. Equations (3) are the single assignment constraints that impose that each UT must be assigned

exactly one AP. Inequalities (4) impose that only one PL can be selected for each AP. Inequalities (5) ensure that no AP can be assigned to any UT if the AP is powered off. Inequalities (6) are the capacity constraints for each AP. If $r_{ij}(\pi_j) = 0$, we simply assume that $x_{ij}/r_{ij}(\pi_j) = 0$. Note that, if $r_{ij}(\pi_j)$ is linear of the form $r_{ij}(\pi_j) = \alpha_{ij}\pi_j \equiv \alpha_{ij} \sum_{k \in \mathcal{K}} p_k y_{jk}$, then the constraints would simplify in the usual way: $\sum_{i \in \mathcal{I}} (w_i/\alpha_{ij}) x_{ij} \leq \sum_{k \in \mathcal{K}} p_k y_{jk}$. Relations (7) and (8) define the integrality of the variables. Note that, given that only one PL can be chosen for each AP, location variables are not needed, since they are captured in the terms $\sum_{k \in \mathcal{K}} y_{jk}$. This is why the fixed location cost p_0 is part of the cost associated with the y variables in the objective function.

For the problem to be feasible, there must be enough capacity at the APs to satisfy the demand of each UT, which yields the valid inequalities:

$$(9) \quad \sum_{j \in \mathcal{J}} r_{ij}(\pi_j) \geq w_i, \quad i \in \mathcal{I}.$$

To derive linear constraints, we could replace $r_{ij}(\pi_j)$ with any linear upper approximation $r_{ij}^u(\pi_j)$ to obtain the valid inequalities:

$$(10) \quad \sum_{j \in \mathcal{J}} r_{ij}^u(\pi_j) \geq w_i, \quad i \in \mathcal{I}.$$

For the function $r_{ij}(\pi_j)$ given in (1), we could simply use $r_{ij}^u(\pi_j) = \alpha_{ij}\pi_j \equiv \alpha_{ij} \sum_{k \in \mathcal{K}} p_k y_{jk}$. The same linear upper approximation of $r_{ij}(\pi_j)$ allows us to derive the following valid inequalities from the capacity constraints (6):

$$(11) \quad \sum_{i \in \mathcal{I}} \frac{w_i x_{ij}}{r_{ij}^u(\pi_j)} \leq 1, \quad j \in \mathcal{J}.$$

Another class of valid inequalities can be derived by observing that there cannot be an assignment between AP j and UT i if $r_{ij}(\pi_j) = 0$. This happens when $\pi_j \equiv \sum_{k \in \mathcal{K}} p_k y_{jk} \leq \gamma_{ij}$. Without loss of generality, we can assume that there is no $k \in \mathcal{K}$ such that $p_k = \gamma_{ij}$, since otherwise, we could slightly increase γ_{ij} without changing the problem. This implies that x_{ij} must be 0 whenever $\pi_j \equiv \sum_{k \in \mathcal{K}} p_k y_{jk} < \gamma_{ij}$, which can be written as follows:

$$(12) \quad x_{ij} \leq \max\{0, \sum_{k \in \mathcal{K}} p_k y_{jk} - \gamma_{ij} + 1\}, \quad i \in \mathcal{I}, j \in \mathcal{J}.$$

Indeed, if $\pi_j < \gamma_{ij}$, then the right-hand side of the inequality is $\max\{0, \pi_j - \gamma_{ij} + 1\} < 1$, which, combined with $x_{ij} \in \{0, 1\}$, forces x_{ij} to take value 0. Otherwise, the inequalities are redundant, since $\max\{0, \pi_j - \gamma_{ij} + 1\} \geq 1$. It is straightforward to linearize these constraints, at the expense of adding variables representing the max on the right-hand side.

3 Benders Decomposition Method

The Benders decomposition method proceeds by iteratively fixing the y variables to values \bar{y} , thus leaving a simple subproblem, having only the x variables, whose purpose is to verify the feasibility of the assignment of y to \bar{y} . To find the values \bar{y} , the so-called Benders master problem is solved. In our case, this problem is defined by (2), (4), (8), the valid inequalities (10), and a set of inequalities called Benders feasibility cuts that will be generated dynamically at each iteration. These inequalities are valid for the problem, no other valid inequalities are necessary, and there are a finite number of them; these hypotheses guarantee the convergence of the method towards an optimal solution. Moreover, since at any iteration, the Benders master problem contains a subset of these valid inequalities, it defines a relaxation and any optimal solution to the master problem therefore produces a lower bound z^l .

The subproblem, denoted $S(\bar{y})$, has the following structure, where $\mathcal{J}(\bar{y}) \subseteq \mathcal{J}$ is the set of APs that are powered on and assigned some PL in \bar{y} :

$$(13) \quad \sum_{j \in \mathcal{J}(\bar{y})} x_{ij} = 1, \quad i \in \mathcal{I},$$

$$(14) \quad \sum_{i \in \mathcal{I}} \frac{w_i x_{ij}}{r_{ij}(\bar{\pi}_j)} \leq 1, \quad j \in \mathcal{J}(\bar{y}),$$

$$(15) \quad x_{ij} \in \{0, 1\}, \quad i \in \mathcal{I}, j \in \mathcal{J}(\bar{y}),$$

where $\bar{\pi}_j = \sum_{k \in \mathcal{K}} p_k \bar{y}_{jk}$ for any $j \in \mathcal{J}(\bar{y})$. In general, inequalities (14) have the structure of knapsack constraints, which implies that $S(\bar{y})$ cannot be solved as a linear program. Therefore, we cannot use classical Benders decomposition and we rely instead on some form of logic-based Benders decomposition [2], which also bears resemblance with combinatorial Benders decomposition [1].

If $S(\bar{y})$ is feasible, we then obtain an upper bound $z(\bar{y})$ on z ; if $z(\bar{y}) = z^l$, \bar{y} is an optimal solution and we can stop the algorithm. This case occurs when the Benders master problem is solved to optimality and the optimal solution identified is feasible for the subproblem.

If $S(\bar{y})$ is infeasible, a simple Benders feasibility cut would be:

$$(16) \quad \sum_{j \in \mathcal{J} | \bar{Y}_j = 0} \sum_{k \in \mathcal{K}} y_{jk} + \sum_{j \in \mathcal{J}, k \in \mathcal{K} | \bar{y}_{jk} = 1} (1 - y_{jk}) \geq 1,$$

where $\bar{Y}_j = \sum_{k \in \mathcal{K}} \bar{y}_{jk}$. This valid inequality can be significantly improved, however, by remarking that an infeasible $S(\bar{y})$ implies that \bar{y} does not provide enough capacity to satisfy the demands. Hence, at least one value of $r_{ij}(\pi_j)$ must be increased; using the fact that $r_{ij}(\pi_j)$ is nondecreasing, and given that

$\bar{\pi}_j = \sum_{k \in \mathcal{K}} p_k \bar{y}_{jk}$, we must increase the power of at least one AP. This can be written as follows:

$$(17) \quad \sum_{j \in \mathcal{J} | \bar{Y}_j = 0} \sum_{k \in \mathcal{K}} y_{jk} + \sum_{j \in \mathcal{J} | \bar{Y}_j = 1} \sum_{k \in \mathcal{K} | p_k > \bar{\pi}_j} y_{jk} \geq 1.$$

The validity of (17) follows from the observation that if some AP, say j , is assigned power $\bar{\pi}_j$ in infeasible \bar{y} , then a feasible solution cannot be obtained by reducing the PL to k such that $p_k < \bar{\pi}_j$. The dominance of (17) over (16) follows from the relation $\sum_{j \in \mathcal{J}, k \in \mathcal{K} | \bar{y}_{jk} = 1} (1 - y_{jk}) \geq \sum_{j \in \mathcal{J} | \bar{Y}_j = 1} \sum_{k \in \mathcal{K} | p_k > \bar{\pi}_j} y_{jk}$.

The Benders decomposition method thus follows the general steps outlined in Algorithm 1, where z^* is the value of the best feasible solution found so far.

Algorithm 1

- (i) $z^* = \infty$.
- (ii) Solve a relaxation of the problem (normally, the Benders master problem) to derive a lower bound z^l and candidate solutions $\bar{y}^0, \bar{y}^1, \dots, \bar{y}^n$ that satisfy constraints (4) and (8).
- (iii) For each candidate solution \bar{y}^q , $q = 0, 1, \dots, n$:
 - (a) Solve subproblem $S(\bar{y}_q)$.
 - (b) If $S(\bar{y}^q)$ is feasible, then $z^* = \min\{z^*, z(\bar{y}^q)\}$; if $z^* = z^l$, then STOP the algorithm.
 - (c) If $S(\bar{y}^q)$ is infeasible, then generate a Benders feasibility cut of the form (17) with $\bar{y} = \bar{y}^q$.
- (iv) Go to Step ii.

In Step ii, a relaxation of the problem, normally the Benders master problem, is solved to derive candidate solutions that satisfy (4) and (8). In our case, however, we initially solve a different relaxation than the Benders master problem. When no Benders feasibility cuts are known, we solve the linear IP relaxation defined by (2)-(5), (7), (8) and (11). The resulting relaxation can be efficiently solved, thus providing an initial set of effective Benders feasibility cuts. A second adaptation consists in solving not only subproblem $S(\bar{y}^q)$ for each candidate solution \bar{y}^q , but also, as a preliminary step, subproblem $S(\tilde{y}^q)$, where \tilde{y}^q is defined as follows: $\tilde{y}_{jk_{max}}^q = 1$ if $\bar{Y}_j = 1$, and $\tilde{y}_{jk}^q = 0$ otherwise, where k_{max} is such that $p_{k_{max}} \geq p_k$, $k \in \mathcal{K}$. If subproblem $S(\tilde{y}^q)$ is infeasible, then there cannot be a feasible solution with an increased PL assigned to any AP j that is powered on in \bar{y}^q , i.e., in other words, we must power on at least

one AP that is powered off in \bar{y}^q :

$$(18) \quad \sum_{j \in \mathcal{J} | \bar{Y}_j = 0} \sum_{k \in \mathcal{K}} y_{jk} \geq 1.$$

Note that this cut has the same form as inequality (17), but with \bar{y} replaced with \tilde{y} . It is, however, a stronger cut, if there is a pair $j-k$ with $\bar{Y}_j = 1$ and $p_k > \bar{\pi}_j$. To summarize, if subproblem $S(\tilde{y}^q)$ is infeasible, we generate the stronger Benders feasibility cut (18) and then immediately verify the feasibility of the next candidate solution. Otherwise, if $S(\tilde{y}^q)$ is feasible, then we solve subproblem $S(\bar{y}^q)$ and proceed as in Steps **b** and **c** of Algorithm 1.

If the relaxation in Step **ii** is solved to optimality, the examination of the candidate solutions should obviously start with an optimal solution, say \bar{y}^0 . If $S(\bar{y}^0)$ is feasible, then the algorithm stops, since we have found an optimal solution to the relaxation that is also feasible for the original problem. In this case, we necessarily have $z^* = z(\bar{y}^0) = z^l$. In general, many candidate solutions can be derived when solving the relaxation in Step **ii**. Indeed, if the relaxation is solved with a general IP solver, any feasible solution encountered during the search of the branch-and-bound tree can be used as a candidate solution \bar{y}^q , since it satisfies constraints (4) and (8). This condition ensures that a corresponding feasible subproblem $S(\bar{y}^q)$ yields a feasible solution to the original problem and z^* can be updated accordingly. However, if \bar{y}^q is not the optimal solution to the relaxation, then we necessarily have $z(\bar{y}^q) \geq z^* > z^l$ and the algorithm continues.

4 Computational Results

To assess the performance of the Benders decomposition method, we have implemented a standard cutting-plane approach that solves, at each iteration, a relaxation containing x and y variables. The initial relaxation is defined by (2)-(5), (7), (8) and (11), to which we add the valid inequalities (12), linearized by the addition of extra variables to get rid of the max on the right-hand side. Let us denote by (\bar{x}, \bar{y}) the optimal solution to this model and by $\bar{\pi}_j = \sum_{k \in \mathcal{K}} p_k \bar{y}_{jk}$ the power consumed by AP j in this solution. We then compute the available capacity between AP j and UT i as follows: $\bar{r}_{ij} = \min\{r_{ij}^u(\bar{\pi}_j), r_{max}\}$. Because of the presence of constraints (12) in the relaxation, we know that $\bar{r}_{ij} = r_{ij}(\bar{\pi}_j)$. However, the capacity constraints (6) are not necessarily satisfied; if this is the case, we then add the following cut to the relaxation:

$$(19) \quad \sum_{j \in \mathcal{J}} \left\{ \sum_{i \in \mathcal{I} | \bar{x}_{ij} = 0} x_{ij} + \sum_{i \in \mathcal{I} | \bar{x}_{ij} = 1} (1 - x_{ij}) + \sum_{k \in \mathcal{K} | \bar{y}_{jk} = 0} y_{jk} + \sum_{k \in \mathcal{K} | \bar{y}_{jk} = 1} (1 - y_{jk}) \right\} \geq 1.$$

$ \mathcal{I} , \mathcal{J} , \mathcal{K} $	BD			CP		
	AVG	STD	NUI	AVG	STD	NUI
50, 5, 3	0.01	0.00	0	0.02	0.00	0
50, 10, 3	0.60	0.23	0	1.17	0.73	0
100, 10, 2	4.25	0.59	0	5303	1471	7
100, 10, 3	2.88	0.81	0	1612	1110	2
100, 10, 4	4.56	1.21	0	3437	1377	5
150, 10, 3	23.65	8.66	0	7382	1619	9
150, 15, 3	549.54	263.79	0	18256	4791	7

Table 1
CPU (s), average (AVG) + standard deviation (STD); number of unsolved instances (NUI)

Note that this constraint is considerably weaker than the Benders cut (17), since the only information we use when the solution (\bar{x}, \bar{y}) is infeasible is that at least one of the variables must change its value. We then solve the new relaxation thus obtained and stop the cutting-plane algorithm when the solution is feasible, because it is then optimal for the original problem.

For evaluating the relative performance of the two methods, Benders decomposition (BD) and cutting-plane (CP), we have compared them on seven classes of 20 randomly generated instances, each one characterized by their dimension, i.e., $|\mathcal{I}|$, $|\mathcal{J}|$, and $|\mathcal{K}|$. Table 1 reports the average (AVG) and the standard error (STD) of the CPU time for each class. The methods are stopped after one hour of CPU time; therefore, we also report the number of unsolved instances (NUI). The table highlights the better performance of BD with respect to CP. BD is often orders of magnitude faster than CP and solves all instances within the one hour time limit, while CP fails to solve 30 of the 140 instances. The superiority of BD is amplified when the problem dimension increases, especially for instances with at least 100 UTs.

References

- [1] Codato, G. and M. Fischetti, *Combinatorial Benders' cuts for mixed-integer linear programming*, Operations Research **54** (2006), 756–766.
- [2] Hooker, J.N. and G. Ottosson, *Logic-based Benders decomposition*, Mathematical Programming **96** (2003), 33–60.

k-Edge Failure Resilient Network Design

Richard Li-Yang Chen^{1,2}

*Quantitative Modeling & Analysis
Sandia National Laboratories
Livermore, CA, USA*

Cynthia A. Phillips³

*Discrete Mathematics & Complex Systems
Sandia National Laboratories
Albuquerque, NM, USA*

Abstract

We design a network that supports a feasible multicommodity flow even after the failures of *any k* edges. We present a mixed-integer linear program (MILP), a cutting plane algorithm, and a column-and-cut algorithm. The algorithms add constraints to repair vulnerabilities in partial network designs. Empirical studies on previously unsolved instances of SNDlib demonstrate their effectiveness.

Keywords: Column-cut generation, valid inequalities, separation problems, network design, edge-failure resiliency

¹ Sandia National Laboratories (SNL) is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000. Partly funded by SNL's Laboratory-Directed Research and Development and the U.S. Department of Energy's Advanced Scientific Computing Research programs.

² Email: rlchen@sandia.gov

³ Email: caphill@sandia.gov

1 Introduction

Telecommunications networks must handle increasing traffic, including supporting critical services such as emergency response. To ensure flows can be rerouted after failures, sufficient excess capacity must be available on the surviving edges. The expense of capacity overprovisioning motivates the design of low-cost failure-resilient networks.

We address the k edge-failure resilient network design problem (R-NDP): minimize the total cost to build a network that survives the failure of *any* k edges, maintaining flow feasibility with no loss of demand. Earlier work by [2,3] describe polyhedral approaches and cutting plane algorithms for solving the single edge (or node) failure variant of R-NDP. Given edge set \mathcal{E} and a failure budget k , the number of distinct failure scenarios is $(|\mathcal{E}|)^k$. As network size and/or the failure budget increases, the number of scenarios grows quickly. As our computational results in Section 5 demonstrate, explicit methods that require the evaluation of each failure scenario are intractable in most cases, even with the use of advanced decomposition techniques such as Benders decomposition (BD) or column generation.

This type of resiliency model differs from probabilistic models where edges fail with some specified probabilities or failure scenario models where a manageable input set of failure scenarios are pre-selected. For example, given scenarios, [7] uses Benders decomposition: evaluate each scenario and generate a Benders feasibility cut and other valid inequalities to “guide” the master problem. However, accurately estimating edge failure probabilities or correctly selecting “critical” failure scenarios are extremely challenging tasks. Another popular alternative is requiring graph-theoretic connectivity such as k -connectivity: k edge-disjoint paths between each pair of vertices. For an overview of survivable network design, see [4] and a recent survey [6].

R-NDP generalizes k -edge disjoint path network design and the minimum cost k -edge connected spanning subgraph [5], and so is NP hard. Using a multicommodity flow MILP formulation, we present several families of valid inequalities, use separation to explicitly consider only a tractable-sized failure set, exploit structure to eliminate multiple cutset violations simultaneously, and present computational results demonstrating our algorithms’ efficacy.

2 Models

The *nominal network design problem* (N-NDP) is failure-free multicommodity network design. Each edge has a fixed capacity that is partitioned between

flows in both direction. Thus edges are undirected but flows are directed. For N-NDP the objective is to select a minimum cost set of edges that can feasibly support the multicommodity flow, that is, satisfy all demands. The optimal solutions to N-NDP usually have little excess capacity or redundancy.

2.1 Edge-failure resilient network design problem

Let $\mathcal{G}(\mathcal{N}, \mathcal{E})$ be an undirected graph with node set \mathcal{N} and edge set \mathcal{E} . Let \mathcal{A} be the set of all directed arcs corresponding to edge set \mathcal{E} . Let \mathcal{Q} be the set of commodities to be routed in the network and let (s^q, t^q, b^q) be a triplet representing a commodity $q \in \mathcal{Q}$, where s^q is the origin node, t^q is the destination node, and b^q is the demand to be routed from s^q to t^q . For all $\{i, j\} \in \mathcal{E}$, let c_{ij} and u_{ij} be the fixed cost of installation and the edge capacity, respectively. Let x_{ij} be the binary edge selection variable that takes value 1 if edge $\{i, j\} \in \mathcal{E}$ is selected and 0 otherwise. Given a failure budget k , let $\mathcal{D}(k)$ be the set of all k -edge failures. Let $y_{ij}^{q\ell}$ be the flow of commodity q on directed arc $(i, j) \in \mathcal{A}$ in failure scenario $\ell = 1, \dots, |\mathcal{D}(k)|$.

We wish to design a minimum cost network such that a feasible multi-commodity flow exists under any failure scenario prescribed by 0-1 vector $\tilde{d} \in \mathcal{D}(k)$. R-NDP is then:

$$\begin{aligned} \min_{x,y} \quad & \sum_{\{i,j\} \in \mathcal{E}} c_{ij} x_{ij} \\ \text{s.t.} \quad & \left. \begin{aligned} \sum_{\{j|(i,j) \in \mathcal{A}\}} y_{ij}^{q\ell} - \sum_{\{j|(j,i) \in \mathcal{A}\}} y_{ji}^{q\ell} &= \begin{cases} b^q & i = s^q \\ -b^q & i = t^q \\ 0 & \text{otherwise} \end{cases} \\ \forall i \in \mathcal{N}, q \in \mathcal{Q}, \quad & \end{aligned} \right\} \forall \ell = 1, \dots, |\mathcal{D}(k)|, \quad (1) \\ & \sum_{q \in \mathcal{Q}} (y_{ij}^{q\ell} + y_{ji}^{q\ell}) \leq u_{ij}(x_{ij} - \tilde{d}_{ij})^+ \quad \forall \{i, j\} \in \mathcal{E}, \\ & y_{ij}^{q\ell} \geq 0 \quad \forall q \in \mathcal{Q}, (i, j) \in \mathcal{A}, \\ & x_{ij} \in \{0, 1\} \quad \forall \{i, j\} \in \mathcal{E}. \end{aligned}$$

The objective minimizes the total network design cost. Constraints, in order, enforce flow balance at each node; edge capacity limits; and non-negativity and integrality constraints. R-NDP resembles a two-stage stochastic program structurally, so it is amenable to a Benders decomposition (BD) [1].

3 Valid inequalities

In this section, we present three classes of valid inequalities characterizing the feasibility of R-NDP (1). Given a candidate network design \tilde{x} and a k -edge failure scenario $\tilde{d} \in \mathcal{D}(k)$ the following are valid inequalities.

Benders inequalities. A feasible multicommodity flow exists \iff

$$\sum_{q \in \mathcal{Q}} b^q (\pi_{s^q}^q - \pi_{t^q}^q) + \sum_{\{i,j\} \in \mathcal{E}} u_{ij} (\tilde{x}_{ij} - \tilde{d}_{ij})^+ \alpha_{ij} \leq 0, \quad \forall (\pi, \alpha) \in \mathcal{P} \quad (2)$$

where π and α are dual variables associated with flow-balance and edge-capacity constraints, respectively, and \mathcal{P} is the dual feasible domain of of (1).

Cutset inequalities. We adjust the *cut condition*, necessary for a feasible multicommodity flow, for failures. For any $\mathcal{S} \subset \mathcal{N}$, $\mathcal{S} \neq \emptyset$, let *cutset* $\delta(\mathcal{S})$ be the set of edges in \mathcal{E} with one endpoint in \mathcal{S} and the other in $\mathcal{N} \setminus \mathcal{S}$. Let $\tau(\mathcal{S})$ be the set of commodity demands with the origin or destination in \mathcal{S} and the other end node in $\mathcal{N} \setminus \mathcal{S}$. Then these constraints hold for R-NDP.

$$\sum_{\{i,j\} \in \delta(\mathcal{S})} u_{ij} (\tilde{x}_{ij} - \tilde{d}_{ij})^+ \geq \sum_{q \in \tau(\mathcal{S})} b^q \quad \forall \mathcal{S} \subset \mathcal{N}, \mathcal{S} \neq \emptyset \quad (3)$$

k -paths inequalities. A feasible R-NDP solution must contain at least $k + 1$ edge disjoint paths between (s^q, t^q) for each commodity $q \in \mathcal{Q}$. The necessary k -paths inequalities are:

$$(k + 1)(\gamma_{s^q}^q - \gamma_{t^q}^q) + \sum_{\{i,j\} \in \mathcal{E}} \eta_{ij}^q \tilde{x}_{ij} \leq 0, \quad \forall (\gamma, \eta) \in \mathcal{P}^q, q \in \mathcal{Q} \quad (4)$$

where γ and η are dual variables associated with flow-balance and edge-capacity constraints and \mathcal{P}^q is the feasible domain of the dual of (6).

4 Separation problems

Using BD, R-NDP may not be tractable for practical-sized networks if we explicitly check each failure scenario. We present separation oracles to *implicitly* search for a k -edge-failure scenario that cannot be survived by the current network design \tilde{x} . Define $\mathcal{E}^{\tilde{x}}$ as the set of edges in the candidate network design \tilde{x} , that is the set of edges $\{i, j\} \in \mathcal{E}$ with $\tilde{x}_{ij} = 1$.

Separation for Benders feasibility cuts: For a candidate network design \tilde{x} , this bilevel MILP identifies a failure scenario, prescribed by d_{ij} for all $\{i, j\} \in \mathcal{E}^{\tilde{x}}$,

that maximizes unsatisfied demand.

$$\begin{aligned}
& \max_d \min_{w,y} \sum_{q \in \mathcal{Q}} w^q \\
\text{s.t. } & \sum_{\{i,j\} \in \mathcal{E}} d_{ij} = k \\
& \sum_{\{j|(j,i) \in \mathcal{A}\}} y_{ij}^q - \sum_{\{j|(i,j) \in \mathcal{A}\}} y_{ji}^q = \begin{cases} b^q - w^q & i = s^q \\ -b^q + w^q & i = t^q \\ 0 & \text{otherwise} \end{cases} \\
& \forall i \in \mathcal{N}, q \in \mathcal{Q}, \\
& \sum_{q \in \mathcal{Q}} (y_{ij}^q + y_{ji}^q) \leq u_{ij}(1 - d_{ij}) \quad \forall \{i,j\} \in \mathcal{E}^{\tilde{x}} \\
& y_{ij}^q \geq 0 \quad \forall q \in \mathcal{Q}, (i,j) \in \mathcal{A}, \\
& w^q \geq 0 \quad \forall q \in \mathcal{Q}, \\
& d_{ij} \in \{0,1\} \quad \forall \{i,j\} \in \mathcal{E}.
\end{aligned} \tag{5}$$

Variables d represent edge failures (upper-level). Vectors w and y represent unmet demand and flow variables (lower-level) given failure scenario d . Let $z(\tilde{x}) \geq 0$ be the optimal solution to (5). If $z(\tilde{x}) > 0$, the optimal d generates a violated Benders inequality (2). Otherwise, network \tilde{x} survives all size- k edge failures.

Separation for k-paths inequalities: Given network expansion decisions \tilde{x} and a commodity $q \in \mathcal{Q}$, we can identify violated k -edge-disjoint paths for commodity q by solving the *linear program* (LP) in (6). Let $z(\tilde{x}, q)$ be the optimal solution to (6). If LP optimal solution $z(\tilde{x}, q) > 0$, its dual multipliers generate a violated k -paths inequality (4). Otherwise, $z(\tilde{x}, q) = 0$ and there are no violations of (6).

$$\begin{aligned}
& \min_{\beta,y} \beta \\
\text{s.t. } & \sum_{\{j|(j,i) \in \mathcal{A}\}} y_{ij} - \sum_{\{j|(i,j) \in \mathcal{A}\}} y_{ji} = \begin{cases} k + 1 - \beta & i = s^q \\ -k - 1 + \beta & i = t^q \\ 0 & \text{otherwise} \end{cases} \\
& \forall i \in \mathcal{N}, \\
& y_{ij} + y_{ji} \leq \tilde{x}_{ij} \quad \forall \{i,j\} \in \mathcal{E}, \\
& \beta \geq 0, \quad y_{ij} \geq 0 \quad \forall (i,j) \in \mathcal{A}.
\end{aligned} \tag{6}$$

Separation based on cutset inequalities (3): Given a candidate network design \tilde{x} , construct a graph containing an edge with weight u_{ij} for each $\tilde{x}_{ij} = 1$ and

an edge with weight $-b^q$ between s^q and t^q for every commodity $q \in \mathcal{Q}$. The weight of a cut is the sum of the weights of all edges in the cut. A cut of negative weight is violated, found by solving the following MILP, a *minimum-cut-like* problem. Binary variable ρ_i , for each node $i \in \mathcal{N}$, indicates sides of a cut $(\mathcal{S}, \mathcal{N} - \mathcal{S})$. Vector d_{ij} for all $\{i, j\} \in \mathcal{E}$ again represents binary edge failures. The edge failure variables prevent the use of min-cut network algorithms like Edmonds-Karp and Karger.

$$\min_{d, \rho, w} \sum_{\{i, j\} \in \mathcal{E}^{\tilde{x}}} u_{ij} w_{ij} - \sum_{q \in \mathcal{Q}} b^q w^q \quad (7a)$$

$$\text{s.t. } \sum_{\{i, j\} \in \mathcal{E}} d_{ij} = k \quad (7b)$$

$$\left. \begin{array}{l} \rho_i - \rho_j + w_{ij} + d_{ij} \geq 0 \quad \forall \{i, j\} \in \mathcal{E}^{\tilde{x}}, \\ \rho_j - \rho_i + w_{ij} + d_{ij} \geq 0 \quad \forall \{i, j\} \in \mathcal{E}^{\tilde{x}}, \end{array} \right\} \quad (7c)$$

$$\left. \begin{array}{l} 2 - \rho_{s^q} - \rho_{t^q} - w^q \geq 0 \quad \forall q \in \mathcal{Q}, \\ \rho_{s^q} + \rho_{t^q} - w^q \geq 0 \quad \forall q \in \mathcal{Q}, \end{array} \right\} \quad (7d)$$

$$\begin{aligned} \rho_i, w^q &\in \{0, 1\} \quad \forall i \in \mathcal{N}, q \in \mathcal{Q} \\ w_{ij}, d_{ij} &\in \{0, 1\} \quad \forall \{i, j\} \in \mathcal{E}^{\tilde{x}} \end{aligned}$$

Objective (7a) minimizes the difference between the cut capacity and the commodity demand that crosses the cut. Constraint (7b) enforces the edge failure budget. Constraints (7c) ensure any edge connecting \mathcal{S} to $\mathcal{N} \setminus \mathcal{S}$ either fails ($d_{ij} = 1$) or contributes to the cut capacity ($w_{ij} = 1$). Objective pressure ensures that $w_{ij} = 0$ if $\rho_i = \rho_j$ or $d_{ij} = 1$. Constraints (7d) ensure that if a commodity's source and terminal nodes are in different partitions then its demand contributes to the objective.

Combinatorial cutset inequalities: Unlike N-NDP where a single cutset inequality secures a vulnerable node bipartition, R-NDP may require a sequence of cutset inequalities to secure a vulnerability. For example, suppose there are n edges with identical cost and capacity in a cutset and let $k = 1$. If all edges have identical cost and capacity, there are n failure sets that select one edge from this cutset. Each might be generated and require a cutset inequality. This increases runtime, and will likely weaken the master problem LP relaxation.

Proposition 4.1 *Adding (8), derived from the dual knapsack problem, forces sufficient capacity across the cutset $\delta(\mathcal{S})$.*

$$\left. \begin{array}{l} \sum_{\{i, j\} \in \delta(\mathcal{S})} u_{ij} x_{ij} - k \lambda^s - \sum_{\{i, j\} \in \delta(\mathcal{S})} \mu_{ij}^s \geq \sum_{q \in \tau(\mathcal{S})} b^q \\ \lambda^s + \mu_{ij}^s \geq u_{ij} x_{ij} \quad \forall \{i, j\} \in \delta(\mathcal{S}) \\ \mu_{ij}^s \geq 0 \quad \forall \{i, j\} \in \delta(\mathcal{S}) \\ \lambda^s \geq 0 \end{array} \right\} \quad (8)$$

Table 1

Solving R-NDP. time = wall clock seconds. OM = Out of memory. RN = solving root. o-gap% = optimality gap at exit. f-gap% = feasibility gap percent (see text).

Instance	k	EF		BD		CPA		CCA	
		time	o-gap%	time	f-gap%	time	f-gap%	time	f-gap%
<i>newyork</i>	0	10,800	28.16	10,800	98.25	29	0	28	0
	1	10,800	65.87	10,800	96.34	1	0	1	0
	2	10,800	OM	10,800	98.20	10,800	52.76	4,472	0
	3	10,800	OM	10,800	100	8	0	8	0
<i>pdh</i>	0	10,800	0	10,800	67.09	2,701	0	2,725	0
	1	10,800	89.61	10,800	25.49	7,175	0	4,102	0
	2	10,800	OM	10,800	26.05	10,800	50.75	4,419	0
	3	10,800	OM	10,800	100	10,800	34.19	4,016	0
<i>norway</i>	0	10,800	27.81	10,800	96.02	10,800	15.28	10,800	15.28
	1	10,800	RN	10,800	97.94	436	0	365	0
	2	10,800	OM	10,800	98.99	745	0	233	0
	3	10,800	OM	10,800	100	3,494	0	3,487	0

5 Computational Experiments

We implemented our proposed algorithms in C++ using IBM’s Concert Technology Library 2.9 and the CPLEX 12.4 MILP solver. We ran all experiments on a workstation with two quad-core, hyper-threaded 2.93GHz Intel Xeon processors with 96GB of memory. By default CPLEX 12.4 allocates a thread for each core. Hyper-threaded architectures present each core as a virtual dual-core, though the performance is worse than a dual core. The workstation is shared by other users, so our run-times are conservative. We used CPLEX default options, except we set the optimality gap to 0.1%. We stopped runs after 10,800 wall-clock seconds (3 hours). We selected three networks (*newyork*, *pdh*, and *norway*) from the SNDlib [8] and four failure budgets $k = 0, 1, 2, 3$. We considered these algorithmic variants using valid inequalities from Section 3 and separation problems from Section 4.: (A) the extensive form (EF) (1), (B) Bender decomposition (BD), (C) a cutting plane algorithm (CPA) that generates violated k -paths, cutset, and Benders inequalities, and (D) a column-and-cut algorithm (CCA) that generates k -paths, combinatorial cutset (column-and-cut), and Benders inequalities.

Table 1 gives results. EF can only solve the smallest instances, since it has multicommodity flow constraints for each failure scenario. BD delays cut generation. The feasibility gap, computed at the end of the allotted time, is the maximum percentage of demand loss under the worst case k -edge failure scenario. Although BD does not explicitly incorporate flow constraints for each failure scenario, each

scenario must still be explicitly screened. Thus BD mitigates the memory issues associated with EF, but still has exponential runtime. Overall, BD appears to scale slightly better than EF, but it still fails to converge optimally for any test instance.

Except for the *norway* network with $k = 0$, our column-and-cut algorithm CCA solved all instances within the 3-hour runtime limit. We are able to implicitly evaluate all the scenarios to identify a violated failure scenario and then generate columns-and-cuts to eliminate groups of failure scenarios simultaneously. Although CCA failed to solve the *norway* network with $k = 0$ within the allocated time, the final solution is within 1.5% of the optimal solution cost computed using CCA without time limits. Space constraints prohibit giving timing breakdowns among the MIPs but the master problem and (7) are the hardest. For large k , the k -paths constraints are constraining and extremely helpful. This in part explains the relative ease of solving instances with larger failure budgets $k = 1, 2, 3$ and the difficulties associated with solving less constrained instances with zero failure budgets.

Future work: Many of the ideas in this paper carry over to budget-constrained failure sets where edge-failure costs may be greater than one.

References

- [1] Benders, J., *Partitioning Procedures for Solving Mixed-Variables Programming Problems*, Numer Mathematik **4** (1962), 238–252.
- [2] Bienstock, D. and M. Gabriella, *Strong inequalities for capacitated survivable network design problems*, Math. Program., **89** (2000), 124–147.
- [3] Dahl, G. and M. Stoer *A cutting plane algorithm for multicommodity survivable network design problems*, INFORMS Journal on Computing, **10** (1998), 1–11.
- [4] Balakrishnan, A., P. Mirchandani, H. Natarajan, *Connectivity upgrade models for survivable network design*, Operations Research, **57** (2009), 170–186.
- [5] Magnanti, T., S. Raghavan, *Strong formulations for network design problems with connectivity requirements*, Networks **45** (2005), 61–79.
- [6] Kerivin, H. and A.R. Mahjoub, *Design of Survivable Networks: A survey*, Networks **46** (2005), 1–21.
- [7] Garg, M. and J.C. Smith, *Models and algorithms for the design of survivable multicommodity flow networks with general failure scenarios*, Omega **36** (2008), 1057–1071.
- [8] Orlowski, S., R. Wessäly, M. Piòro, and A. Tomaszewski, *SNDlib 1.0 - Survivable Network Design Library*, Networks **55** (2010), 276–286. Providence, R.I., 1961.

A New Lower Bound for the Kirchhoff Index using a numerical procedure based on Majorization Techniques

Alessandra Cornaro^{1,2}

*Department of Mathematics and Econometrics
Catholic University of Milan, Italy*

Gian Paolo Clemente^{1,3}

*Department of Mathematics and Econometrics
Catholic University of Milan, Italy*

Abstract

In this note, we use a procedure, proposed in [1], based on a majorization technique, which localizes real eigenvalues of a matrix of order n . Through this information, we compute a lower bound for the Kirchhoff index (see [3]) that takes advantage of additional eigenvalues bounds. An algorithm has been developed with MATLAB software to evaluate the above mentioned bound. Finally, numerical examples are provided showing how tighter results can be obtained.

Keywords: Kirchhoff Index; Graphs; Majorization order

¹ We thank Monica Bianchi and Anna Torriero for their valuable comments and suggestions

² Email: alessandra.cornaro@unicatt.it

³ Email: gianpaolo.clemente@unicatt.it

1 Introduction

The evaluation of the effective resistance between any pair of vertices of a network and the computation of the Kirchhoff index have interest in electric circuit and probabilistic theory. The Kirchhoff index has been also used in Chemistry as an alternative for discriminating among different molecules with similar shapes and structures (see [9]). Recently Bianchi et al. [3] proposed a variety of lower and upper bounds for the Kirchhoff index based on majorization techniques. In particular, the authors showed that it is possible to obtain tighter results taking into account additional information on the localization of the eigenvalues of the transition matrix of the graph. From a theoretical point of view, some well-known inequalities on the localization of real eigenvalues have been provided in literature and they can be used to compute the above mentioned bounds. Our aim is to provide a numerical procedure in order to obtain the eigenvalues bounds by means of a technique proposed in Bianchi and Torriero [1] and based on nonlinear global optimization problems solved through majorization techniques. Using this approach, we compute a lower bound for the Kirchhoff index and we compare our results to those existing in literature. In the last section, some numerical examples are presented.

2 The Kirchhoff index

Let us recall some basic graph notations (for more details see [10]).

Let $G = (V, E)$ be a simple, connected, undirected graph where $V = \{1, 2, \dots, n\}$ is the set of vertices and $E \subseteq V \times V$ the set of edges, $|E| = m$.

The degree sequence of G is denoted by $\pi = (d_1, d_2, \dots, d_n)$ and it is arranged in non-increasing order $d_1 \geq d_2 \geq \dots \geq d_n$, where d_i is the degree of vertex i . The equality $\sum_{i=1}^n d_i = 2m$ holds.

Let A be the adjacency matrix of G and $\lambda_1(A) \geq \lambda_2(A) \geq \dots \geq \lambda_n(A)$ be the set of its (real) eigenvalues. Given the diagonal matrix D of vertex degrees, the matrix $L = D - A$ is known as the Laplacian matrix of G . Let $\lambda_1(L) \geq \lambda_2(L) \geq \dots \geq \lambda_n(L) = 0$ be its eigenvalues. The inequality $\lambda_1(L) \geq 1 + d_1$ is well known. The condition $\lambda_{n-1}(L) > 0$ characterizes the connected graphs.

The transition matrix is $P = D^{-1}A$ and its real eigenvalues are $1 = \lambda_1(P) > \lambda_2(P) \geq \dots \geq \lambda_n(P) \geq -1$.

The *Kirchhoff index* $K(G)$ of a simple connected graph G was defined by

Klein and Randić in [5] as

$$K(G) = \sum_{i < j} R_{ij},$$

where R_{ij} is the effective resistance between vertices i and j , which can be computed using Ohm's law.

This index has been intensively studied in different fields as Chemistry, Complex Networks, Electric Networks and so on.

An alternative expression of the Kirchhoff index is:

$$(1) \quad K(G) = n \sum_{i=1}^{n-1} \frac{1}{\lambda_i(L)},$$

which involves the non-null eigenvalues of the Laplacian L (see [4], [11]).

If G is d -regular, then $L = dI - A$, $P = D^{-1}A = I - \frac{1}{d}L$ and

$$\lambda_{n-i+1}(P) = 1 - \frac{\lambda_i(L)}{d}, \quad i = 1, \dots, n.$$

In this case, we can rewrite (1), in terms of the eigenvalues of the transition matrix P (see [8]) as follows:

$$(2) \quad K(G) = \frac{n}{d} \sum_{i=2}^n \frac{1}{1 - \lambda_i(P)}.$$

When G is an arbitrary connected graph, we have the following bounds (see Corollary 2 in [7]):

$$(3) \quad \left(\frac{n}{d_1} \right) \sum_{i=2}^n \left(\frac{1}{1 - \lambda_i(P)} \right) \leq K(G) \leq \left(\frac{n}{d_n} \right) \sum_{i=2}^n \left(\frac{1}{1 - \lambda_i(P)} \right).$$

Using a majorization technique that identifies the maximal and minimal vectors of a variety of subsets of \mathbb{R}^n (see [2]), several lower and upper bounds for $K(G)$ have been derived in [3] through the expressions in (3).

In particular, in what follows, we make use of the lower bound (13) in [3]. This bound takes into account additional information on the localization of the eigenvalues of the transition matrix P . In fact, if $\lambda_2(P) \geq \beta > 0$, the bound, in terms of β , is given by:

$$(4) \quad K(G) \geq \frac{n}{d_1} \left[\frac{1}{1 - \beta} + \frac{(n - 2)^2}{n - 1 + \beta} \right].$$

In the next section we explain the procedure used to numerically estimate the value of β .

3 Bounds for nonnegative ordered real numbers

In this section, we use the bounds developed in [1] in order to obtain additional information on the eigenvalues localization for computing (4). In what follows, we briefly describe the main steps of the technique presented in Bianchi and Torriero (see [1]). Each bound can be obtained by taking advantage of some results on majorization order applied to the following optimization problem:

$$(P(h)) \quad \max F(x_h) \quad \text{subject to } \mathbf{x} \in S$$

where $F : \mathbb{R} \rightarrow \mathbb{R}$ is an increasing function of x_h , $S = \Sigma_a \cap \{\mathbf{x} \in \mathbb{R}^n : g(\mathbf{x}) = b\}$, with $\Sigma_a = \{\mathbf{x} \in \mathbb{R}^n : x_1 \geq x_2 \geq \dots \geq x_n \geq 0, \sum_{i=1}^n x_i = a\}$ and g a continuous function, homogeneous of degree p , p -real and strictly Schur-Convex (see [6]).

Let \mathbf{e}^j , $j = 1, \dots, n$, be the fundamental vectors of \mathbb{R}^n and set:

$$\mathbf{s}^0 = \mathbf{0}, \mathbf{s}^j = \sum_{i=1}^j \mathbf{e}^i, j = 1, \dots, n, \quad \mathbf{v}^n = \mathbf{0}, \mathbf{v}^j = \sum_{i=j+1}^n \mathbf{e}^i, j = 0, \dots, (n-1).$$

For problem $P(h)$ the following results hold (see Lemma 2.1 and Theorem 3.1 in [1]):

Lemma 3.1 *Let fix $b \in \mathbb{R}$ and consider the set S . Then either $(\frac{a^p}{n^p})g(\mathbf{s}^n) = b$ or there exists a unique integer $1 \leq h^* < n$ such that*

$$(5) \quad \frac{a^p}{(h^* + 1)^p} g(\mathbf{s}^{h^*+1}) < b \leq \frac{a^p}{(h^*)^p} g(\mathbf{s}^{h^*})$$

Theorem 3.2 *The solution of the optimization problem $(P(h))$ is the h -th component of vector $\mathbf{x}_* = (\frac{a}{n})\mathbf{s}^n$ if $(a^p/n^p)g(\mathbf{s}^n) = b$. If $(a^p/n^p)g(\mathbf{s}^n) \neq b$, let h^* the integer satisfying Lemma (3.1) with $1 \leq h^* < n$. The solution of the optimization problem $(P(h))$ is α^* where*

- (i) *for $h > h^*$, α^* is the unique root of the function $f(\alpha) = [g(\mathbf{x}^*(\alpha)) - b]$ in $I = (0, \frac{a}{h}]$, that is the h -th component of the vector $\mathbf{x}^*(\alpha^*) = (a - h\alpha^*)\mathbf{e}^1 + \alpha^*\mathbf{s}^h$.*
- (ii) *for $h \leq h^*$, α^* is the unique root of the function $f(\alpha) = [g(\mathbf{x}_*(\alpha)) - b]$ in $I = (\frac{a}{n}, \frac{a}{h}]$, that is the h -th component of the vector $\mathbf{x}_*(\alpha^*) = \alpha^*\mathbf{s}^h + \rho^*\mathbf{v}^h$ and $\rho^* = \frac{(a - h\alpha^*)}{n - h}$.*

Assume that $g(\mathbf{x}) = \sum_{i=1}^n x_i^p$, with $p > 1$.

If $b = a^p/n^{p-1}$ then $S = \{(a/n)\mathbf{s}^n\}$ and the solution is a/n .

Otherwise, by Lemma (3.1), there is an integer $h^* < n$ such that

$$\frac{a^p}{(h^* + 1)^p} < b \leq \frac{a^p}{(h^*)^p} \quad \text{where} \quad h^* = \left\lfloor \sqrt[p-1]{\frac{a^p}{b}} \right\rfloor.$$

From Theorem 3.2, part (1), we have

$$(6) \quad f(\alpha, p) = (h-1)\alpha^p + (a - h\alpha + \alpha)^p - b$$

and the unique root α^* of the equation $f(\alpha, p) = 0$ in $I = (0, \frac{a}{h}]$ is an upper bound for x_h with $h > h^*$.

From Theorem 3.2, part (2) we find

$$(7) \quad f(\alpha, p) = h\alpha^p + (n-h)\frac{(a-h\alpha)^p}{(n-h)^p} - b$$

and thus the unique root α^* of the equation $f(\alpha, p) = 0$ in $I = (\frac{a}{n}, \frac{a}{h}]$ is an upper bound for x_h with $h \leq h^*$.

In the next section we apply this procedure in order to compute the lower bound (4) for $K(G)$.

4 Application to the Kirchhoff index

To this aim we make a change of variable setting

$$\nu_i = 1 - \lambda_{n-i+1}(P), \quad i = 1, \dots, (n-1).$$

For the vector $\nu \in \mathbb{R}^{n-1}$ we have

$$0 < \nu_{n-1} \leq \nu_{n-2} \leq \dots \leq \nu_1 \leq 2$$

and $\sum_{i=1}^{n-1} \nu_i = n$ since

$$\text{tr}(P) = \sum_{i=1}^n \lambda_i(P) = 0 \Rightarrow \sum_{i=2}^n \lambda_i(P) = -1.$$

Let us now consider the set

$$S = \{\nu \in \mathbb{R}^{n-1} : \sum_{i=1}^{n-1} \nu_i = n, g(\mathbf{x}) = b\}$$

where, now $g(\mathbf{x}) = \sum_{i=1}^{n-1} \nu_i^p$.

We can now rewrite $g(\mathbf{x})$ as follows for every $p \in \mathbb{N}$:

$$(8) \quad g(\mathbf{x}) = \sum_{i=1}^{n-1} \nu_i^p = \sum_{i=1}^{n-1} \left(\sum_{k=0}^p \binom{p}{k} (1)^k (-\lambda_{n-i+1})^{p-k} \right)$$

where

$$(9) \quad \left(\sum_{k=0}^p \binom{p}{k} (1)^k (-\lambda_{n-i+1})^{p-k} \right) = (1 - \lambda_{n-i+1})^p.$$

Now we can reshape (8) as

$$\sum_{k=0}^p \binom{p}{k} (-1)^{p-k} \sum_{i=1}^{n-1} (\lambda_{n-i+1})^{p-k} = \sum_{k=0}^p \binom{p}{k} (-1)^{p-k} \left[\sum_{i=1}^n (\lambda_i)^{p-k} - (\lambda_1)^{p-k} \right]$$

and this is

$$(10) \quad \sum_{k=0}^p \binom{p}{k} (-1)^{p-k} [tr(P^{p-k}) - 1].$$

(Note that (10) is equal to $n - 1$ when $p = k$).

From this starting point, we can say that if

$$\nu_{n-1} = 1 - \lambda_2(P) \leq 1 - \beta = \gamma < \frac{n}{n-1},$$

we face the set

$$T = \{\nu \in S : \nu_{n-1} \leq \gamma\},$$

where γ can be computed by using the methodology explained in Section 3.

5 Numerical examples

In order to compute the bound (4), we have developed an algorithm through the MATLAB software based on the following steps involving the methodology previously described:

- Step 1 generate⁴ randomly a simple connected graph G by fixing the number of vertices;
- Step 2 repeat the following steps for several values of p (with $p \in \mathbb{N}$ and $p > 1$) and for each h (with $1 \leq h \leq (n - 1)$);
 - a) evaluate a equal to the number of vertices and b equal to (10);
 - b) verify if $b = a^p/n^{p-1}$, then the bound is equal to a/n and the procedure will be stopped. Otherwise compute $h^* = \left\lfloor \sqrt[p-1]{\frac{ap}{b}} \right\rfloor$;
 - c) compare h^* to h in order to choose the proper equation between (6) or (7);
 - d) evaluate the unique root $\alpha_{h,p}^*$ by setting the function (6) or (7) equal to zero;

⁴ The graph is generated by extending an existing procedure in Matlab in order to assure that the minimum and maximum number of edges are satisfied and that the graph is simple and connected. The procedure assumes a probability equal to 0.5 of the existence of each edge. For more details on the original procedure, see the "random graph" construction routine developed by Gergana Bounova.

Step 3 evaluate the minimum upper bound α_h^* of ν_h between all the $\alpha_{h,p}^*$;

Step 4 pick the value of the upper bound α_{n-1}^* of ν_{n-1} ;

Step 5 set α_{n-1}^* equal to γ in order to evaluate the lower bound (4) and compare to other bounds existing in literature.

Table (1) shows the comparison between the bound (4) obtained for different number of vertices (from 4 to 20) and by assuming $p \in [2, 100]$ with the bounds

$$(11) \quad K(G) \geq \frac{(n-1)^2}{d_1}$$

in [7] and

$$(12) \quad K(G) \geq \frac{n}{d_1} \left[\frac{1}{1 + \frac{\sigma}{\sqrt{n-1}}} + \frac{(n-2)^2}{n-1 - \frac{\sigma}{\sqrt{n-1}}} \right]$$

in [3] with $\sigma = \sqrt{\frac{\text{tr}(P^2)}{n} - \left(\frac{\text{tr}(P)}{n}\right)^2}$. For any details related to these formulae see the corresponding references.

n	$K(G)$	bound (4)	bound (11)	bound (12)
4	3.3889	3.0810	3.0000	3.0027
5	5.0000	4.0625	4.0000	4.0036
6	11.1667	8.4797	8.3333	8.3490
7	8.3907	7.2381	7.2000	7.2034
8	11.0288	9.8302	9.8000	9.8028
9	11.7450	10.6909	10.6667	10.6692
10	12.7312	11.5873	11.5714	11.5731
11	15.7060	14.3006	14.2857	14.2874
12	16.6994	15.1393	15.1250	15.1269
13	16.1925	14.4126	14.4000	14.4019
14	23.1564	21.1378	21.1250	21.1268
15	27.3171	24.5148	24.5000	24.5025
16	27.5860	25.0119	25.0000	25.0020
17	25.2170	23.2801	23.2727	23.2738
18	28.2156	26.2798	26.2727	26.2738
19	26.4583	24.9280	24.9231	24.9238
20	34.9126	32.8240	32.8182	32.8191

Table 1
Lower bounds for $K(G)$

Notice that bound (4) is tighter than bound (11) and bound (12). By an inspection of the table, as n increases, slight differences are observed. At this regard, both the upper bound of ν_{n-1} used in (4) and the value of $1 + \frac{\sigma}{\sqrt{n-1}}$ used in (12) tend to one showing similar lower bounds for $K(G)$.

Furthermore it could be noticed that the low value of $K(G)$ is a result of the

procedure used to generate the graph. We observe indeed that the generated graphs have often a number of edges not so far from the half of its maximum value. Lowering the probability of existence of edges, we derive greater values of Kirchhoff Index, with comparable patterns for the bounds (i.e. bound (4) is tighter than bound (11) and bound (12)). Finally, it could be noteworthy that the upper bound of ν_{n-1} has been obtained by picking the minimum value observed among different p , allowing us to get the tighter one. Numerical analyses show that usually the upper bounds of ν_h with $h = 1, \dots, (n - 1)$ improve monotonically until a minimum value when p increases and after this threshold further values of p are not significant for the performed procedure. In particular, the minimum value is reached for lower values of p when ν_{n-1} is considered, while very high values of p are needed to achieve the best bound for ν_1 . Further development of this approach will involve how the bounds on other ν_h could be used in order to obtain sharper bounds for $K(G)$. We have indeed that a more specified localization could lead to better results.

References

- [1] Bianchi, M., and A. Torriero, *Some localization theorems using a majorization technique*, Journal of Inequalities and Applications **5** (2000), 433–446.
- [2] Bianchi M., A. Cornaro and A. Torriero, *Majorization under constraints and bounds of the second Zagreb index*, Mathematical Inequalities and Applications **16-2** (2013), 329–347.
- [3] Bianchi M., A. Cornaro, J.L. Palacios and A. Torriero, *Bounds for the Kirchhoff index via majorization techniques*, Journal of Mathematical Chemistry, (2012) online first.
- [4] Gutman I., and B. Mohar, *The quasi-Wiener and the Kirchhoff indices coincide*, J. Chem. Inf. Comput. Sci. **36** (1996) , 982–985.
- [5] Klein D. J., and M. Randić, *Resistance Distance*, J. Math. Chem **12** (1993), 81.
- [6] Marshall A. W., and I.Olkin I., *Inequalities: Theory of Majorization and Its Applications*, Academic Press (1979), London.
- [7] Palacios J.L., and J.M. Renom, *Broder and Karlin's formula for hitting times and the Kirchhoff index*, Int J Quantum Chem **111** (2011), 35–39.
- [8] Palacios J.L., and J.M. Renom, *Bounds for the Kirchhoff index of regular graphs via the spectra of their random walks*, Int J Quantum Chem **110** (2010), 1637–1641.
- [9] Xiao W., and I. Gutman, *Resistance distance and Laplacian spectrum*, Theor. Chem. Acc. **110** (2003), 284–289.
- [10] Wilson R. J., *Introduction to graph theory*, Addison Wesley (1996).
- [11] Zhu H. Y. , D.J. Klein and I. Lukovits, *Extensions of the Wiener number*, J. Chem. Inf. Comput. Sci. **36** (1996) , 420–428.

A heuristic approach for an integrated fleet-assignment, aircraft-routing and crew-pairing problem

Valentina Cacchiani¹

DEI, University of Bologna, Bologna, Italy

Juan-José Salazar-González²

DEIOC, Universidad de La Laguna, Tenerife, Spain

Abstract

This paper deals with the fleet-assignment, aircraft-routing and crew-pairing problems of an airline flying between Canary Islands. There are two major airports (bases). The company is subdivided in three operators. There are no flight during the night. A crew route leaves from and returns to the same base. An aircraft route starts from one base and arrive to the other base due to maintenance requirements. Therefore some crews must change aircrafts, which is an undesired operation. This paper presents a mathematical formulation based on a binary variable for each potential crew and aircraft route, and describes a column-generation algorithm for obtaining heuristic solutions. Computational results on real-world instances are given and compared to manual solutions by the airline.

Keywords: Integrated airline scheduling, heuristic algorithm, column generation, real-world application

¹ Email: valentina.cacchiani@unibo.it

² Email: jj.salaza@ull.es

Airline companies need solution approaches for solving complex logistic problems. Aircrafts and crews are expensive resources that need efficient utilization. They must be allocated to flights in a schedule of minimum cost subject to many rules so that each flight is covered exactly once. The rules are due to technical reasons or requirements forced by the company or by the crew unions, and insert a high degree of complexity in the logistic problems. For that reason in most of the cases the planning of the two resources are determined in a separate way. There are several attempts in the literature to optimize both resources in an integrated problem. Due to the hardness of the problem, the approaches in the literature propose heuristic procedures for solving it. See e.g. Cordeau et al [2], Mercier et al [6], Cohn and Barnhart [3], Klabjan et al [4], Mercier [5], Sandhu and Klabjan [10], Mercier and Soumis [7], Papadakos [8]) and Weide et al [11], among others. See e.g. Belobaba et al [1] for a survey.

The main contribution of our paper is to develop a heuristic algorithm to solve real-world instances. These real-world instances have been considered in a research project founded by a major airline company in Canary Islands. Salazar [9] presented a mathematical model based on two-index variables for the integrated aircraft-routing and crew-pairing problem. The current paper shows an alternative model based on exponentially many variables, corresponding to feasible routes for the aircrafts and for the crews. It also presents a column-generation approach for solving the integrated problem. Section 1 describes the aircraft-routing and crew-pairing problems of our real-world application. The two problems are mathematically formulated in a single model in Section 2. A column-generation approach is described in Section 3 and computational results are discussed in Section 4.

1 Problem description

This section describes the aircraft-routing and crew-pairing problems of the airline company motivating this research. The company has 18 aircrafts to serve around 100-150 flights every day between 13 airports. The company is divided in three operators. Each operator owns some aircrafts and has some crews. Although all the aircrafts are identical (ATR72), the crews of each operator have special rules. A crew of an operator cannot operate an aircraft of another operator. The flights are not pre-assigned to operators, thus the assignment of each flight to an operator is also part of the optimization problem. This problem is usually known as *fleet assignment*.

Due to the geographical proximity between the airports, most of the flights

take 30 minutes. No flight is scheduled during the night (i.e., between 11pm and 7am). The airport closest to the home of a crew is called *base* of that crew. Among the thirteen airports, only two of them are bases: TFN and LPA. A crew route must leave from and return to the same base.

For each operator, the numbers of aircrafts and crews available in each base at the beginning of each day are given. Aircrafts have different types of maintenance and we emphasize the two operations affecting our problem. Long-term maintenance is scheduled during a long-term period (a year), and take several days. During this maintenance, the corresponding aircrafts are not available. Short-term maintenance takes place every two days, and is done during the night. To perform these maintenances the aircraft must be in LPA, which is the airport with all the necessary equipments. As a consequence, an aircraft that is in an airport different than LPA one night, it must be in LPA the night after.

A flight is determined by the departure time, the departure airport, the arrival time and the arrival airport. Two flights can be consecutively in the route of an aircraft when the arrival airport of the first flight is equal to the departure airport of the second flight, and the departure time of the second flight is at least 20 minutes after the arrival time of the first flight. The difference between the departure time of the second flight minus the arrival time of the first flight is called *connection time* (also *plane-turn time* or *sit-connection time*). An aircraft is not allowed to go from one airport to another unless it is serving a commercial flight. Two flights can be consecutively in the route of a crew when the arrival airport of the first flight is equal to the departure airport of the second flight, and the connection time is between 20 minutes and 3 hours. Only one crew can be in each aircraft, thus it is not possible to move a crew from one airport to another unless it is the one serving the flight.

Given a crew and two consecutive flights in the route of this crew, we say that there is an *aircraft change* when the aircrafts assigned to these flights are different. Analogously, given an aircraft and two consecutive flights in the route of this aircraft, we said that there is a *crew change* when the crew assigned to these flights are different. Aircraft changes and crew changes are not allowed when the connection time between two flights is smaller than 30 minutes. Aircraft changes are undesired because they increase the propagation of delays.

An aircraft route does not have constraints on the time duration nor on the number of flights in the sequence. A crew route, instead, has constraints both on the time duration and on the number of flights.

Due to the page limitation of this article, we do not give other details of the real-world problem. However, with the above description we already have the main features of our optimization problem.

2 Mathematical Formulation

In order to model the described problem, we introduce two directed acyclic graphs: the first one, called $G^a = (N, A^a)$, is related to the aircrafts and the second one, called $G^c = (N, A^c)$, is related to the crews. The set N consists of the union of the set of nodes N^f representing the flights and the set of nodes N^b representing the bases (i.e. TFN and LPA). A pair of nodes (i, j) ($i, j \in N^f$) is an arc in A^c (respectively A^a) when they represents two flights than can be consecutively in the route of a crew (respectively aircraft), or when $i \in N^b$ represents the departure airport of flight $j \in N^f$, or when $j \in N^b$ represents the arrival airport of flight $i \in N^f$. Note that A^c is a subset of A^a since connection times larger than 3 hours are allowed to aircrafts but not to crews. We denote by A_s^c the arcs with connection times shorter than 30 minutes.

The operators are represented by the set K . For each operator $k \in K$ and each base $l \in N^b$, let n_c^{kl} and n_a^{kl} be the number of crews and aircrafts, respectively, available of k in l at the beginning of the day.

We define \mathcal{R}_c^{kl} the set of feasible routes in graph $G^c = (N, A^c)$ for a crew of operator $k \in K$ in base $l \in N^b$. A route is feasible if it satisfies constraints on time duration of the route and on the maximum number of flights it executes. In addition, it must respect the constraints on the departure and arrival airports and on the sequencing of flights (described above). We define \mathcal{R}_a^{kl} the set of feasible routes $G^a = (N, A^a)$ for the aircraft of operator $k \in K$ starting from base $l \in N^b$. A feasible route must satisfy the constraints on the departure and arrival airports (described above) in order to respect maintenance requirements. When convenient for the notation, a route R is a sequence of arcs or a sequence of nodes. Each crew route R has an associated cost c_R which depends on the connection times between consecutive flights and on the operator. To model the multi-criteria nature of the problem with a single objective function we consider four weights:

- α : weight on the sum of connection times in the crew routes.
- β : weight on the number of crew routes.
- γ : weight on the number of aircraft routes.
- δ : weight on the number of aircraft changes in the crew routes.

Each solution of the problem can be represented with the following binary variables. For each crew route $R \in \mathcal{R}_c^{kl}$ let x_R be a 0-1 variable such that $x_R = 1$ if and only if the route R is assigned to a crew. For each aircraft route $R \in \mathcal{R}_a^{kl}$ let y_R be a 0-1 variable such that $y_R = 1$ if and only if the route R is assigned to an aircraft. For each arc $(i, j) \in A^c \setminus A_s^c$ let z_{ij} be a 0-1 variable such that $z_{ij} = 1$ if and only if an aircraft change occurs between flights i and j ($i, j \in N^f$).

A mathematical formulation for the problem minimizes

$$\sum_{k \in K, l \in N^b, R \in \mathcal{R}_c^{kl}} (\alpha \cdot c_R + \beta) \cdot x_R + \gamma \cdot \sum_{k \in K, l \in N^b, R \in \mathcal{R}_a^{kl}} y_R + \delta \cdot \sum_{(i,j) \in A^c \setminus A_s^c} z_{ij} \quad (1)$$

subject to constraints on the crew variables:

$$\sum_{k \in K, l \in N^b, R \in \mathcal{R}_c^{kl}: i \in R} x_R = 1 \quad \forall i \in N^f \quad (2)$$

$$\sum_{R \in \mathcal{R}_c^{kl}} x_R \leq n_c^{kl} \quad \forall k \in K, l \in N^b \quad (3)$$

$$x_R \in \{0, 1\} \quad \forall k \in K, l \in N^b, R \in \mathcal{R}_c^{kl} \quad (4)$$

constraints on the aircraft variables:

$$\sum_{l \in N^b, R \in \mathcal{R}_a^{kl}: i \in R} y_R = \sum_{l \in N^b, R \in \mathcal{R}_c^{kl}: i \in R} x_R \quad \forall k \in K, i \in N^f \quad (5)$$

$$\sum_{R \in \mathcal{R}_a^{kl}} y_R \leq n_a^{kl} \quad \forall k \in K, l \in N^b \quad (6)$$

$$y_R \in \{0, 1\} \quad \forall k \in K, l \in N^b, R \in \mathcal{R}_a^{kl} \quad (7)$$

and constraints on the aircraft changes:

$$\sum_{l \in N^b, R \in \mathcal{R}_c^{kl}: (i,j) \in R} x_R = \sum_{l \in N^b, R \in \mathcal{R}_a^{kl}: (i,j) \in R} y_R \quad \forall k \in K, (i, j) \in A_s^c \quad (8)$$

$$\sum_{k \in K, l \in N^b, R \in \mathcal{R}_c^{kl}: (i,j) \in R} x_R \leq \sum_{k \in K, l \in N^b, R \in \mathcal{R}_a^{kl}} y_R + z_{ij} \quad \forall (i, j) \in A^c \setminus A_s^c \quad (9)$$

$$z_{ij} \geq 0 \quad \forall (i, j) \in A^c \setminus A_s^c. \quad (10)$$

Constraints (2) impose that each flight must be assigned to a crew. Constraints (3) require to respect the maximum number of crews available for each operator at each base. Constraints (5) are linking constraints between crew and aircraft routes. They impose that each flight is executed by a crew and an aircraft of the same operator. Constraints (6) impose not to use more

than the maximum number of aircrafts available for each operator at each base. Constraints (8) concern the crew and aircraft changes for the short connections: if an aircraft route (crew route resp.) executes flights i and j such that their connecting time is shorter than 30 minutes, then a crew route (aircraft route resp.) must execute these flights. I.e. crew change (aircraft change resp.) is forbidden for connections shorter than 30 minutes. Finally, constraints (9) are used to count the aircraft changes: given two flights i and j , if they are executed in sequence by a crew route, then either an aircraft route executes them in sequence or there is an aircraft change.

3 Solution Approach

In this section, we present our solution method, which is based on the Linear Programming (LP) solution of model (1)-(10). Model (1)-(10) has exponentially many variables x_R and y_R ($R \in \mathcal{R}$). We solve its LP-relaxation by column generation on both crew routes and aircraft routes. The pricing problem calls for finding routes of negative reduced cost and corresponds to an Elementary Shortest Path with Resource Constraints (ESPRC) to be computed on the acyclic graphs defined above. Once we have obtained the LP-solution, we execute a depth-first truncated branch and price until we find the first integer solution.

At each node of the decisional tree, we construct the values of the flows along the arcs in graph $G^c = (N, A^c)$ for the crews, based on the values of the route variables $x_R \in \mathcal{R}$. Then, we select to branch on the node $i \in N^f$ of graph G^c such that it has the highest fractional flow on its outgoing arcs (i.e. the highest number of outgoing arcs with positive flow). We consider a binary branching. In particular, in order to keep the decisional tree balanced, we apply the following branching rule. We order the outgoing arcs (i, j) according to the time instant associated to the departure time of the flight j . Let f be the total value of the flow from node i along its outgoing arcs. In the first generated child, we forbid the use of the first set of outgoing arcs (in the order) such that the sum of their flows is at least $f/2$. In addition, we forbid to use node i as a terminal node for the route. In the second generated child we forbid the use of the remaining set of outgoing arcs. At each generated child, we forbid to use and to generate routes that contain forbidden arcs.

At each node of the decisional tree, column generation is applied again for both crews and aircrafts. A depth-first strategy is used in order to dive towards an integer solution. The process is iterated until an integer feasible solution is found.

4 Computational Results

To measure the performance of the approach described in the previous section we have considered a set of real-world instances. They correspond to the first week of September 2012. We have considered the following values for the weights: $\alpha = 2$ for the connections between 20 and 30 minutes, $\alpha = 1$ for the connections between 30 minutes and one hour, $\alpha = 3$ for the connections between one and two hours, $\alpha = 5$ for the connections between two and three hours; $\beta = 1000$, $\gamma = 1000$ and $\delta = 10$. The presented algorithm has been implemented in C. All tests were performed on a pc CORE i5-2400 3.10GHZ, 16GB Ram and Cplex 12.4 was used to solve the LP-relaxation models.

Table 1 reports, for each instance, the number of flights (#f) to be scheduled, the solution found by our approach (Col-gen Heur) and the solution generated by the airline (Manual). In particular, for each method, it shows the number of aircrafts (#a), the number of crews (#cr) and the number of aircraft changes (#ch) used in the solution. In addition, for our solution, the table reports the computing time expressed in seconds (time) and the percentage gap (gap%) from the lower bound.

As it can be seen, our method is able to compute heuristic solutions with small gaps in reasonable computing time. In addition, with respect to the solutions computed by the airline, our method is able to reduce the number of aircrafts and/or crews for all instances.

Inst.	#f	Col-gen Heur					Manual		
		#a	#cr	#ch	time	gap%	#a	#cr	#ch
1	102	13	22	6	437	0.34	14	24	8
2	140	15	28	7	3634	0.22	17	28	7
3	130	14	25	6	4212	0.44	15	26	6
4	124	13	24	5	2341	0.27	13	25	5
5	124	13	24	5	2940	0.38	14	25	4
6	128	13	25	5	4543	0.34	13	26	4
7	150	15	28	7	9913	0.26	15	30	5

Table 1

Comparison between the proposed method and the airline solutions.

References

- [1] P. Belobaba, A. Odoni, C. Barnhart. "The Global Airline Industry." John Wiley & Sons, West Sussex, United Kingdom, (2009). ISBN 978-0-470-74077-4.
- [2] J.F. Cordeau, G. Stojkovic, F. Soumis, J. Desrosiers. "Benders decomposition for simultaneous aircraft routing and crew scheduling." *Transportation Science* 35 (2001) 375-388.
- [3] A.M. Cohn, C. Barnhart. "Improving crew scheduling by incorporating key maintenance routing decisions." *Operations Research* 51 (2003) 387-396.
- [4] D. Klabjan, E. L. Johnson, G. L. Nemhauser. "Airline crew scheduling with time windows and plane count constraints." *Transportation Science* 36 (2002) 337-348.
- [5] A. Mercier. "A Theoretical Comparison of Feasibility Cuts for the Integrated Aircraft-Routing and Crew-Pairing Problem." *Transportation Science* 42 (2008) 87–104.
- [6] A. Mercier, J.F. Cordeau, F. Soumis. "A computational study of Benders decomposition for the integrated aircraft routing and crew scheduling problem." *Computers & Operations Research* 32 (2005) 1451-1476.
- [7] A. Mercier, F. Soumis. "An integrated aircraft routing, crew scheduling and flight retiming model." *Computers & Operations Research* 34 (2007) 2251-2265.
- [8] N. Papadakos. "Integrated airline scheduling." *Computers & Operations Reserch* 36 (2009) 176–175.
- [9] J.J. Salazar-González. "Solving a 2-depot driver-and-vehicle routing problem." Conference ROUTE 2011, Sitges, May 31st-June 3th, 2011.
<http://www.uv.es/route2011/Salazar.pdf>
- [10] R. Sandhu, D. Klabjan. "Integrated Airline Fleeting and Crew-Pairing Decisions." *Operations Research* 55 (2007) 439–456.
- [11] O. Weide, E. Ryan, M. Ehrgott. "An iterative approach to robust and integrated aircraft routing and crew scheduling." *Computers & Operations Research* 37 (2010) 833–844.

Intelligent variable neighbourhood search for the minimum labelling spanning tree problem*

S. Consoli^{a,1}, J. A. Moreno Pérez^b and N. Mladenović^c

^a *Joint Research Centre, European Commission, Via Enrico Fermi 2749, 21027 Ispra (VA), Italy*

^b *DEIOC, IUDR, Universidad de La Laguna, Facultad de Matemáticas, 4a planta Astrofísico Francisco Sánchez s/n, 38271 Santa Cruz de Tenerife, Spain*

^c *School of Information Systems, Computing and Mathematics, Brunel University, Uxbridge, Middlesex, UB8 3PH, United Kingdom*

Abstract

Given a connected, undirected graph whose edges are labelled (or coloured), the minimum labelling spanning tree (MLST) problem seeks a spanning tree whose edges have the smallest number of distinct labels (or colours). In recent work, the MLST problem has been shown to be NP-hard and some effective heuristics have been proposed and analyzed. In a currently ongoing project, we investigate an intelligent optimization algorithm to solve the problem. It is obtained by the basic Variable Neighbourhood Search heuristic with the integration of other complements from machine learning, statistics and experimental algorithmics, in order to produce high-quality performance and to completely automate the resulting optimization strategy. Computational experiments show that the proposed metaheuristic has high-quality performance for the MLST problem and it is able to obtain optimal or near-optimal solutions in short computational running time.

Keywords: Combinatorial optimization, graphs and networks, minimum labelling spanning trees, intelligent optimization, variable neighbourhood search.

¹ Corresponding author:

 +39 329 7472128 -  sergio.consoli@jrc.ec.europa.eu; sergio.consoli@yahoo.it

* This work has been partially funded by the Spanish Ministry of Economy and Competitiveness (project TIN2012-32608).

1 Preliminary discussion

In a currently ongoing project, we investigate a new possibility for solving the *minimum labelling spanning tree* (MLST) by an intelligent optimization algorithm. The minimum labelling spanning tree problem is a challenging combinatorial problem [3]. Given an undirected graph with labelled (or coloured) edges as input, with each edge assigned with a single label, and a label assigned to one or more edges, the goal of the MLST problem is to find a spanning tree with the minimum number of labels (or colours). Besides applications in telecommunications network design [13] and data compression [5], such a model is very useful in multimodal transportation systems [11]. A multimodal transportation network can be represented by a graph where each edge is assigned a label, denoting a different company managing that link. In this context, it is often desirable to provide a complete service between the nodes of the network, without cycles, by using the minimum number of companies [14]. Thus, the aim is to find a spanning tree of the graph using the minimum number of labels. This spanning tree may reduce the construction cost and the overall complexity of the network.

The MLST problem can be formulated as a network or graph problem [6]. We are given a labelled connected undirected graph $G = (V, E, L)$, where V is the set of nodes, E is the set of edges, and L is the set of labels. The purpose is to find a spanning tree T of G such that $|L_T|$ is minimized, where L_T is the set of labels used in T . Although a solution to the MLST problem is a spanning tree, it is easier to work firstly in terms of feasible solutions. A feasible solution is defined as a set of labels $C \subseteq L$, such that all the edges with labels in C represent a connected subgraph of G and span all the nodes in G . If C is a feasible solution, then any spanning tree of C has at most $|C|$ labels. Moreover, if C is an optimal solution, then any spanning tree of C is a minimum labelling spanning tree. Thus, in order to solve the MLST problem we first seek a feasible solution with the least number of labels [15].

The MLST problem was first introduced in [3]. The authors also proved that it is an NP-hard problem and provided a polynomial time heuristic, the Maximum Vertex Covering Algorithm (MVCA), successively improved in [10]. The procedure starts with an empty graph. Successively, it adds at random one label from those labels that minimize the number of connected components, until only one connected component is left. Other heuristics for the MLST problem have been proposed in the literature [15,2,6,1,4,7].

The aim of this paper is to present preliminary results concerning the design of a novel heuristic solution approach for the MLST problem, with the

goal of obtaining high-quality performance. The proposed strategy is an intelligent hybrid metaheuristic, obtained by combining Variable Neighbourhood Search (VNS) [8] and Simulated Annealing (SA) [9], with the integration of other complements in order to improve the effectiveness and robustness of the optimization process, and to completely automate the resulting strategy.

2 Complementary Variable Neighbourhood Search

The first extension that we introduce for the MLST problem is a local search mechanism that is inserted at top of the Variable Neighbourhood Search metaheuristic [8]. The resulting local search method is referred to as *Complementary Variable Neighbourhood Search* (Co-VNS).

For our implementation, given a labelled graph $G = (V, E, L)$, with n vertices, m edges, ℓ labels, each solution is encoded as a binary string, i.e. $C = (c_1, c_2, \dots, c_\ell)$ where $c_i = 1$ if label i is in solution C , $c_i = 0$ otherwise, $\forall i = 1, \dots, \ell$. Given a solution C , Co-VNS extracts a solution from the *complementary space* of C , and then replaces the current solution with the solution extracted. The complementary space of a solution C is defined as the set of all the labels that are not contained in C , that is $(L\Delta C)$. To yield the solution, Co-VNS applies a constructive heuristic, such as the MVCA [3,10], to the subgraph of G with labels in the complementary space of the current solution. Note that Co-VNS stops if either a feasible solution is obtained (i.e. a single connected component is obtained), or the set of unused colours contained in the complementary space is empty (i.e. $(L\Delta C) = \emptyset$), producing a final infeasible solution. Then, the basic VNS is applied in order to improve the resulting solution.

The iterative process of selection of a new incumbent solution from the complementary space of the current solution if no improvement has occurred, is aimed at increasing the diversification capability of the basic VNS for the MLST problem. When the local search is trapped at a local minimum, Co-VNS extracts a feasible complementary solution which lies in a very different zone of the search domain, and is set as new incumbent solution for the local search. This new starting point allows the algorithm to escape from the local minimum where it is trapped, producing an immediate peak of diversification.

3 Intelligent optimization algorithm

In order to seek further improvements and to automate on-line the search process, Complementary Variable Neighbourhood Search has been modified

by replacing the inner local search based on the deterministic MVCA heuristic with a *probability-based local search* inspired by a “Simulated Annealing cooling schedule” [9], with the view of achieving a proper balance between intensification and diversification capabilities. The strength of this probabilistic local search is tuned by an automated process which allows the intelligent strategy to adapt on-line to the problem instance explored and to react in response to the search algorithm’s behavior [12]. The resulting metaheuristic represents the intelligent optimization algorithm that we propose.

The probability-based local search is another version of the MVCA heuristic, but with a probabilistic choice of the next label to be added. It extends the basic greedy construction criterion of the MVCA by allowing moves to worse solutions. Starting from an initial solution, successively a candidate move is randomly selected; this move is accepted if it leads to a solution with a better objective function value than the current solution, otherwise the move is accepted with a probability that depends on the deterioration, Δ , of the objective function value.

Following the SA criterion, the acceptance probability is computed according to the Boltzmann function as $\exp(-\Delta/T)$, using the temperature (T) as control parameter. The value of T is initially high, which allows many worse moves to be accepted, and is gradually reduced following a specific geometric cooling schedule: $T_{k+1} = \alpha \cdot T_k$, where $T_0 = |Best_C|$ and $\alpha = 1/|Best_C| \in [0, 1]$, with $Best_C$ being the current best solution, and $|Best_C|$ its number of colours. This cooling law is very fast for the MLST problem, yielding a good balance between intensification and diversification. Furthermore, thanks to its self-tuning parameters setting, which is guided automatically by the best solution $Best_C$ without requiring any user-intervention, the algorithm is allowed to adapt on-line to the problem instance explored and to react in response to the search algorithm’s behavior [12].

The aim of the probabilistic local search is to allow, with a specified probability, worse components with a higher number of connected components to be added to incomplete solutions. Probability values assigned to each label are inversely proportional to the number of components they give. So the labels with a lower number of connected components will have a higher probability of being chosen. Conversely, labels with a higher number of connected components will have a lower probability of being chosen. Thus, the possibility of choosing less promising labels is allowed. Summarizing, at each step the probabilities of selecting labels giving a smaller number of components will be higher than the probabilities of selecting labels with a higher number of components. Moreover, these differences in probabilities increase step by

step as a result of the reduction of the temperature for the adaptive cooling schedule. It means that the difference between the probabilities of two labels giving different numbers of components is higher as the algorithm proceeds. The probability of a label with a high number of components will decrease as the algorithm runs and will tend to zero. In this sense, the search becomes MVCA-like. A simple VNS implementation which uses the probabilistic local search as constructive heuristic has been tested. However, the best results were obtained by combining Complementary Variable Neighbourhood Search with the probabilistic local search, resulting in the intelligent Variable Neighbourhood Search that we propose. Note that the probabilistic local search is applied both in Co-VNS, to obtain a solution from the complementary space of the current solution, and in the inner local search phase, to restore feasibility by adding labels to incomplete solutions.

4 Computational results

To test the performance of the proposed intelligent Variable Neighbourhood Search (Int-VNS), we compare it with the Complementary Variable Neighbourhood Search (Co-VNS) and with other popular MLST algorithms from the literature. In particular we consider an exact method (EXACT) and some classic MLST metaheuristics: Modified Genetic Algorithm (MGA) [15], Pilot Method (PILOT) [2], Greedy Randomized Adaptive Search Procedure (GRASP) [6] and basic Variable Neighbourhood Search (VNS) [6]. In our experiments, we consider 12 different datasets, each one containing 10 instances of the problem (yielding a total of 120 instances), including instances with number of vertices, $n = 200$, number of labels, $\ell = 0.25 \cdot n, 0.5 \cdot n, n, 1.25 \cdot n$. The number of edges, m , is obtained indirectly from the density d , whose values are chosen to be 0.8, 0.5, and 0.2. The complexity of the instances increases with the dimension of the graph (i.e. increasing of ℓ), and the reduction in the density of the graph. For each dataset, solution quality is evaluated as the average objective value among the 10 problem instances. A maximum allowed CPU time of 60 s, that we call *max-CPU-time*, is chosen as the stopping condition for all the metaheuristics, determined experimentally with respect to the dimension of the considered datasets. For the Modified Genetic Algorithm, we set the number of generations for each instance such that the computations take approximately *max-CPU-time*.

Our results are reported in Table 1. All the computations have been made on a Pentium Centrino microprocessor at 2.0 GHz with 512 MB RAM. All the metaheuristics run for *max-CPU-time* and, in each case, the best solution

is recorded. The computational times reported in the table are the average times at which the best solutions are obtained, except in the case of the exact method, where the exact solution is reported unless a single instance computes for more than 3 hours of CPU time (in this case a not found status (NF) is reported). All the reported times have precision of ± 5 ms. The performance of an algorithm can be considered better than another one if either it obtains

Parameters			Average objective function values						
n	ℓ	d	EXACT	PILOT	MGA	GRASP	VNS	Co-VNS	Int-VNS
50	0.8	2	2	2	2	2	2	2	2
	0.5	2.2	2.2	2.2	2.2	2.2	2.2	2.2	2.2
	0.2	5.2	5.2	5.2	5.2	5.2	5.2	5.2	5.2
100	0.8	2.6	2.6	2.6	2.6	2.6	2.6	2.6	2.6
	0.5	3.4	3.4	3.4	3.4	3.4	3.4	3.4	3.4
	0.2	NF	8.3	8.3	8.1	7.9	8	7.9	7.9
200	0.8	4	4	4	4	4	4	4	4
	0.5	NF	5.5	5.4	5.4	5.4	5.4	5.4	5.4
	0.2	NF	12.4	12.4	12.2	12	12.1	12	12
250	0.8	4	4	4	4.1	4	4.1	4	4
	0.5	NF	6.3	6.3	6.3	6.3	6.3	6.3	6.3
	0.2	NF	13.9	14	13.9	13.9	13.9	13.9	13.9
TOTAL:			-	69.8	69.8	69.4	68.9	69.2	68.9
Parameters			Computational times (milliseconds)						
n	ℓ	d	EXACT	PILOT	MGA	GRASP	VNS	Co-VNS	Int-VNS
50	0.8	15.6	90.7	26.5	7.8	3.1	0	0	0
	0.5	15.9	164.1	68.8	9.2	12.4	15.7	3.1	
	0.2	4.3×10^3	320.4	326.6	20.3	228.6	205.4	190.4	
100	0.8	87.3	876.5	139.3	31.3	120.2	446.1	21.8	
	0.5	727.1	1.2×10^3	1.6×10^3	76.2	71.9	173.1	54.5	
	0.2	NF	1.3×10^3	2.2×10^3	513.3	1.7×10^3	7.9×10^3	1.6×10^3	
200	0.8	22.1×10^3	5.9×10^3	204.6	36.4	27.7	59.3	17.9	
	0.5	NF	5.6×10^3	16.1×10^3	552.1	1.1×10^3	2.5×10^3	463.3	
	0.2	NF	5×10^3	12.7×10^3	7.2×10^3	12.8×10^3	2.3×10^3	11.5×10^3	
250	0.8	19.2×10^3	9.1×10^3	2.2×10^3	4.8×10^3	1.3×10^3	1.8×10^3	1.1×10^3	
	0.5	NF	8.4×10^3	17.6×10^3	388.5	2.3×10^3	2.2×10^3	1.1×10^3	
	0.2	NF	8×10^3	26.4×10^3	1.4×10^3	2.5×10^3	1.5×10^3	1.3×10^3	
TOTAL:			-	45.9×10^3	79.6×10^3	15.1×10^3	22.2×10^3	19.1×10^3	17.3×10^3

Table 1
Computational results for $n = 200$, $\ell = 0.25n$, $0.5n$, n , $1.25n$

a smaller average objective value, or an equal average objective value but in a shorter computational running time.

Looking at Table 1, it is shown that Int-VNS is able to obtain very high performances for the MLST problem, as it always obtained the solutions with the best quality in shorter computational running time. The motivation to introduce a high diversification capability in Int-VNS is to obtain high performances also in complex problem instances. Inspection of the table shows that this aim is achieved, as Int-VNS outperformed the other algorithms in terms of solution quality and, in general, computational running time. It was slower in some problem instances with respect to GRASP (see for example the instance $[n = 200, \ell = 200, d = 0.2]$), but because it obtained a solution of better quality. When Int-VNS obtained the same performance of the others with respect to solution quality, as with VNS, it was considerable faster.

5 Summary and Outlook

Concerning the achieved optimization strategy, the whole approach seems to be highly promising for the MLST problem. Ongoing investigation will consist in a statistical comparison of the resulting strategy against the best MLST algorithms in the literature, considering in addition larger instances of the problem and investigating in depth the features of the explored algorithms, in order to characterize with more detail the improvements obtained by the proposed intelligent Variable Neighbourhood Search.

References

- [1] Brüggemann, T., J. Monnot and G. J. Woeginger, *Local search for the minimum label spanning tree problem with bounded colour classes*, Operations Research Letters **31** (2003), pp. 195–201.
- [2] Cerulli, R., A. Fink, M. Gentili and S. Voß, *Metaheuristics comparison for the minimum labelling spanning tree problem*, in: B. L. Golden, S. Raghavan and E. A. Wasil, editors, *The Next Wave on Computing, Optimization, and Decision Technologies*, Springer-Verlag, New York, 2005 pp. 93–106.
- [3] Chang, R. S. and S. J. Leu, *The minimum labelling spanning trees*, Information Processing Letters **63** (1997), pp. 277–282.
- [4] Chwatal, A. M. and G. R. Raidl, *Solving the minimum label spanning tree problem by ant colony optimization*, in: *Proceedings of the 7th International*

Conference on Genetic and Evolutionary Methods (GEM 2010), Las Vegas, Nevada, 2010.

- [5] Chwatal, A. M., G. R. Raidl and O. Dietzel, *Compressing fingerprint templates by solving an extended minimum label spanning tree problem*, in: *Proceedings of the 7th Metaheuristics International Conference (MIC 2007)*, Montreal, Canada, 2007.
- [6] Consoli, S., K. Darby-Dowman, N. Mladenović and J. A. Moreno-Pérez, *Greedy randomized adaptive search and variable neighbourhood search for the minimum labelling spanning tree problem*, European Journal of Operational Research **196** (2008), pp. 440–449.
- [7] Consoli, S. and J. A. Moreno-Pérez, *Solving the minimum labelling spanning tree problem using hybrid local search*, in: *Proceedings of the mini EURO Conference XXVIII on Variable Neighbourhood Search*, Electronic Notes in Discrete Mathematics **39**, Herceg Novi, Montenegro, 2012, pp. 75–82.
- [8] Hansen, P. and N. Mladenović, *Variable neighbourhood search: Principles and applications*, European Journal of Operational Research **130** (2001), pp. 449–467.
- [9] Kirkpatrick, S., C. D. Gelatt and M. P. Vecchi, *Optimization by simulated annealing*, Science **220** (1983), pp. 671–680.
- [10] Krumke, S. O. and H. C. Wirth, *On the minimum label spanning tree problem*, Information Processing Letters **66** (1998), pp. 81–85.
- [11] Miller, J. S., “Multimodal statewide transportation planning: A survey of state practices,” Transportation Research Board, National Research Council, Virginia, US, 2006.
- [12] Osman, I. H., *Metastrategy simulated annealing and tabu search algorithms for the vehicle routing problem*, Annals of Operations Research **41** (1993), pp. 421–451.
- [13] Tanenbaum, A. S., “Computer networks,” Prentice-Hall, Englewood Cliffs, New Jersey, 1989.
- [14] Van-Nes, R., “Design of multimodal transport networks: A hierarchical approach,” Delft University Press, 2002.
- [15] Xiong, Y., B. Golden and E. Wasil, *Improved heuristics for the minimum labelling spanning tree problem*, IEEE Transactions on Evolutionary Computation **10** (2006), pp. 700–703.

Fitness Landscape Analysis for Scalable Multicast RRM Problem in Cellular Network

Qing Xu^{a,1}, Frédéric Lassabe^{a,1}, Hakim Mabed^{b,2} and Alexandre Caminada^{a,1}

^a *IRTES-SeT, Université de Technologie de Belfort-Montbéliard,
13 Rue Thierry Mieg 90010 Belfort, France*

^b *FEMTO-ST, Université de Franche-Comté,
1 Rue Claude Goudimel 25030 Montbéliard, France*

Abstract

This paper aims to solve the Radio Resource Management (RRM) problem for Multimedia Broadcast Multicast Service (MBMS) system in cellular network. We develop a flexible model to perform dynamic radio resource allocation for MBMS service by using metaheuristic approach. We conduct fitness landscape analysis to study the characteristics of the proposed model, which helps us to select appropriate search strategy. Simulation results show that the proposed algorithm provides better performance than existing algorithms.

Keywords: fitness landscape, metaheuristic approach, multimedia multicast, radio resource management.

¹ Email: qing.xu, frederic.lassabe, alexandre.caminada@utbm.fr

² Email: hakim.mabed@pu-pm.univ-fcomte.fr

1 Introduction

The MBMS service over UMTS could be carried by PTM or PTP mode in a base station. A cell is defined to be the geographic coverage of a UMTS base station. In PTM mode, service is transmitted by a forward access channel (FACH) covering the whole cell. Each FACH needs one channel code serving large amount of users, but may waste power when users are close to the base station [4]. The PTP mode uses dedicated channel (DCH) or shared channel (HS-DSCH). Each DCH needs one channel code serving one dedicated user. A HS-DSCH occupies up to 15 channel codes. PTP mode is more efficient than PTM but the served user number is limited. In UMTS where the radio resources (transmit power and channelization codes) are limited, the transmission mode selection is crucial. To improve the efficiency, **MBMS Power Counting (MPC)** [2] proposes to switch from PTP to PTM when pre-estimated power exceeds an operator-defined threshold, and vice versa. MPC is simple to implement but has very limited flexibility since it does not consider multimedia scalability. **Dual Transmission Mode (DTM)** allows the mixture of PTP and PTM mode for a same service [5]. However, simulation shows that it is only beneficial for up to 5 users [10]. **Scalable FACH Transmission (S-FACH)** [6] transmits scalable encoded service through PTM mode. Services are divided into single layer (SL) and multiple layer (ML) transmission schemes, the latter is divided into several flows. The basic flow is transmitted with lower bit rate while the advanced flows with smaller but fixed geographical coverages. However, S-FACH is still based on the fixed coverage so that the resource allocation can not be optimal.

Consider the quite dynamic property of the RRM problem, all mentioned existing approaches provide only limited flexibility. Therefore, the power and date rate allocation can not be optimal. To address this problem, we propose a Flexible Radio Resource Management Model (F2R2M) by using metaheuristic approach. This model combines two objectives: to satisfy the QoS requirement of multicast service and to minimize the power consumption while avoiding saturation of channel code occupation. To understand the behavior of the proposed model, we conduct fitness landscape analysis on the model under five different scenarios. Based on the analysis results, we evaluate two search operators, and compare the performance with existing approaches.

This paper is structured as follows. The proposed model is formulated in [section 2](#). The fitness landscape analysis and the simulation results are given in [section 3](#). The conclusion is drawn in [section 4](#).

2 Model Description

We focus on multicast transmission within one cell c , which is the geographic coverage of a base station. The input variables are defined as follows: $T(c) = \{t_1, \dots, t_{N_t}\}$ is a set of users located in c , and their instantaneous geometry coordinates $C(c) = \{(x_1, y_1), \dots, (x_{N_t}, y_{N_t})\}$. A set of services $S(c) = \{s_1, \dots, s_{N_s}\}$ and their flows $F(s_i) = \{f_{s_i,0}\}$ or $\{f_{s_i,1}, f_{s_i,2}[f_{s_i,3}]\}$ are transmitted in c . $f_{s_i,0}$ indicates s_i is SL transmission scheme. $f_{s_i,j}(j > 0)$ is the sublayers of ML scheme service, $f_{s_i,1}$ is the basic flow and $f_{s_i,j}(j > 1)$ is the advanced flow which is optional for quality improvement. Each user group subscribing to the same service is called a multicast group: $Dist(s_i) = \{t_n, t_m, \dots\}$.

F2R2M supports the co-existing of PTM and PTP modes for each flow. It partitions $Dist(s)$ into four disjointed sets: users served through a FACH $UE_{fach}(f_{s,j})$; users transferred through DCHs $UE_{dch}(f_{s,j})$; users sharing HS-DSCH $UE_{hs}(f_{s,j})$ and non-served users $UE_{noch}(f_{s,j})$. The decision of user set follows two principles: 1) to guarantee the service coverage, all users in multicast group should be selected to receive $f_{s_i,0}$ or $f_{s_i,1}$, 2) in ML transmission, the advanced flow is sent only when a lower flow has been received by users. The partition of user depends on channel characteristics: 1) FACH can be listened by all users within its coverage, hence $UE_{fach}(f_{s,j})$ includes the nearest group of users in $Dist(s)$, with the distance from the base station under a threshold d_{thr} . d_{thr} is dynamically determined during search procedure. 2) the users in $Dist(f_{s,j})$ farther than d_{thr} , are assigned to $UE_{dch}(f_{s,j})$ or $UE_{hs}(f_{s,j})$. Consequently, according to the user partition and flow bandwidth, available channel code(s) is associated with a nonempty user set. When no channel code is available for a given user, this user is switched to $UE_{noch}(f_{s,j})$.

A solution is defined as variable x , which is a matrix with N_t (number of flows) rows and N_f (number of users) columns:

$$x = t_i \begin{pmatrix} f(s_1, 1) & f(s_1, 2) & \dots & f(s_{N_s}, 0) \\ ch_{1,1} & ch_{1,2} & \dots & ch_{1,N_f} \\ \dots & \dots & \dots & \dots \\ ch_{i,1} & ch_{i,2} & \dots & ch_{i,N_f} \\ \dots & \dots & \dots & \dots \\ ch_{N_t,1} & ch_{N_t,2} & \dots & ch_{N_t,N_f} \end{pmatrix}, ch_{i,j} \in \{-1, 0, 1, 2, 3\}$$

$x(i, j)$ indicates the channel allocation of user t_i for flow f_j . Values 0, 1, 2, 3 represent four possible user allocations: $UE_{noch,fach,dch,hs}(f_{s,j})$. -1 means the user does not belong to the multicast group. Hamming distance is used to measure the distance of two feasible solutions, which will be used in the calculation of indicators for fitness landscape analysis.

The power consumption is implicitly determined once user and channel code allocation are determined. As simulated in [1], P_{FACH} can be got through

a look up table based on the user distribution in $\text{UE}_{\text{fach}}(f_{s,j})$. The DCH transmission power for n users [9] is calculated by : $P_{\text{DCH}} = \frac{P_p + \sum_{i=1}^n Lp_i \cdot \frac{P_n + I_i}{(E_b/N_o)R_{b,i} + p}}{1 - \sum_{i=1}^n \frac{p}{(E_b/N_o)R_{b,i}} + p}$, where P_p is common control channel power, P_n is the background noise, Lp_i is user path loss, W is the bandwidth in UMTS, $R_{b,i}$ is the transmit rate of i th user, p is the orthogonality factor, I_i is the intercell interference observed by i th user. The transmit power to guarantee the required HS-DSCH throughput [8] is: $P_{\text{HS-DSCH}} \geq \text{SINR} \times [p - G^{-1}] \frac{P_{\text{own}}}{16}$, in which P_{own} is the own cell interference experienced by user, G is the geometry factor defined by $G = \frac{P_{\text{own}}}{P_{\text{other}} + P_{\text{noise}}}$, related with the user position.

A two-dimensional fitness value $F(x) = \{Th(x), Po(x)\}$ is defined to measure the solution quality. $Po(x)$ is the MBMS transmission power: $Po(x) = \sum_{s_i \in S(c)} \sum_{f_j \in F(s_i)} \sum_{ch_l} P_{f_j, ch_l}$. $Th(x)$ is the throughput loss in cell: $Th(x) = \sum_{s_i \in S(c)} \sum_{f_j \in F(s_i)} \sum_{t_u \in \text{Dist}(f_{s_i, j})} \max[-\Delta_{j,u}, 0]$. $\Delta_{j,u}$ is the difference between allocated channel bit rate (determined by its channel code(s) [3]) and flow bandwidth. The optimization target is to first satisfy the throughput requirement within the maximum power budget, then to minimize the transmission power. Therefore, the comparison of a new solution x' and current solution x is conducted in lexicographic order: x' is evaluated as a better solution when $Th(x') = Th(x)$ and $Po(x') \leq Po(x)$, or $Th(x') < Th(x)$.

We propose two operators to move solution into a neighbor solution.

- **Hybrid-moves operator** δ_H randomly selects user t_k from a non-empty $\text{UE}_{\text{ch}_o}(f_{s,j})$, then t_k or a block of users including t_k is moved from $\text{UE}_{\text{ch}_o}(f_{s,j})$ to $\text{UE}_{\text{ch}_i}(f_{s,j})$. The moved users depends on the chosen set. For example, if ch_i is FACH, the coverage is enlarged to t_k , all the users nearer than t_k can now hear from FACH, thus, no matter what user sets they are currently allocated at, they should stay or be inserted in $\text{UE}_{\text{fach}}(f_{s,j})$.
- **Insert operator** δ_I moves only one user for each operator application. Once it chooses FACH as ch_i or ch_o , then t_k is determined: the nearest user in $\text{UE}_{\text{hs}}(f_{s,j}) \cup \text{UE}_{\text{dch}}(f_{s,j}) \cup \text{UE}_{\text{noch}}(f_{s,j})$, or the farthest user within $\text{UE}_{\text{fach}}(f_{s,j})$.

3 Fitness Landscape of F2R2M

In [Table 1](#), five test scenarios with different service configurations and user numbers are designed for simulations. For example, in the first scenario $2s-50-sn$, two services s_1 and s_2 are transmitted to 30 and 20 users respectively. s_1 consists of three flows: f_1, f_2 at 32 kbps, and f_3 at 64 kbps. s_2 consists of only

one flow f_0 at 128 kbps. The users are assumed to be uniformly distributed around the base station.

Table 1
Simulation scenarios

Scenarios	Service number, Flow bandwidth	User number per service
scen 1: 2s-50u-sn	2, $s_1(f_1, f_2 : 32\text{kbps}, f_3 : 64\text{kbps}); s_2(f_0 : 128\text{kbps})$	30,20
scen 2: 2s-50u-ss	2, $s_1(f_1, f_2 : 32\text{kbps}, f_3 : 64\text{kbps}); s_2(f_1, f_2 : 64\text{kbps})$	30,20
scen 3: 3s-80u-snn	3, $s_1(f_1, f_2 : 32\text{kbps}, f_3 : 64\text{kbps}); s_2(f_0 : 128\text{kbps}); s_3(f_0 : 64\text{kbps})$	30,20,30
scen 4: 3s-80u-ssn	3, $s_1(f_1, f_2 : 32\text{kbps}, f_3 : 64\text{kbps}); s_2(f_1, f_2 : 64\text{kbps}); s_3(f_0 : 64\text{kbps})$	30,20,30
scen 5: 3s-100u-ssn	3, $s_1(f_1, f_2 : 32\text{kbps}, f_3 : 64\text{kbps}); s_2(f_1, f_2 : 64\text{kbps}); s_3(f_0 : 64\text{kbps})$	30,40,30

In our model, two fitness landscapes L_{hy} and L_{in} are defined by δ_H and δ_I . We study their characteristics in three aspects: 1) the distribution of feasible solutions within search space; 2) the distribution of fitness space; and 3) the links between distance and fitnesses of solutions. Two populations of solution S_{ini} and S_{lo} are generated. S_{ini} is composed of 600 solutions randomly chosen from the initial solution space. S_{lo} is the population of local optima found by applying hill climbing (HC) [7] to S_{ini} . Starting from a random solution in S_{ini} , HC evaluates all its feasible neighbors and replaces it by the neighbor that has the best fitness. HC stops at S_{lo} when no neighbor is better than current solution.

3.1 Analysis of Search Space

To study the distribution of feasible solutions in search space, two types of distance are calculated: d_{ini} and d_{lo} are the distances among any two solutions in S_{ini} and S_{lo} . Table 2 presents the minimum, the maximum and the quartiles

Table 2
Analysis of Search Space: distance between solutions of landscapes

	$L_{in} : d_{ini}$ in $S_{ini,in}$			$L_{in} : d_{lo}$ in $S_{lo,in}$			$L_{hy} : d_{ini}$ in $S_{ini,hy}$			$L_{hy} : d_{lo}$ in $S_{lo,hy}$		
	Min	Quartiles	Max	Min	Quartiles	Max	Min	Quartiles	Max	Min	Quartiles	Max
scen 1	15	62 _{54,72}	109	17	62 _{54,71}	108	19	62 _{53,71}	108	0	34 _{23,45}	109
scen 2	20	64 _{55,74}	114	20	65 _{55,75}	117	23	64 _{54,73}	121	0	33 _{22,46}	127
scen 3	20	64 _{55,73}	116	20	64 _{55,74}	116	21	65 _{54,73}	115	0	43 _{30,57}	138
scen 4	26	66 _{56,76}	148	20	66 _{56,76}	148	20	64 _{55,73}	116	0	35 _{24,49}	156
scen 5	21	64 _{55,73}	130	19	65 _{55,76}	180	21	64 _{55,73}	129	0	38 _{23,56}	194

(median, Q1 and Q3) of these distances. Firstly, for both operators, the distance statistics in the initial solution space (d_{ini} in $S_{ini,in}$ and $S_{ini,hy}$) are homogeneous, that is simply because they are randomly selected in the initial

solution space. Secondly, when looking at the the distance statistics in the local optima space (d_{lo} in $S_{lo,in}$ and $S_{lo,hy}$), δ_H generates much converged space.

3.2 Analysis of Fitness Space

[Figure 1](#) illustrates the fitness value distribution of S_{ini} and S_{lo} for the fifth scenario $3s\text{-}100u\text{-}ssn$. The fitness values of S_{ini} are well diversified for both operators. Comparing the first and third subfigures, the quality of $S_{lo,hy}$ is

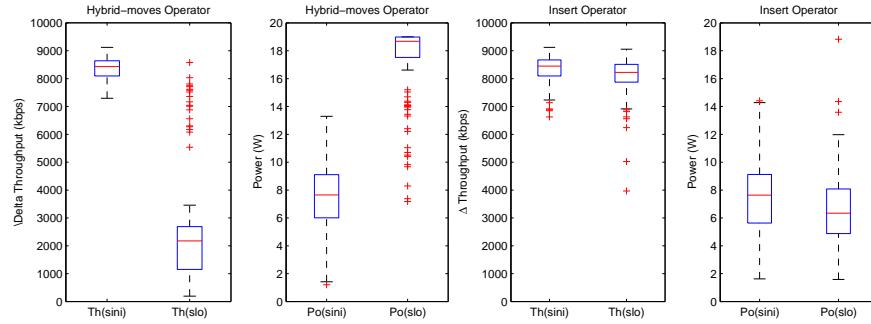


Fig. 1. Analysis of Fitness Space: Comparison of two fitness space (3s-100u-ssn)

better than that of $S_{lo,in}$ ($Th_{hy}(lo) < Th_{in}(lo)$). The same situation for the other scenarios is confirmed in [Table 3](#), in which we can see δ_H finds solutions with smaller $Th(x)$, hence better solutions than δ_I . This can be explained by the fact that, unlike δ_I , δ_H moves a block of elements which explores larger search space, so that it can easily escape from the local optimal.

Table 3
Fitness space analysis: fitness values of populations of local optima

Scenarios	Fitness	$L_{hy}: S_{lo,hy}$				$L_{in}: S_{lo,in}$			
		Min	Med _{Q1,Q3}	Max	Mean	Min	Med _{Q1,Q3}	Max	Mean
2s-50u-sn	Th(x) %	0	0 _{0,0}	58	1.7	4.5	57.5 _{52.25,62}	69	56.21
	Po(x) w	7.08	14.89 _{12.95,18.18}	19.0	15.28	1.5	6.53 _{5.03,8.33}	18.54	6.44
2s-50u-ss	Th(x) %	0	0 _{0,0}	62.5	0.94	0	58.75 _{54.62}	69.5	10.74
	Po(x) w	5.81	14.63 _{14.36,15.02}	16.58	14.57	1.69	6.24 _{4.69,7.93}	16.3	6.44
3s-80u-snn	Th(x) %	0	4 _{0,15}	91.5	9.43	37	87.5 _{82.92}	99	86.61
	Po(x) w	5.543	17.29 _{15.92,18.79}	18.99	17.2	2.10	6.56 _{4.96,8.21}	18.96	6.74
3s-80u-ssn	Th(x) %	0	1 _{0,2}	93.5	3.8	20.5	88 _{83.5,91.5}	99	86.34
	Po(x) w	6.38	17.57 _{17.22,17.63}	19.0	17.2	2.1	6.56 _{4.96,8.21}	18.96	6.75

3.3 Analysis of Links between Distance and Fitness

In F2R2M, step length is the number of implemented operator applications by using hill climbing method. The average step length in L_{hy} for five scenarios are: 26, 27, 43, 68, 100. While in L_{in} , they are: 10, 10, 9, 11, 13. In general, δ_I moves shorter distance than δ_H , which makes the search procedure walks nearby the initial solution. This explains that the search space of $S_{lo,in}$ does not concentrate the $S_{ini,in}$ (Table 2). Therefore, L_{in} is “shallower” than L_{hy} .

3.4 Results Comparison with Conventional Algorithms

The solutions of competing approaches and the best found solutions based on F2R2M are shown in Table 4. Solutions are represented in two-dimensional: $Th(x)$ in percentage and $Po(x)$ in watts. The maximum power budget is set as 19w hence solutions with $Th(x) < 19$ are feasible and emphasized in boldface. Both operators in F2R2M can always find good enough solutions: less throughput loss with less power consumption than existing approaches. The mean fitness values of found solution with F2R2M-in for scenarios 2 to 5

Table 4
Results comparison with existing approaches

Scenarios	MPC $\{Th(x), Po(x)\}$	DTM $\{Th(x), Po(x)\}$	S-FACH $\{Th(x), Po(x)\}$	F2R2M-in $\{Th(x), Po(x)\}$	F2R2M-hy $\{Th(x), Po(x)\}$
2s-50u-sn	0%, 27.2	0%, 30.5	65% 10.2	4.5%, 18.5	0% 10.2
2s-50u-ss	0%, 27.2	0%, 30.5	47% 15.4	0%, 15.58	0% 13.1
3s-80u-snn	0%, 32.5	0%, 37.7	25.4% 27.0	25.4%, 16.9	0% 15.0
3s-80u-ssn	0%, 32.5	0%, 37.7	36.2% 22.6	15.4%, 16.5	0% 14.4
3s-100u-ssn	0%, 37.73	0%, 37.69	51.18%, 15.9	15.9%, 18.12	1.76%, 17.5

are from (56.2%, 6.4 w) to (74.4%, 6.65 w) with standard deviation from (8.12, 2.45) to (7.13, 2.55). While for F2R2M-hy, the mean fitness values are from (1.7%, 15.3 w) to (7.29%, 18.12 w) with standard deviation from (8.2, 2.56) to (11.93, 1.4). δ_H performs better than δ_I , which proves the discussion in section 3 that δ_H has capacity of “jump” from bad solutions, while δ_I can only stay in basins. Besides, the average consuming time of one HC trial is calculated, δ_I and δ_H needs 0.06s and 0.16s for 2s-50u-sn, 0.77s and 0.31s for 3s-100u-ssn. Both operators can find good enough solution with acceptable time. δ_H can move more steps hence farther than δ_I , therefore the hybrid-moves operator costs double time than insert operator.

4 Conclusion and Perspective

In order to solve the RRM problem in MBMS system, we propose a novel model named Flexible Radio Resource Management Model (F2R2M) using metaheuristic approach. In our model, two objectives are took into account to satisfy the QoS request and minimize the power consumption. Based on fitness landscape analysis, a study is investigated on the performance of the two proposed neighborhood functions. Simulation results show that our approaches can achieve significant gains comparing with the other existing approaches. In the future work, we are interested in applying the other local search algorithms such as Simulated Annealing and Tabu Search to improve the search efficiency.

References

- [1] 3GPP Standards: TR 25.803 S-CCPCH performance for MBMS, Sep 2005.
- [2] 3GPP Standards: TR 25.922 radio resource management strategies, Mar 2007.
- [3] 3GPP Standards: TS 25.213 spreading and modulation (FDD), December 2009.
- [4] A. Alexiou, C. Bouras, and E. Rekkas. A power control scheme for efficient radio bearer selection in MBMS. In *World of Wireless, Mobile and Multimedia Networks, 2007. WoWMoM 2007. IEEE International Symposium on*, pages 1–8, June.
- [5] C. Christophorou, A. Pitsillides, and T. Lundborg. Enhanced radio resource management algorithms for efficient MBMS service provision in UTRAN. *Computer Networks*, 55(3), 2011.
- [6] A. M. C. Correia, J. C. M. Silva, N. M. B. Souto, L. A. C. Silva, A. B. Boal, and A. B. Soares. Multi-resolution broadcast/multicast systems for MBMS. *Broadcasting, IEEE Transactions on*, 53(1):224–234, Mar. 2007.
- [7] F. W. Glover and G. A. Kochenberger, editors. *Handbook of Metaheuristics*. Springer, 1 edition, Jan. 2003.
- [8] A. T. Harri Holma. *HSDPA/HSUPA for UMTS: High Speed Radio Access for Mobile Communications*. John Wiley & Sons, 2006.
- [9] J. P. Romero, O. Sallent, R. Agusti, and M. A. Diaz-Guerra. *Radio Resource Management Strategies in UMTS*. Wiley, 1 edition, Aug. 2005.
- [10] L. Technologies. 3GPP TSG RAN WG128 R1-02-1240 power usage for mixed FACH and DCH for MBMS, Oct 2002.

Formulations for the Minimum 2-Connected Dominating Set Problem

Vinicio Leal do Forte ^{a,1}, Abilio Lucena^a and
Nelson Maculan^a

^a *Programa de Engenharia de Sistemas e Computação - COPPE, Universidade Federal do Rio de Janeiro, Rio de Janeiro, Brazil*

Abstract

Three formulations for the Minimum 2-Connected Dominating Set Problem, valid inequalities, a primal heuristic and Branch-and-Cut algorithms are introduced in this paper. As shown here, the preliminary computational results so far obtained indicate that these algorithms are quite promising.

Keywords: dominating sets, 2-connected graphs, branch-and-cut algorithms

Let $G = (V, E)$ be a connected undirected graph with a set of vertices V and a set of edges E . For a given $W \subset V$, let $\Gamma_W \subset V$ be the closed neighborhood of W , i.e., the subset of vertices of V formed by W and those vertices that share an edge with a vertex of W . For simplicity, we write Γ_i instead of $\Gamma_{\{i\}}$ when W contains a single vertex i . Additionally, we define $\Gamma_i^- = \Gamma_i \setminus \{i\}$. If $\Gamma_W = V$ holds, W is called a dominating set of G . Furthermore, denoting by $E(W) = \{\{i, j\} \in E : i, j \in W\}$ the subset of edges of G with both end vertices in W , let $G_W = (W, E(W))$ be the subgraph W induces in G . A dominating set W is 2-edge connected (resp. 2-node connected) for G if G_W

¹ The authors were partially supported with CNPq grants 310561/2009-4, 300694/2009-1 and 475245/2012-1, and FAPERJ grant E26-110.552/2010.

is 2-edge-connected (resp. 2-node-connected), i.e., if for every pair of distinct vertices $u, v \in W$ there exists two edge (resp. node) disjoint paths in G_W connecting them. Initially, leaving aside the specific type of two connectivity involved, the Minimum 2-Connected Dominating Set Problem (M2CDSP) is to find a 2-connected dominating set W of minimum cardinality. To contain a 2-connected dominating set it suffices that G itself be 2-connected.

Applications involving minimum connected dominating sets can be found in the design of ad-hoc wireless sensor networks [1], in the design of defense strategies against the attack of worms in peer-to-peer networks [5], in the design of fiber optics networks where regenerators of information may be required at some network vertices [2], and as models to investigate protein-protein interactions [6]. Reliability is frequently a key issue in the design of some of these networks, particularly for telecommunications. When that applies, requiring that the dominating sets be 2-connected is a natural next step to take.

In this paper, we present three mixed integer programming formulations, valid inequalities, a primal heuristic, and Branch-and-Cut algorithms for the M2CDSP, in its 2-edge and 2-node connected variants. Two formulations are based on undirected cutset inequalities while the third is a directed reformulation with the addition of an artificial root vertex to the graph.

The paper is organized as follows. In Section 1, we introduce Mixed Integer Programming (MIP) formulations for M2CDSP, valid inequalities for them, and primal heuristics. Additionally, at the same section, we also describe Branch-and-Cut algorithms for these formulations. Finally, preliminary computational results and some conclusions are presented in Section 2.

1 Formulations, Primal Heuristic, and Algorithms

In this section we firstly introduce MIP formulations for the 2-edge-connected variant of M2CDSP. This is followed by some valid inequalities and an extension of these formulations to the 2-node-connected variant of the problem.

The first formulation uses variables $\{y_i \in \{0, 1\} : i \in V\}$ to select vertices for W . Additional variables, $\{x_e \in \mathbb{R}_+ : e \in E\}$, are used to enforce that subgraph G_W is 2-edge-connected. Denote by $\delta(S) = \{\{i, j\} \in E : i \in S, j \in V \setminus S\}$ the set of edges of G with only one end point in $S \subset V$ and consider a polyhedral region \mathcal{R}_1 defined as

$$\sum_{e \in \delta(S)} x_e \geq 2(y_i + y_j - 1), \quad S \subset V, \quad i \in S, \quad j \in V \setminus S \quad (1)$$

$$x_e \leq y_i \quad \text{and} \quad x_e \leq y_j \quad e = \{i, j\} \in E \quad (2)$$

$$\sum_{j \in \Gamma_i^-} y_j \geq y_i + 1, \quad i \in V \quad (3)$$

$$0 \leq x_e \leq 1, \quad e \in E \quad \text{and} \quad 0 \leq y_i \leq 1, \quad i \in V \quad (4)$$

A formulation for M2CDSP is then given by

$$\min \left\{ \sum_{i \in V} y_i : (\mathbf{x}, \mathbf{y}) \in \mathcal{R}_1 \cap (\mathbb{R}_+^{|E|}, \mathbb{B}^{|V|}) \right\} \quad (5)$$

and its corresponding Linear Programming (LP) relaxation is

$$\min \left\{ \sum_{i \in V} y_i : (\mathbf{x}, \mathbf{y}) \in \mathcal{R}_1 \right\}. \quad (6)$$

Inequalities (1) ensure that the solution is connected. Inequalities (3) impose that if $i \in V$ belongs to W , at least two neighbor vertices to it must also belong to that set. Otherwise, at least one neighbor vertex to i must belong to W . Finally, inequalities (1), (2), and (4) conform to the Theorem of Menger [3] to impose, for every pair of distinct vertices $i, j \in W$, that there must exist two edge disjoint paths in G_W connecting i and j .

A directed graph $D = (V, A)$ is obtained from G by taking, for every edge $e = \{i, j\} \subset E$, two arcs with the same end points and opposite directions. As before, let us denote by W a dominating set of D and by D_W the subgraph it induces in D . In accordance with a theorem of Nash-Williams [3], which applies to 2-arc-connected directed graphs, a reformulation of the problem must ensure, for every pair of distinct vertices $i, j \in W$, the existence of arc-disjoint paths in D_W going from i to j and vice versa. We have already investigated a formulation conforming to these conditions. However, it was shown to be equivalent to (5).

To obtain a directed graph reformulation with a LP relaxation that is not in a one-to-one equivalence to (6), let us consider an extended directed graph $D_r = (V_r, A_r)$ where $V_r = V \cup \{r\}$, r being an artificial root vertex, and $A_r = A \cup \{(r, i), (i, r) : i \in V\}$. Accordingly, the subgraph of D_r induced by feasible solutions must contain a dominating set $W \subset V$ of D , root vertex r , and a particular subset of A_r . In this subset, r must only have arcs pointing to and from a single vertex of W , say vertex i . Additionally, it must also

contain, for every other vertex $j \in W$, arc disjoint paths going from i to j and vice versa.

The formulation outlined above involves variables $\{y_i \in \{0, 1\} : i \in V\}$, as defined before. It also involves variables $\{x_{(i,j)} \in \mathbb{R}_+ : (i, j) \in A_r \text{ and } i \neq r\}$ and $\{x_{(r,j)} \in \{0, 1\} : (r, j) \in A_r\}$. To describe it, let $\delta^+(S) = \{(i, j) \in A : i \in S, j \in V \setminus S\}$ be the set of arcs leaving S and $\delta^-(S) = \{(i, j) \in A : i \in V \setminus S, j \in S\}$ be the set of arcs entering that set. Let us then consider a polyhedral region \mathcal{R}_2 defined as

$$\sum_{a \in \delta^-(S)} x_a \geq y_i, \quad \text{and} \quad \sum_{a \in \delta^+(S)} x_a \geq y_i, \quad S \subset V_r, \quad r \in S, \quad i \in V \setminus S \quad (7)$$

$$\sum_{i \in V} x_{(r,i)} = 1 \quad (8)$$

$$x_{(r,i)} = x_{(i,r)}, \quad i \in V \quad (9)$$

$$x_{(r,i)} \leq y_i, \quad i \in V \quad (10)$$

$$\sum_{j \in \Gamma_i^-} y_j \geq y_i + 1, \quad i \in V \quad (11)$$

$$x_{(i,j)} + x_{(j,i)} \leq y_i \quad (i, j) \in A_r, \quad i, j \neq r \quad (12)$$

$$0 \leq x_a \leq 1, \quad a \in A_r \quad \text{and} \quad 0 \leq y_i \leq 1, \quad i \in V. \quad (13)$$

A directed graph reformulation of M2CDSP is then given by

$$\min \left\{ \sum_{i \in V} y_i : (\mathbf{x}, \mathbf{y}) \in \mathcal{R}_2 \cap (\mathbb{R}_+^{|A_r|-|V|}, \mathbb{B}^{2|V|}) \right\} \quad (14)$$

and its corresponding LP relaxation is

$$\min \left\{ \sum_{i \in V} y_i : (\mathbf{x}, \mathbf{y}) \in \mathcal{R}_2 \right\}. \quad (15)$$

Inequalities (7) are the backward and forward cuts. They guarantee the existence of paths going from the root vertex r to a dominating set vertex and backwards from it.

As for the undirected formulation, inequalities (11) ensure the existence of a dominating set. The unique orientation of the arcs is guaranteed by (12).

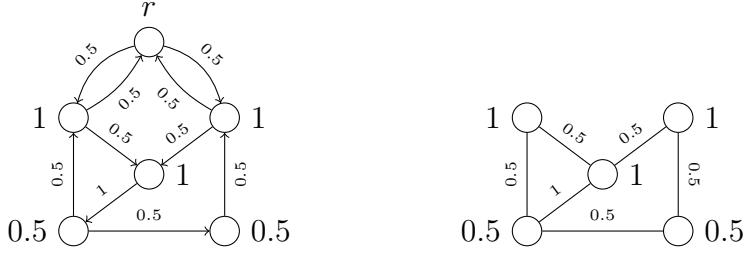


Fig. 1. From left to right, a feasible LP relaxation of directed graph reformulation (15) and its corresponding projection into the space of the (\mathbf{x}, \mathbf{y}) variables.

Inequalities (8), (9), and (10) enforce that only one arc leaves and enters r . Finally, one may use equations (9) to get rid of half of the variables they involve and also to get rid of equations (9) themselves.

Polyhedral regions \mathcal{R}_1 and \mathcal{R}_2 are not equivalent, as indicated in Figure 1. Specifically, the support graph for the LP relaxation of (15), after removing vertex r and transforming arcs into corresponding edges, does not necessarily map into a feasible solution to (6).

A formulation for the 2-node-connected variant of the problem, follows. It is obtained by further restricting \mathcal{R}_1 with the use of some additional inequalities. Accordingly, for $G_k = (V \setminus \{k\}, E \setminus \{\delta(\{k\})\})$, denote by $\delta_k(S)$ the subset of edges of G_k with only one end vertices in $S \subset V$ and define cutset inequalities

$$\sum_{e \in \delta_k(S)} x_e \geq y_i + y_j - 1 \quad S \subset V \setminus \{k\}, i \in S, j \in V \setminus S \cup \{k\}, k \in V. \quad (16)$$

These are adaptations of the node-cut inequalities introduced in [4]. They rely on the definition that a graph G is 2-node-connected if G remains connected after a single vertex, say vertex k , is removed from it. One then defines a polyhedral region \mathcal{R}_3 as the intersection of \mathcal{R}_1 with inequalities (16). Accordingly, a 2-node-connected formulation of M2CDSP is then given by

$$\min \left\{ \sum_{i \in V} y_i : (\mathbf{x}, \mathbf{y}) \in \mathcal{R}_3 \cap (\mathbb{R}_+^{|E|}, \mathbb{B}^{|V|}) \right\}. \quad (17)$$

Valid inequalities for (5) and (17) can be obtained as follows. From the definition of a dominating set, it must contain at least one vertex of Γ_i for every $i \in V$. Hence, given any pair of vertices $i, j \in V$, such that $\Gamma_i \cap \Gamma_j = \emptyset$ holds, any cut separating Γ_i from Γ_j must have at least 2 edges. The corresponding valid inequalities, that follow from valid inequalities suggested in [7] for the

Minimum Connected Dominating Set Problem, are then given by

$$\sum_{e \in \delta(S))} x_e \geq 2 \quad \Gamma_i \subset S, \quad \Gamma_j \subset V \setminus S, \quad \Gamma_i \cap \Gamma_j = \emptyset. \quad (18)$$

Let us now turn to heuristics for the problem. As proved in [3], a graph is 2-node-connected if and only if it can be constructed from a cycle by successively adding paths, starting and ending at different vertices of the structure at hand. Using this idea, we developed a primal heuristic for M2CDSP. It consists in progressively enlarging a subset W of V to become 2-node connected dominant for G . Accordingly, for every vertex $i \in V$, one builds sets W and L such that W contains all the vertices in the smallest cycle of G containing i and $L = \Gamma_W \setminus W$. As the heuristic progresses, ones selects $\{u, v\} = \text{argmax}\{|\delta(v) \setminus W \cup L| : v \in L, u \in W, \{u, v\} \in E\}$. Then, one adds to W those vertices that belong to the smallest cycle of G containing u and v , plus an additional vertex $l \in W \cap \delta(\{u\})$. As before, one sets $L = \Gamma_W \setminus W$. This procedure is repeated until $|W| + |L| = |V|$ holds, as a 2-node-connected dominating set W is then obtained for G . For the procedure above, in order to obtain a 2-edge connected G_W , no need exists of including in W the additional vertex l .

Preliminary computational experiments were carried out with the formulations introduced here. In particular, Branch and Cut algorithms were implemented for formulations (5), (14), and (17) and are respectively denoted by EUCUT, ERDCUT, and NUCUT. For all of them, dual bounds are obtained by enforcing cutsets inequalities as cutting planes. Separation of cutsets are carried out through the push-relabel algorithm [3]. The primal heuristic discussed above was used in all algorithms. For EUCUT and NUCUT, branching was conducted on the \mathbf{y} variables, while for ERDCUT it was carried out over \mathbf{y} and those \mathbf{x} variables involving r . We also implemented a Branch-and-Cut algorithm, denoted by EUCUT+, resulting from reinforcing formulation (5) with cutset inequalities (18).

2 Computational Results and Conclusions

Branch-and-Cut algorithms were implemented with solver CPLEX 12.4 being used to manage the search tree and to obtain LP relaxations at every enumeration tree node. All the other solver's features, such as automatic cut generation, primal heuristics, and pre-processing, were deactivated. All experiments were conducted on a Intel Xeon machine running at 2.67 GHz with 24 Gbits of RAM memory.

Test instances used are defined over randomly generated 2-connected graphs with edge densities 5%, 10%, and 50%, and $|V| \in \{50, 100, 150\}$. The results obtained are shown in Table 1. The first and the second column in that table respectively indicate, for each underlying graph $G = (V, E)$, $|V|$ and graph density. This is followed by four groups of columns, each containing four columns and corresponding to EUCUT, ERDCUT, EUCUT+, and NUCUT. Each of them contains, in the following order, lower and upper bounds at the root node of the enumeration tree, optimal solution values and CPU times to find proven optimal solutions. For those instances where optimality certificates were not obtained under the imposed CPU time limit of 7200 seconds, the percentage gaps of upper bounds over primal bounds are quoted instead of CPU times.

$ V $	den(%)	EUCUT				ERDCUT				EUCUT+				NUCUT			
		LB_0	UB_0	OPT	$t(s)$	LB_0	UB_0	OPT	$t(s)$	LB_0	UB_0	OPT	$t(s)$	LB_0	UB_0	OPT	$t(s)$
30	5	11.75	14	12	0.016	11.75	14	12	0.058	11.89	14	12	0.038	11.75	14	12	0.017
	10	7.5	10	8	0.019	7.5	10	8	0.029	8	10	8	0.047	7.5	10	8	0.020
	25	5	5	5	0.028	5	5	5	0.021	5	5	5	0.075	5	5	5	0.053
	50	3	3	3	0.015	3	3	3	0.027	3	3	3	0.016	3	3	3	0.016
50	5	14	18	15	0.091	14	18	15	0.096	14	18	15	1.39	14	18	15	0.139
	10	7.83	10	9	0.126	7.83	10	9	0.144	7.83	10	9	1.685	7.83	10	9	0.123
	25	3.71	6	5	0.348	3.71	6	5	3.298	3.71	6	5	33.48	3.71	6	5	0.364
	50	3	3	3	0.057	3	3	3	0.069	3	3	3	0.057	3	3	3	0.057
100	5	16.86	23	19	5.041	16.86	23	19	251.1	16.86	23	19	417.5	16.86	23	19	2.116
	10	9.14	15	13	1453.2	9.14	15	24.70%	7200	9.14	15	36.58%	7200	9.14	15	13	2293
	25	4.10	7	6	590.7	4.10	7	34%	7200	4.10	7	39.47%	7200	4.10	7	6	877.6
	50	1.98	4	4	31.76	1.98	4	34.42%	7200	1.98	4	4	32.20	1.98	4	4	32.02
150	5	17.81	26	14.30%	7200	17.81	26	19.00%	7200	17.81	26	22.00%	7200	17.81	26	9.70%	7200
	10	9.88	16	22.90%	7200	9.88	16	36.33%	7200	9.88	16	37.00%	7200	9.88	16	23.44%	7200
	25	3.90	8	38.30%	7200	3.90	8	49.47%	7200	3.90	8	46.34%	7200	3.90	8	38.30%	7200
	50	1.96	4	4	344.7	1.96	4	47.85%	7200	1.96	4	4	344.7	1.96	4	4	344.7

Table 1
Preliminary computational results

As one will notice from the results quoted in Table 1, all algorithms for the 2-edge connected variant of M2CDSP produced the same LP relaxation bounds, at the root node of the enumeration tree. Moreover, algorithms ERDCUT and EUCUT+ failed to solve instances with more than 100 vertices. Contrary to that, EUCUT managed to solve dense instances with 150 ver-

tices. CPU times for the 2-node connected variant of the M2CDSP, given by algorithm NUCUT, were essentially the same demanded by EUCUT. It should be pointed out that although NUCUT and EUCUT produced, for every test instance, optimal solutions with the same $|W|$, corresponding support graphs are different in either case. Finally, overall, our primal heuristic produced good quality bounds.

In terms of the future work, effort will be put into finding additional families of valid inequalities for our formulations. More efficient separation algorithms for the inequalities we already use will also be investigated. Finally, our primal heuristic is likely to improve through the use of different node selection strategies and local search neighborhoods.

References

- [1] B. Balasundaram and S. Butenko. Graph domination, coloring and cliques in telecommunications. In *Handbook of Optimization in Telecommunications*, pages 865–890. Springer, 2006.
- [2] S. Chen, I. Ljubić, and S. Raghavan. The regenerator location problem. *Networks*, 55(3):205–220, 2010.
- [3] Reinhard Diestel. *Graph Theory*. Springer, 4 edition, 2010.
- [4] M. Grötschel, C. L. Monma, and M. Stoer. Design of survivable networks. In M. O. Ball, C. L. Monma, and G. Nemhauser, editors, *Handbook in Operations Research and Management Science*, volume 7, pages 617–671. North Holland, 1995.
- [5] X. Liang and Z. Sencun. A Feasibility Study on Defending Against Ultra-Fast Topological Worms. In *Proceedings of The Seventh IEEE International Conference on Peer-to-Peer Computing (P2P’07)*, 2007.
- [6] T. Milenković, V. Memišević, A. Bonato, and N. Pržulj. Dominating Biological Networks. *PLoS ONE*, 6(8), 2011.
- [7] L. Simonetti, A. S. da Cunha, and A. Lucena. The minimum connected dominating set problem: Formulation, valid inequalities and a branch-and-cut algorithm. In Julia Pahl, Torsten Reiners, and Stefan Voß, editors, *INOC*, volume 6701 of *Lecture Notes in Computer Science*, pages 162–169. Springer, 2011.

Formulating and Solving the Minimum Dominating Cycle Problem

Abilio Lucena¹

*Departamento de Administração / PESC-COPPE
Universidade Federal do Rio de Janeiro
Rio de Janeiro, Brazil*

Alexandre Salles da Cunha

*Departamento de Ciência da Computação
Universidade Federal de Minas Gerais
Belo Horizonte, Brazil*

Luidi Simonetti

*Instituto de Computação
Universidade Federal Fluminense
Niteroi, Brazil*

Abstract

This paper introduces a formulation for the Minimum Dominating Cycle Problem. Additionally, a Branch and Cut algorithm, based on that formulation, is also investigated. So far, the algorithm contains no primal heuristics. However, it managed to solve to proven optimality, in acceptable CPU times, all test instances with up to 120 vertices.

Keywords: Minimum Dominating Cycle Problem, Formulations, Valid Inequalities, Branch and Cut Algorithm.

1 Introduction

Let $G = (V, E)$ be an undirected graph with a set of vertices V and a set of edges E and assume that $G_C = (V_C, E_C)$ is a subgraph of G . G_C is a simple cycle if it is connected and has exactly two edges incident to each of its vertices. Furthermore, C is dominating if every vertex of V is either in V_C or shares an edge with a vertex in V_C . Accordingly, the Minimum Dominating Cycle Problem (MDCP) is to find a dominating cycle of G with the least possible number of vertices.

More formally, for a given graph $G = (V, E)$, let $\Gamma(W) \subset V$ be the closed neighborhood of $W \subset V$. $\Gamma(W)$ is thus the subset of vertices of V formed by W and those vertices that share an edge with a vertex of W . If $\Gamma(W) = V$, W is dominating for G . Additionally, W is connected dominant if, on top of being dominant, it induces a connected subgraph $G_W = (W, E_W)$ of G . Finally, W is 2-edge connected dominating (resp. 2-node connected dominating) if it is dominant and G_W remains connected after removing any of its edges (resp. vertices). It thus follows that a dominating cycle of G is simultaneously 2-edge and 2-node connected dominant.

Applications involving minimum connected dominating sets can be found in the design of ad-hoc wireless sensor networks [1], in the design of defense strategies against the attack of worms in peer-to-peer networks [2], in the design of fiber optics networks where regenerators of information may be required at some network vertices [3], and as models to investigate protein-protein interactions [4]. When greater reliability is required, dominating cycles are particularly attractive structures for, among others, ad-hoc wireless sensor networks.

MDCP has been shown to be *NP*-hard in [5]. For some special classes of graphs, polynomial time algorithms have been suggested, among others, in [5,6,7]. However, for general graphs, no exact solution algorithm appears to exist in the literature.

In this paper we introduce a formulation for MDCP. As one would expect, it involves a formulation for the Simple Cycle Problem (SCP) [8,9,10] plus inequalities that impose the dominance condition. The SCP portion of our formulation distinguishes itself from existing SCP formulations through its connectivity enforcing approach. More specifically, we explicitly use General-

¹ Abilio Lucena was partially funded by CNPq grant 310561/2009-4 and FAPERJ grant E26-110.552/2010. Alexandre Salles da Cunha was partially funded by CNPq grants 302276/2009-2 and 477863/2010-8, and FAPEMIG PRONEX APQ-01201-09. Luidi Simonetti was partially funded by grants from FAPERJ and CNPq grant 483.243/2010-8.

ized Subtour Elimination Constraints (GSECs) for that purpose. Doing that, one should notice, is not straightforward since no pre-defined vertex exists for an optimal cycle. Furthermore, the number of vertices appearing in an optimal SCP solution is not known in advance. Consequently, it is unclear what an illegal cycle might be.

We also present in this paper some very preliminary computational results for a MDCP Branch and Cut algorithm. Although that algorithm does not yet contain a primal heuristic, it is capable of solving, to proven optimality, in no more than 1500 CPU seconds, instances of the problem with up to 120 vertices.

The paper contains two additional sections. Our MDCP formulation is presented in Section 2 while preliminary computational results for it are presented in Section 3. Still in Section 3, the future work we plan to do is discussed.

2 A Formulation for MDCP

In this section, we briefly discuss the SCP formulation in [10], prior to introducing our own new formulation for that problem. Dominance implying inequalities, to be appended to the latter formulation, then follows, thus giving rise to a MDCP formulation.

For every $S \subseteq V$ denote by $\delta(S) \subseteq E$ (resp. $E(S) \subseteq E$) the set of edges with exactly one end vertex in S (resp. the two end vertices in S) and, for simplicity, write $\delta(i)$ instead of $\delta(\{i\})$, for every $i \in V$.

The SCP formulation in [10] associates binary 0-1 variables y_i and x_e respectively with every vertex $i \in V$ and every edge $e \in E$. To impose solution connectivity, it relies on the use of cutset inequalities

$$(1) \quad \sum_{e \in \delta(S)} x_e \geq 2(y_i + y_j - 1), \quad S \subset V, \quad i \in S, \quad j \in S \subset V.$$

The formulation is not explicitly described here due to space limitations.

Our formulation of SCP [11] is described next. It is based on the idea of decomposing a simple cycle C in terms of one of its edges (any will do) and the simple path defined by remaining edges. Accordingly, we associate two different sets of variables to the edges of G . Namely, $\{x_e \in \mathbb{B} : e \in E\}$ for simple path edges and $\{z_e \in \mathbb{R}_+ : e \in E\}$ for the missing cycle edge. As before, we also associate variables $\{y_i \in \mathbb{B} : i \in V\}$ to the vertices of G .

Let \mathcal{R}_1 be the polyhedral region defined as

$$(2) \quad \sum_{e \in E(S)} x_e \leq \sum_{i \in S \setminus \{j\}} y_i, \quad \forall j \in S, \quad S \subset V$$

(3)

$$\sum_{e \in E} x_e \geq 2$$

(4)

$$\sum_{e \in E} z_e = 1$$

(5)

$$\sum_{e \in \delta(i)} (x_e + z_e) = 2y_i, \quad \forall i \in V$$

(6)

$$x_e + z_e \leq y_k, \quad \forall e = \{i, j\} \in E, \quad k = i \vee k = j$$

(7)

$$x_e, z_e \geq 0, \quad \forall e \in E$$

(8)

$$0 \leq y_i \leq 1, \quad \forall i \in V.$$

Assuming that edge costs $\{c_e \in \mathbb{R} : e \in E\}$ are associated with the edges of G , a formulation for SCP [11] is then given by

$$(9) \quad \min \left\{ \sum_{e \in E} c_e (x_e + z_e) : (\mathbf{x}, \mathbf{y}, \mathbf{z}) \in \mathcal{R}_1 \cap (\mathbb{B}^{|E|}, \mathbb{B}^{|V|}, \mathbb{R}_+^{|E|}) \right\}.$$

In order to show that (9) is valid, let $(\bar{\mathbf{x}}, \bar{\mathbf{y}}, \bar{\mathbf{z}})$ be a feasible solution for it and $G_{\mathbf{xyz}} = (V_{\mathbf{xyz}}, E_{\mathbf{xyz}})$ be the subgraph of G induced by the vertices and edges corresponding to nonzero entries in $(\bar{\mathbf{x}}, \bar{\mathbf{y}}, \bar{\mathbf{z}})$. Accordingly, let $G_{\mathbf{xy}} = (V_{\mathbf{xy}}, E_{\mathbf{xy}})$ be the subgraph of G induced by $(\bar{\mathbf{x}}, \bar{\mathbf{y}})$. Firstly notice that summing (5) over all $i \in V$, after some algebraic manipulation, one arrives at $\sum_{e \in E} x_e + \sum_{e \in E} z_e = \sum_{i \in V} y_i$. Accordingly, $\sum_{e \in E} x_e = \sum_{i \in V} y_i - 1$ results from (4) and $|E_{\mathbf{xy}}| = |V_{\mathbf{xy}}| - 1$ thus hold for $G_{\mathbf{xy}}$. Furthermore, that graph must be cycle free, as imposed by GSECs (2). As such, the combination of that new equation with (2), (7) and (8) ensures $G_{\mathbf{xy}}$ to be a tree of G (observe that $x_e \leq 1$, for all $e \in E$, is implied by (2)). Turning now to $G_{\mathbf{xyz}}$, notice that $V_{\mathbf{xy}} = V_{\mathbf{xyz}}$ holds and consequently $G_{\mathbf{xyz}}$ must be connected, since $G_{\mathbf{xy}}$ is connected. Additionally, as imposed by (4), a single edge of G is implied by $\bar{\mathbf{z}}$ and, as enforced by (6), that edge must be distinct from those found in $G_{\mathbf{xy}}$. Finally, every vertex in $G_{\mathbf{xyz}}$ must have exactly two edges incident to it, as imposed by (5), and it thus follows that $G_{\mathbf{xyz}}$ must be a simple cycle of G . Accordingly, $G_{\mathbf{xy}}$ must be a simple path for that graph. It could be shown [11] that no dominance relation exists between the Linear Programming (LP) relaxation of the formulation above and that of [10].

As pointed out before, \mathcal{R}_1 may be reinforced through the use of cutset inequalities (1). Indeed, the equivalent packing description of these inequalities has been shown in [8] to induce facets for the Cycle Polytope (CP). Still in that reference, a lifting procedure was introduced for obtaining facets of CP from facets of the Traveling Salesman Problem polytope.

In this paper we investigate the strengthening of \mathcal{R}_1 through *generalized*

2-matching inequalities(G2Ms). These are defined through a *handle* $H \subset V$ together with an accompanying *teeth* $T \subseteq \delta(H)$, where $|T| \geq 3$ and is odd, as

$$(10) \quad \sum_{e \in E(H)} (x_e + z_e) + \sum_{e \in E(T)} (x_e + z_e) \leq \sum_{i \in H} y_i + \lfloor \frac{|T|}{2} \rfloor.$$

Let us now turn to our formulation for MDCP. Accordingly, denote $(\Gamma(i) \setminus \{i\})$ by $\Gamma^-(i)$ and let \mathcal{R}_2 be the polyhedral region defined by the intersection of \mathcal{R}_1 with the dominance imposing inequalities

$$(11) \quad -y_i + \sum_{j \in \Gamma^-(i)} y_j \geq 1, \quad i \in V.$$

A formulation for MDCP is then given by

$$(12) \quad \min \left\{ \sum_{e \in E} (x_e + z_e) : (\mathbf{x}, \mathbf{y}, \mathbf{z}) \in \mathcal{R}_2 \cap (\mathbb{B}^{|E|}, \mathbb{B}^{|V|}, \mathbb{R}_+^{|E|}) \right\}$$

and its corresponding LP relaxation is

$$(13) \quad \min \left\{ \sum_{e \in E} (x_e + z_e) : (\mathbf{x}, \mathbf{y}, \mathbf{z}) \in \mathcal{R}_2 \right\}.$$

The validity of inequalities (11) can be checked as follows. If a vertex $i \in V$ is not in W , then $y_i = 0$ holds and the inequality corresponding to i imposes that a neighbor vertex to i must be in W . Alternatively, if i is in W , $y_i = 1$ holds and at least two neighbor vertices to i must also be in W , as one would expect from a cycle containing that vertex. Accordingly, the negative coefficient of y_i then ensures satisfaction of the condition.

3 Computational results

Very preliminary computational tests were carried out with a Branch and Cut algorithm based on formulation (12). The algorithm separates GSECs (2) and G2Ms (10) and contains no heuristic for finding feasible solutions to MDCP. Although our separation algorithms were capable of identifying violated G2Ms and we used them as cutting planes, no overall improvement on LP relaxation bounds was attained out of their use. As for GSECs, experiments were carried out involving their exact and heuristic separation. Better Branch and Cut CPU times were obtained under heuristic separation. We therefore only present here results for that variant of the Branch and Cut algorithm where G2Ms and GSECs are heuristically separated. The heuristic separation of GSECs, in this case, being conducted as suggested in [12].

The XPRESS Mixed Integer Programming package was used to solve linear programs and manage the search trees. All other features of the solver, such

as pre-processing, primal heuristics and automatic cut generation were kept switched-off. Experiments were conducted on a XEON Quad Core based machine running at 2.0Ghz and 8 Gbytes of RAM memory. Test instances for our experiments are the randomly generated graphs used in the computational experiments carried out in [12], for the Minimum Connected Dominating Set Problem (MCDSP).

A summary of the results obtained is presented in Table 1. Six columns are contained in that table. The first one identifies the test instances. Accordingly, instance "v_30d_10", for example, corresponds to an underlying graph with 30 vertices and 10% density. Underlying graphs in our test bed therefore involve from 30 up to 200 vertices and densities ranging from 5% up to 70%. The second column entries correspond to LP relaxation bounds at the root node of the enumeration tree. Entries in the third and fourth columns respectively correspond to the best lower and upper bounds available at the time optimality is proven or else at the time the CPU limit of 3600 seconds was reached. Given that lower bounds may be rounded up, whenever a difference of less than one unit exists between lower and upper bounds, optimality of the upper bound is proven. Finally, entries in the fifth and the sixth columns respectively correspond to CPU time and number of enumeration tree nodes. An entry "tl" in the fifth column indicates that the CPU time limit was reached prior to obtaining an optimality certificate.

From the results quoted in Table 1, for lower density instances, lower bounds at the root node of the enumeration tree tend to be far from corresponding optimal solution values. The reverse occurs for higher density instances. Additionally, one may observe that, in acceptable CPU times, all test instances with 120 vertices or less could be solved to proven optimality.

We believe that computational results should improve significantly when a MDCP heuristic is introduced in our Branch and Cut algorithm. Likewise, that algorithm should also benefit from the use of additional valid inequalities for SCP, cutset inequalitites (1), in particular. The same is likely to occur out of our use of valid inequalities for MCDSP, such as those proposed in [12]. Among these we single out the domination enforcing inequalities $\{\sum_{j \in \Gamma(i)} y_j - \sum_{e \in E(\Gamma(i))} x_e \geq 1, i \in V\}$, that are valid since they apply to the path, i.e., spanning tree, portion of our MDCP formulation. Finally, notice that one may either append to \mathcal{R}_2 inequalities imposing that a feasible solution contains at least 3 vertices or else interrupt the tree search when a feasible solution involving exactly 3 vertices is obtained. Neither of these alternatives is currently being enforced. Our plans for the near future are to implement all the suggestions made above.

Instance	LP	Branch-and-cut results			
		BLB	BUB	$t(s)$	nodes
v30_d10	12.25	18.00	18	0.01	9
v30_d20	6.82	7.01	8	0.03	23
v30_d30	3.31	4.01	5	0.21	119
v30_d50	2.01	2.01	3	0.02	5
v30_d70	1.42	2.01	3	0.08	27
v50_d10	10.42	13.01	14	0.41	171
v50_d20	5.22	6.01	7	0.51	149
v50_d30	3.45	4.01	5	1.1	175
v50_d50	2.04	3.00	3	0.59	60
v50_d70	1.45	2.01	3	0.69	51
v70_d05	23.551	34.010	35	1.62	585
v70_d10	11.09	13.01	14	3.56	565
v70_d20	5.29	7.01	8	10.21	1033
v70_d30	3.37	4.01	5	5.48	371
v70_d50	2.07	3.00	3	2.11	62
v70_d70	1.45	2.01	3	2.51	69
v100_d05	21.04	28.00	28	858.75	16738
v100_d10	10.85	14.01	15	45.26	2425
v100_d20	5.34	7.01	8	19.89	613
v100_d30	3.43	5.01	6	223.11	4319
v100_d50	2.12	3.01	4	19.04	381
v100_d70	1.46	2.01	3	9.72	105
v120_d05	22.43	27.01	28	380.99	8727
v120_d10	10.51	13.01	14	392.67	7237
v120_d20	5.28	7.01	8	1409.42	9993
v120_d30	3.43	5.01	6	795.93	6723
v120_d50	2.02	3.01	4	25.16	171
v120_d70	1.44	2.01	3	17.41	131
v150_d05	21.63	26.02	29	tl	15701
v150_d10	10.76	13.07	16	tl	9936
v150_d20	5.19	7.08	10	tl	7896
v150_d30	3.47	4.68	8	tl	7163
v150_d50	1.99	3.01	4	54.31	197
v150_d70	1.45	2.01	3	98.23	343
v200_d05	22.40	24.85	31	tl	9015
v200_d10	10.55	12.13	20	tl	4084
v200_d20	5.02	5.92	12	tl	1765
v200_d30	3.36	4.30	8	tl	3357
v200_d50	2.01	3.01	4	2502.01	6501
v200_d70	1.44	2.01	3	313.33	413

Table 1

Linear Programming bounds and Branch-and-cut computational results.

References

- [1] B. Balasundaram and S. Butenko. Graph domination, coloring and cliques in telecommunications. In *Handbook of Optimization in Telecommunications*, pages 865–890. Springer, 2006.
- [2] X. Liang and Z. Sencun. A Feasibility Study on Defending Against Ultra-Fast Topological Worms. In *Proceedings of The Seventh IEEE International Conference on Peer-to-Peer Computing (P2P'07)*, 2007.
- [3] S. Chen, I. Ljubić, and S. Raghavan. The regenerator location problem. *Networks*, 55(3):205–220, 2010.
- [4] T. Milenković, V. Memišević, A. Bonato, and N. Pržulj. Dominating Biological Networks. *PLoS ONE*, 6(8), 2011.
- [5] A Proskurowski. Minimum dominating cycles in 2-trees. *International Journal of Computer & Information Sciences*, 8(5):405–417, 1979.
- [6] A. Proskurowski and M. Syslo. Minimum dominating cycles in outerplanar graphs. *International Journal of Computer & Information Sciences*, 10(2):127–139, 1981.
- [7] C.J. Colburn, J.M. Keil, and L.K. Stewart. Finding minimum dominating cycles in permutation graphs. *Operations Research Letters*, 4(1), year =.
- [8] J. J. Salazar. On the cycle polytope of an undirected graph. Working Paper, University of La-Laguna, 1994.
- [9] N. Maculan, G. Plateau, and A. Lisser. Integer models with a polynomial number of variables and constraints for some classical combinatorial optimization problems. *Pesquisa Operacional*, 23(1):161–168, 2003.
- [10] M. Fischetti, J.J. Salazar, and P. Toth. *The Traveling Salesman Problem and its Variations*, chapter The generalized traveling salesman and orienteering problems, pages 865–890. Kluwer Academic Publishers, New York, 2004.
- [11] A. Lucena, A. Salles da Cunha, and L. Simonetti. A new formulation and computational results for the simple cycle problem. Technical report, Universidade Federal do Rio de Janeiro, submitted to the Latin-American Graphs and Optimization Symposium, 2012.
- [12] L. Simonetti, A.S. da Cunha, and Lucena A. The Minimum Connected Dominating Set Problem: Formulation, Valid Inequalities and Branch-and-cut Algorithm. *Lecture Notes in Computer Science*, LNCS 6701:162–169, 2011. Network Optimization - Proceedings of the 5th International Network Optimization Conference, Hamburg.

Efficient algorithms for the 2-Way Multi Modal Shortest Path Problem

Marie-José Huguet^{1,2}, Dominik Kirchler^{3,4,5}
Pierre Parent⁵, Roberto Wolfier Calvo⁵

¹ CNRS, LAAS; Toulouse, France

² Université Toulouse; LAAS; France

³ LIX; Ecole Polytechnique

⁴ Mediamobile; Ivry-sur-Seine

⁵ LIPN; Université Paris 13

Abstract

We consider the 2-Way Multi Modal Shortest Path Problem (2WMMSPP). Its goal is to find two multi modal paths with total minimal cost, an outgoing path and a return path. The main difficulty lies in the fact that if a private car or bicycle is used during the outgoing path, it has to be picked up during the return path. The shortest return path is typically not equal to the shortest outgoing path as traffic conditions and timetables of public transportation vary throughout the day. In this paper we propose an efficient algorithm based on bi-directional search and provide experimental results on a realistic multi modal transportation network.

Keywords: 2-Way Shortest Path, time-dependency, multi modal transportation network, label constrained shortest path

¹ Email: kirchler@lix.polytechnique.fr, parent@lipn.fr

² Email: huguet@laas.fr, wolfier@lipn.fr

1 Introduction

Multi modal transportation networks include roads, public transportation, bicycle, etc. A *multi modal path* on such a network may consist of several modes of transportation. The goal of the 2-Way Multi Modal Shortest Path problem (2WMMSPP) is to find two multi modal paths with total minimal cost: an outgoing path from, e.g., home to work in the morning, and a return path from work to home in the evening. The main difficulty lies in the fact that if a private car or bicycle is used during the outgoing path, it has to be picked up during the return path. On a multi modal transportation network the shortest return path is typically not equal to the shortest outgoing path because of one-way roads and because traffic conditions and timetables of public transportation vary throughout the day. See Figure 1 for an example.

To the best of our knowledge, the 2WMMSPP has previously been studied only in [2]. The authors adopt a brute force algorithm by calculating all paths between the start and final location and a predetermined set of possible parking spaces.

Our Contribution. In this paper, we propose a new bi-directional multi modal shortest path algorithm for optimally solving the 2WMMSPP. We run experiments on a realistic multi modal transportation network. Our algorithm is much more efficient than the algorithm described in [2].

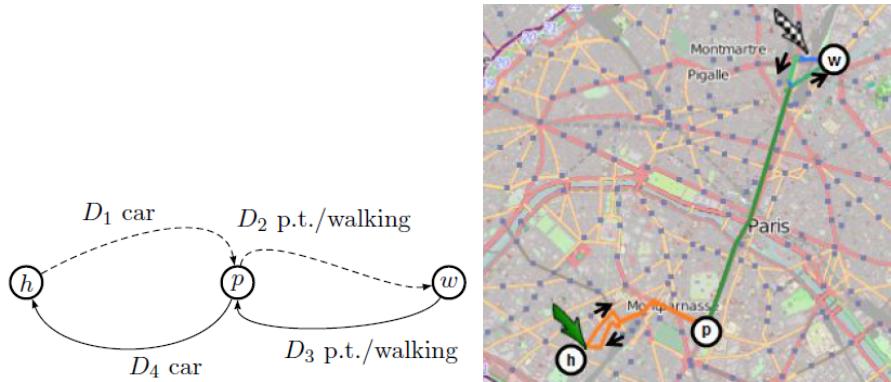


Fig. 1. Example of a 2-way path: h represents the home (source node), p the parking space, and w the work location (destination node). For the paths between h and p , only a private vehicle may be used (car or bike, orange line), and between p and w , only public transportation (p.t., green line) or walking (blue line). In this example arrival time at work in the morning is set to 9am and departure time from work in the evening is set to 5pm.

2 Algorithm to solve the 2WMMSP

Given a starting and arrival node as well as starting times and a set of modes of transportation, our algorithm determines the optimal parking location and the corresponding multi modal paths. First, we discuss how to calculate multi modal shortest paths and we introduce the graph we are using. Next, we discuss the basic version of our algorithm and some techniques which substantially improve its efficiency.

2.1 Multi Modal Shortest Path

Multi modal shortest paths are subject to particular constraints: passengers may want to exclude some modes of transportation (e.g., the bike when it is raining) or limit the number of changes. The *regular language constrained shortest path problem* (RegLCSP) introduced by [1] deals with these kind of issues, and it uses an appropriately labeled graph and a regular language to model constraints. A generalization of the well-known Dijkstra's algorithm to solve RegLCSP has been proposed (called hereafter D_{RegLC}).

Similar to [1] we use a *labeled* directed graph $G = (V, A, \Sigma)$ to model the multi modal transportation network. It consists of a set of nodes V , a set of labels Σ , and a set of labeled arcs $(i, j, l) \in A$ which are triplets in $V \times V \times \Sigma$. They represent an arc from node i to node j having label l . The labels are used to mark arcs as, e.g., foot paths, bike lanes, etc. Arc costs represent travel time. They are positive and may be *time-dependent* as we consider exact timetable information and changing traffic conditions: $c_{ijl}(\tau)$ gives the cost of an arc (i, j, l) when arriving at i at time τ .

2.2 Basic Algorithm

The *basic version* of our algorithm works as follows. We alternate the execution of four D_{RegLC} algorithms on G : algorithms D_1 and D_2 for the outgoing path, and D_3 and D_4 for the return path (see Figure 1). D_1 and D_4 calculate the shortest path from h to all other nodes by using only personal vehicle, and D_2 and D_3 from w to all other nodes by using only public transportation and foot. Algorithms D_1 and D_3 use forward search, and algorithms D_2 and D_4 use backward search. At each iteration, we choose the algorithm D_i which node x to be settled next has the lowest key among the nodes yet to be settled by the four D_{RegLC} algorithms. Node x is settled by D_i following the Dijkstra principle. Now we check if node x has been settled by all four algorithms. If this is the case, a new 2-way path has been found, for which x is the parking

space. To evaluate the cost of the 2-way path it suffices to add the costs of the 4 shortest paths to x as calculated by the 4 D_{RegLC} algorithms. If the total cost is better than the cost μ of the current best 2-way path, then μ is updated and x is memorized. The algorithm may stop as soon as the key δ of the next node to be settled is greater than or equal to μ .

Time dependency. In scenarios involving modes of transportation with time-dependent arc costs (car, public transportation), starting times of some algorithms have to be specified in order to be able to correctly evaluate the arc costs. Some starting times will be known, but others will not be. For example (with reference to Figure 1), if arrival and starting time at the destination w are given, then the starting time for algorithms D_1 and D_4 are not known. Therefore, for modes such that arc cost are not time-dependent (for instance foot or bike), we can use the basic algorithm without modifications. On the other hand, if personal vehicle arcs are time-dependent, the algorithm has to be adapted in the following way. Whenever D_1 and D_4 examine time-dependent arc costs, the minimum weight cost function $c_{ij}^{\min} = \min_{\tau \in H} c_{ij}(\tau)$ is used, where H is the time horizon. Then, when evaluating a newly found 2-way shortest path with parking node x , first its cost is calculated as described above. We call this cost *temporary cost*. If the temporary cost is lower than μ , the cost of the paths produced by D_1 and D_4 is *re-evaluate*. To do this, we run two D_{RegLC} algorithms starting in x . Starting times are given by the key of x in D_2 and D_3 . In this way, we obtain the real cost of the 2-way shortest path.

Complexity. Let n be the number of nodes in the graph, and m the number of arcs. Run time of the algorithm is $O(4m \times \log(n))$ in scenarios which do not require re-evaluation of shortest paths (using binary heap and for sparse graphs). Otherwise the complexity is $O(2n \times m \times \log(n))$.

2.3 Improved Stopping Condition

Proposition 2.1 *Let mc_i be the minimum total cost of a parking space settled by i of the 4 algorithms. $\lambda = \min_{i \in 1..3}(mc_i + (4 - i)\delta)$ (where δ is the cost of the next node to be settled) is a valid lower bound and the algorithm may stop when $\lambda \geq \mu$.*

Proof 1 *Any parking node which has not been settled yet by all four algorithms must have been settled either 0, 1, 2, or 3 times and its cost is at least 4δ , $mc_1 + 3 * \delta$, $mc_2 + 2 * \delta$, or $mc_3 + \delta$, respectively. Note that $4\delta \geq mc_1 + 3 * \delta$.*

Thus, as soon as $\lambda > \mu$, the optimal solution must have been found.

2.4 Further Improvements

We introduce three improvements for scenarios where re-evaluation of parking nodes is necessary.

- 1) Prior to starting the algorithm, we produce an upper-bound μ_0 (e.g., by choosing arbitrarily a parking node p and then calculating the four shortest paths between p , h , and w . This requires 4 times the execution of the D_{RegLC} algorithm.) The better the upper-bound the sooner the algorithm will stop and the less parking nodes will have to be re-evaluated.
- 2) Instead of calculating the minimum weight of an arc over the entire time horizon H , we may use the minimum weight of a more restricted time interval. In the example in Figure 1, arrival time t_0 and departure time t_1 at w are given. In this case we may use $c_{ij}^{\min} = \min_{\tau \in [t_0 - \mu_0 + \delta, t_0]} c_{ij}(\tau)$ for D_1 and $c_{ij}^{\min} = \min_{\tau \in [t_1, t_1 + \mu_0 - \delta]} c_{ij}(\tau)$ for D_4 .
- 3) For the re-evaluation, the SDALT algorithm [5] instead of D_{RegLC} may be used. SDALT applies A^* [4] and *landmarks* [3] to speed up D_{RegLC} . It uses an estimated lower bound of the distance to the destination node w to guide the search more directly toward the destination.

3 Experiments and Discussion

The algorithm was implemented in C++ and compiled with GCC 4.1. Experiments are run on an AMD Opteron, clocked at 2.2 Ghz. We use a realistic multi-modal transportation network based on road and public transportation data of the French region Ile-de-France which includes the city of Paris and its suburbs. It consists of four layers: bike, foot, car, and public transportation. The four layers are connected by transfer arcs. See [6] for more information about graph models of a multi-modal network and time-dependency. Data of the public transportation network have been provided by STIF³. They include geographical information, as well as timetable data on bus lines, tramway's, subways and regional trains. Data for the car layer is based on road and traffic information provided by Mediamobile⁴. Circa 15% of the road arcs have a time-dependent cost function to represent changing traffic

³ Syndicat des Transports IdF, www.stif.info, data for scientific use (01/12/2010)

⁴ www.v-trafic.fr, www.mediamobile.fr

conditions throughout the day. The foot as well as the bike layer are based on road data (foot paths, bike paths, etc.) extracted from geographical data freely available from OpenStreetMap⁵. The graph consists of circa 3.7m arcs and 1.1m nodes. There are 240 000 possible parking spaces for bikes and 20 000 for cars.

To evaluate run times of our algorithm, we build three realistic scenarios (see Table 1). For each scenario we specify the allowed modes of transportation for the paths between home and parking ($h \leftrightarrow p$) and between parking and work ($p \leftrightarrow w$). In all scenarios, arrival time at work for the outgoing path is 9am and departure time at work for the return path is 5pm.

In Table 1, we present the average CPU run time over 100 instances where h and w have been determined randomly. We compare the results of 5 variants of our algorithm. First, we run the basic version as described in Section 2.2. To this version we added incrementally the improved stopping condition (Section 2.3) and the three improvements discussed in Section 2.4. Note that a re-evaluation of parking nodes because of time-dependent arc costs is only necessary in scenario 3, thus the three improvements only apply to that scenario.

We are able to report faster run times than those reported by the authors of [2]. Run times of their brute force algorithm are quite high even when limiting the number of possible parking nodes. They report average run times of 1min when considering 20 parking nodes and of 30min for 80 parking nodes (on a Pentium M, 1.86 Ghz). Note also that we work on a considerably larger graph. The average run time of our algorithm for scenarios 1 and 2 is 1sec. The improved stopping condition has an important impact on run time for these scenarios. On the other hand, it has no impact on scenario 3 as the re-evaluation of parking nodes dominates run times. Improvements 1 and 2 succeed in decreasing the number of parking nodes which have to be re-evaluated and considerably accelerate the algorithm. Improvement 3 provides a further small speed-up.

4 Conclusions

We presented a new algorithm to solve the 2WMMSPP. We were able to report better run times than those reported in the literature. Future research directions include the investigation of stronger stopping conditions, of techniques to further decrease the number of parking nodes which have to be re-evaluated,

⁵ See www.openstreetmap.org

Scenario	Basic	Stop cond.	Imp 1	Imp 1+2	Imp 1+2+3
1) $h \leftrightarrow p$: bike, $p \leftrightarrow w$: foot	1.050	0.889	na	na	na
2) $h \leftrightarrow p$: bike, $p \leftrightarrow w$: foot, pt	2.143	1.050	na	na	na
3) $h \leftrightarrow p$: car, $p \leftrightarrow w$: foot, pt	199.5	194.6	60.4	4.7	3.7

pt: public transportation, na: not applicable, Imp: Improvement

Table 1
Average run times in seconds over 100 instances.

and of the use of parallelization.

References

- [1] Christoper L. Barrett, Riko Jacob, and Madhav Marathe. Formal-Language-Constrained Path Problems. *SIAM Journal on Computing*, 30(3):809–837, 2000.
- [2] Aurelie Bousquet, Sophie Constans, and El Faouzi Nour-Eddin. On the adaptation of a label-setting shortest path algorithm for one-way and two-way routing in multimodal urban transport networks. In *International Network Optimisation Conference*, pages 1–8, 2009.
- [3] Andrew V. Goldberg and C. Harrelson. Computing the shortest path: A* search meets graph theory. In *Proceedings of the Symposium on Discrete Algorithms (SODA)*, pages 156–165. SIAM, Philadelphia, 2005.
- [4] Peter E. Hart, Nils J. Nilsson, and Bertram Raphael. A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968.
- [5] Dominik Kirchler, Leo Liberti, and Roberto Wolfger Calvo. A label correcting algorithm for the shortest path problem on a multi-modal route network. In *Symposium on Experimental Algorithms (SEA)*, volume 7276 of *LNCS*, pages 236–247, 2012.
- [6] Evangelia Pyrga, Frank Schulz, Dorothea Wagner, and Christos D. Zaroliagis. Efficient models for timetable information in public transportation systems. *ACM Journal of Experimental Algorithms*, 12(2.4), June 2008.

On the Hardness of Equal Shortest Path Routing

Frédéric Giroire, Stéphane Pérennes, Issam Tahiri¹

*MASCOTTE
INRIA, I3S (CNRS/UNSA)
Sophia Antipolis, France*

Abstract

In telecommunication networks packets are carried from a source s to a destination t on a path that is determined by the underlying routing protocol. Most routing protocols belong to the class of shortest path routing protocols. In such protocols, the network operator assigns a length to each link. A packet going from s to t follows a shortest path according to these lengths. For better protection and efficiency, one wishes to use multiple (shortest) paths between two nodes. Therefore the routing protocol must determine how the traffic from s to t is distributed among the shortest paths. In the protocol called OSPF-ECMP (for Open Shortest Path First -Equal Cost Multiple Path) the traffic incoming at every node is uniformly balanced on all outgoing links that are on shortest paths. In that context, the operator task is to determine the “best” link lengths, toward a goal such as maximizing the network throughput for given link capacities.

In this work, we show that the problem of maximizing *even a single* commodity flow for the OSPF-ECMP protocol cannot be approximated within any constant factor ratio. Besides this main theorem, we derive some positive results which include polynomial-time approximations and an exponential-time exact algorithm. We also prove that despite their weakness, our approximation and exact algorithms are, in a sense, the best possible.

Keywords: OSPF-ECMP, max flow, NP-Hard, approximation.

¹ Corresponding author. Email: issam.tahiri@inria.fr

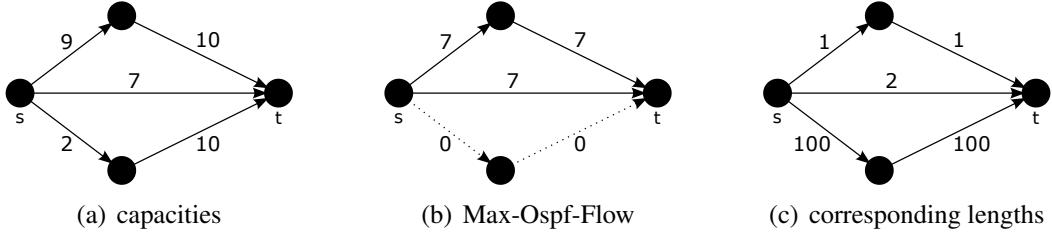


Fig. 1. Example of maximum ospf-flow of value 14.

1 Introduction

In this paper we study the complexity of routing according to the *Open Shortest Path First protocol* (OSPF) [1].

The goal of a routing protocol is to make every router capable of deciding, whenever it receives a packet, the next hop router. The decision must be taken locally and quickly, while allowing an efficient usage of the network resources. When OSPF is the routing protocol used in the network, all packets follow shortest paths, according to the link lengths set by the network administrator.

When there are several shortest paths between two nodes u and v , the routing depends on the rule that is used for load balancing the traffic among the shortest paths. There are several rules. One of the most used is ECMP (Equal Cost Multiple Path). According to this rule, a router which has several outgoing links on shortest paths toward a destination v , balances the incoming traffic directed to v evenly among all of them.

In order to understand the algorithmic difficulty of OSPF routing, we look into the most essential problem in which the goal is to maximize the throughput when only a *single pair* of nodes is communicating over the network. We call this problem *Max-Ospf-Flow*:

- INSTANCE: a directed capacitated graph $D = (V, A, c)$ where V is the set of nodes, A is the set of arcs, and c is the capacity function that assigns to each arc (u, v) a capacity $c(u, v)$, and two vertices $s, t \in V$, respectively the source and the sink of the flow.
- QUESTION: find the *maximum flow* going from s to t along with its corresponding arc-length assignment under “Open Shortest Path First” protocol with “Equal Cost Multiple Path” strategy.

We give an example of maximum ospf-flow in Figure 1 and a formal description of the problem in Section 2. Let us point out that there may be a large gap between the Max-Ospf-Flow and the standard maximum flow: Figure 2 shows a flow graph with $n + 2$ vertices in which the value of the Max-Ospf-Flow is 2 while the value of the standard maximum flow is n . To the best of our knowledge, this problem has been only proved to be NP-Hard [12].

In this paper, we show that the Max-Ospf-Flow cannot be approximated within any constant factor ratio. Nevertheless, we derive several positive results. First, the problem can be ρ -approximated. This implies that, when the capacities of all arcs are equal, the problem can even

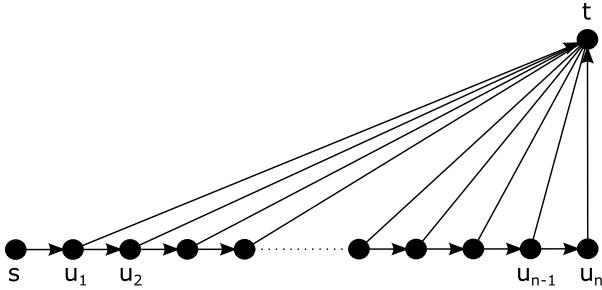


Fig. 2. A flow graph with a big gap between Max-Ospf-Flow and the standard maximum flow. Arcs on the path $s - u_1 - u_2 - \dots - u_n$ have all capacities equal to n while arcs entering t have all capacities equal to 1.

be solved *exactly* in polynomial time. We then present an exact exponential algorithm of complexity $\tilde{O}(2^{|A|})$ [†] and show that any exact algorithm is likely to cost $2^{\Theta(|A|)}$. Finally, we provide an approximation for networks with small longest distance of factor $\frac{1}{2}\mathcal{H}(\lfloor \frac{c_{\max}}{c_{\min}} \rfloor)^{1-L}$ where L is the length of a longest simple path between the source s and the destination t of the flow.

This study shows that the difficulty of Max-Ospf-Flow highly depends on the range of the capacity function and of the “depth” of the network.

Related Work

Some previous works [8], [11], [4] showed that finding an OSPF routing that optimizes some criterias (load, capacity, ...) of the network performance is a difficult task; and this holds for both unsplittable and splittable (with ECMP rule) routings. When the traffic is unsplittable, the lengths have to be set in a way that for every pair of nodes there is a unique shortest path. In [8], the authors studied the problem of optimizing OSPF-ECMP lengths so as to minimize the total cost of the network when the cost function on arcs is convex and increasing with the congestion. They showed that this problem is NP-Hard. In particular, they defined the maximum utilization of the network as $\max_{a \in A} \frac{l(a)}{c(a)}$ where $l(a)$ is the load of arc a , and they proved that minimizing the maximum utilization is NP-Hard. They also provided worst-case results about the performance of OSPF-ECMP routing vs. an optimal multi-commodity flow routing in term of congestion. In [4], the authors studied the unsplittable OSPF that minimizes the maximum utilization of the network. They showed that this problem is hard to approximate within a factor of $O(|V|^{1-\varepsilon})$.

As these problems are difficult, the methods to solve them in practice usually follow a two-phase approach based on linear and integer programming: In the first phase a routing that is not necessarily feasible with OSPF is found and in the second phase lengths that achieve this routing through OSPF are computed, when possible. This second phase attempts to solve what is called the *inverse shortest path problem (ISP)*. This problem can be formulated as a linear program and therefore can be solved in polynomial time. Even if the weights are constrained to take non-negative integer values, ISP remains polynomial thanks to a rounding scheme presented in [2]. However finding a solution to ISP minimizing the maximum length over all arcs is NP-Hard [3].

[†] $\tilde{O}(g(n)) = O(g(n) \cdot \log^k g(n))$, for $k \in \mathbb{N}$. Logarithmic factors are ignored.

In this paper, we consider the problem of approximating the maximum ospf-flow for a network with a single commodity. This problem has been proved to be NP-Hard in [12], using a reduction to set cover. To the best of our knowledge, no approximation with non trivial guaranteed ratio has been suggested before for solving this problem.

To conclude, note that, more generally, the results presented here are valid with minor modifications for other shortest path routing protocols as IS-IS [7] or PNNI [11]. For more information, a survey on the optimization of OSPF routing can be found in [5].

2 Problem Definition

In this section, we show that in the context of our study, which assumes a single commodity, the inverse shortest path problem is guaranteed, under some conditions on the routing, to have a solution and can be easily solved. Therefore, even though Max-Ospf-Flow remains difficult, we can focus on finding an adequate routing function while forgetting about arc-length assignment.

In the case of a single commodity flow, solving the inverse shortest path problem is trivial for any flow f going from s to t . For this we define $A' = \{a \in A \mid f(a) > 0\}$, and set $\forall a \in A', l(a) = 0$ and $\forall a \notin A', l(a) = 1$. Then the shortest paths from the source to the sink are all the paths using only arcs in A' . One may wish for a better length function, such that $\forall a \in A, l(a) > 0$, but it exists iff A' is acyclic. First, if A' contains a cycle then $l(a)$ must be null on all the arcs of the cycle. Second, if A' is acyclic, one easily gets a length function since there exists then a potential function $p : V \rightarrow \mathbb{R}$ such that $p(u) < p(v)$ if there is a path from u to v (this potential function is given by the dual problem, see [6]). Then, one sets $\forall a = (u, v) \in A, l(u, v) = p(v) - p(u)$. Notice that all paths from s to t have length $p(t) - p(s)$. Combining those two facts, one can hence always define lengths such that $\forall (u, v) \in V^2$, all the paths from u to v in A' have the same length and with $l(a) > 0$ on all arcs a not contained in a cycle. Note that, even if it may seem strange, there exist digraphs for which any solution of Max-Ospf-Flow contains a flow loop (see Figure 3).

Therefore, for any flow function, there always exist lengths for which all the paths used by the flow are shortest paths, and this lengths can be non-negative if the flow is acyclic. Nevertheless, it does not mean that any flow function can be achieved using OSPF-ECMP. Indeed f must also fulfill the “Equally Balanced Condition”. This means that we must have: at any node $a \in V$ and $\forall (u, v) \in A$, either $f(u, v) = 0$ or $f(u, v) = \text{out}(u)$ where $\text{out}(u)$ depends only on u . This allows us to reformulate the Max-Ospf-Flow in a simpler way, without using metric explicitly. But before we need to introduce a few definitions.

Definition 2.1 [flow graph] Given a directed capacitated graph $D = (V, A, c)$ and two particular nodes s and t (to be respectively seen as the source and the sink), their corresponding flow graph is defined as the 5-uplet $\mathcal{D} = (V, A, c, s, t)$.

Definition 2.2 [flow function/value] Given a flow graph $\mathcal{D} = (V, A, c, s, t)$, a function $f : A \rightarrow \mathbb{R}^+$ is a flow function if $\forall a \in A, f(a) \leq c(a)$ and $\forall v \in V \setminus \{s, t\}, \sum_{(u,v) \in A} f(u, v) = \sum_{(v,u) \in A} f(v, u)$.

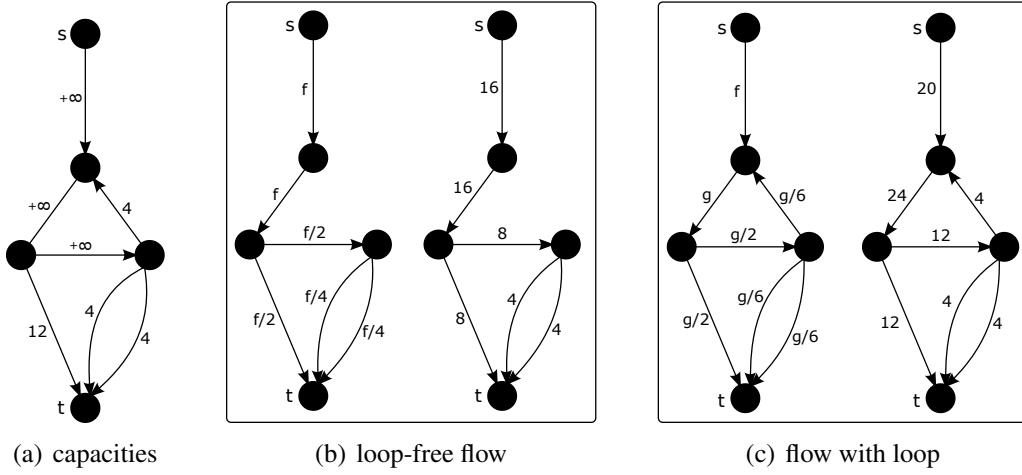


Fig. 3. Example of an instance where the optimal flow can only be achieved using a loop: when avoiding the loop, one can easily check that the limitation comes from the constraint $f/4 \leq 4$ which means that the flow sent from s to t is at most 16; while using a loop the same constraint allows g to be equal to 24 which leads to a flow going from s toward t that is equal to $f = g - g/6 = 20$. Other configurations with less arcs are not depicted but lead to even lower ospf-flow value.

The *flow value* is $\text{val}(f) = \sum_{(s,v) \in A} f(s, v) - \sum_{(v,s) \in A} f(v, s)$.

Definition 2.3 [regular function] A function over a set S is said to be regular if it takes only two values, 0 and another possible one.

Definition 2.4 [balanced function] Given graph $G = (V, A)$ and $V' \subset V$, a function f is said to be balanced on V' iff $\forall u \in V'$ the function f_u , defined by $\forall (u, v) \in A, f_u(v) = f(u, v)$, is regular.

Definition 2.5 [ospf-flow function] For a flow graph $\mathcal{D} = (V, A, c, s, t)$, a function $f : A \rightarrow \mathbb{R}^+$ is an ospf-flow if it is a flow function and if it is balanced on V .

Now, we give an equivalent formulation of the problem **Max-Ospf-Flow**:

- **Instance :** a flow graph $\mathcal{D} = (V, A, c, s, t)$.
 - **Question :** find the maximum flow value of an ospf-flow function in A .

The metric problem has only apparently vanished, since choosing on which arc the flow is null and on which arcs it is strictly positive actually determines the underlying hidden length function.

Some notations: For f a function from a set S to \mathbb{R} , we denote $f(S) = \sum_{s \in S} f(s)$. Given a digraph with vertex set V and arc set A we also abusively denote (U, V) the set of couples $\{(u, v) \in A \mid u \in U, v \in V\}$. We also note the set of integers $\{1, \dots, N\}$ by $[N]$.

3 Inapproximability results

We use a reduction to the MAX-3-SAT problem (which cannot be approximated within a factor $\frac{7}{8} + \varepsilon$ for any constant $\varepsilon > 0$) in order to prove first that Max-Ospf-Flow is hard to approximate, in a polynomial time, within an approximation ratio of $\frac{31}{32}$. Then we use a self-amplifying method to prove that this problem is not in APX. The proofs of following results can be found in the research report [10].

Proposition 3.1 *It is NP-hard to approximate Max-Ospf-Flow within a factor $1 - \frac{l-3}{8l} + \varepsilon, \forall \varepsilon > 0$, even if $\forall a \in A, c(a) \in \{1, l\}$. As example, it is NP-hard to approximate Max-Ospf-Flow within a factor $\frac{31}{32} + \varepsilon, \forall \varepsilon > 0$, even if $\forall a \in A, c(a) \in \{1, 4\}$,*

Theorem 3.2 *It is NP-Hard to approximate Max-Ospf-Flow within any constant factor.*

Sketch of the Proof. To prove the result, we use a self-amplifying error technic. Indeed, the flow graph \mathcal{D}_I associated to an m -clause 3-SAT instance I (in the reduction of Proposition 3.1) connects s to t is like a virtual arc (s, t) with capacity m , but for which it is NP-Hard to use a capacity larger than $\frac{31m}{32}$. Considering a flow graph \mathcal{D} , we will replace each of its edges by those virtual arcs. Two factors of error will then get multiplied: one because no polynomial algorithm can use the full capacity of the virtual arcs, the other because of the error on the original network. \square

4 Positive results

Even though it is hard to find the optimal solution to Max-Ospf-Flow, we can still get some approximation algorithms.

Theorem 4.1 *Given a flow graph $\mathcal{D} = (V, A, c, s, t)$. Let c_{\min} and c_{\max} be respectively the smallest and the largest value of the capacity function c . Then Max-Ospf-Flow can be approximated on \mathcal{D} in polynomial time within a factor c_{\min}/c_{\max} .*

Proof. Let OPT be the maximum value of an ospf-flow on \mathcal{D} , and let \mathcal{D}_i to be the directed graph that have the same elements as \mathcal{D} but where the capacity function is the constant function that assigns i to every arc. We compute a maximum integral flow function F_1 on \mathcal{D}_1 . On the one hand, the value of an ospf-flow on \mathcal{D} cannot exceed the maximum value of a flow on $\mathcal{D}_{c_{\max}}$, which is $c_{\max}v(F_1)$, so $\text{OPT} \leq c_{\max}v(F_1)$. On the other hand, $c_{\min}F_1$ is a feasible ospf-flow for \mathcal{D} with value $c_{\min}v(F_1) \geq \frac{c_{\min}}{c_{\max}}\text{OPT}$. \square

When $c_{\min} = c_{\max}$ we get the next corollary :

Corollary 4.2 *Any instance of Max-Ospf-Flow where all capacities of the flow graph are equal can be solved in polynomial-time.*

We now give an exact algorithm with exponential solving time.

Theorem 4.3 Given a flow graph $\mathcal{D} = (V, A, c, s, t)$, we have an exact algorithm for solving Max-Ospf-Flow on \mathcal{D} which runs in $\tilde{O}(2^{|A|})$.

Proof. Consider an ospf-flow function f_x with a flow value x and remark that the knowledge of the configuration $A' = \{a \in A \mid f_x(a) > 0\}$ entirely determines f_x in the following sense: For a given configuration, there exists at most one ospf-flow of value x . Indeed if $d(v)$ denotes the out-degree in A' of a vertex v the following equations are satisfied:

$$\begin{aligned}\forall v \notin \{s, t\}, \forall (v, w) \in A' \quad f_x(v, w) &= \frac{1}{d(v)} \sum_{(u, v) \in A'} f_x(u, v) \\ \forall (s, w) \in A' \quad f_x(s, w) &= \frac{x}{d(s)}\end{aligned}$$

Note that the above system is well-defined iff all nodes adjacent to some edge in A' belong to a path in A' from s to t . Since the configuration associated to the Max-Ospf-Flow satisfies this condition, we can discard any degenerated set A' violating the above condition.

If f_1 is the function that solves the above system for $x = 1$, then for $x \in \mathbb{R}$ the solution is given by the function $f_x = x \cdot f_1$. Therefore, the maximum flow value of an ospf-flow using the configuration A' , noted $\text{val}(A')$, expresses as the maximum among all $x \in \mathbb{R}$ that respects the following capacity constraints:

$$\forall (v, w) \in A', x f_1(v, w) \leq c(v, w)$$

Hence, we have $\text{val}(A') = \min_{(v, w) \in A'} \frac{c(v, w)}{f_1(v, w)}$.

Finally, to compute the max-ospf-flow, we can simply compute $\text{val}(A')$, in polynomial time, for each non degenerated configuration and take the maximum. \square

We now give a more efficient approximation algorithm when $\frac{c_{\max}}{c_{\min}}$ is large and when the longest path from s to t is short. We call the length of the latter $L_{\mathcal{D}}(s, t)$. The proof is in the research report [10]. The algorithm is based on a sampling argument and on the monotonicity of a (multi-source) ospf-flow, meaning that if a ospf-flow exists, ospf-flows of smaller values always exist.

Theorem 4.4 Let $\mathcal{D} = (V, A, c, s, t)$ be a flow graph, and assume that $c : A \rightarrow [c_{\min}, c_{\max}]$, then Max-Ospf-Flow can be approximated in polynomial time within a factor $\frac{1}{2} \mathcal{H}(\lfloor \frac{c_{\max}}{c_{\min}} \rfloor)^{1-L_{\mathcal{D}}(s, t)}$

5 Conclusion & Open problems

In this study, our main incentive was to investigate the difficulty of routing the data “optimally” in networks that have opted for OSPF with ECMP strategy, as a dynamic routing protocol. To do so, we concentrated on the fundamental problem of maximizing the flow from a single source to a single destination on a network where this protocol is assumed to be running.

The main result shows that there exists no constant factor approximation if the capacities are not bounded. When all capacities are integers in $[c]$, There is a c -factor approximation, but we have been unable to find an algorithm with ratio better than c . So we conjecture that for any $\mu > 0$, Max-Ospf-Flow cannot be approximated within a factor better than $c^{1-\mu}$. Nevertheless, if the length of the longest path from s to t is bounded by L , then we derive an approximation algorithm with the ratio $\frac{1}{2}\mathcal{H}(\lfloor \frac{c_{\max}}{c_{\min}} \rfloor)^{1-L}$.

Additionally, we suggest an exact algorithm that has a complexity of $\tilde{O}(2^{|A|})$. It is an open problem to find an algorithm with a better exponential base. However, there is a limitation on the efficiency of such algorithm, since it would be able to solve MAX-3-SAT (see the reduction of Theorem 3.1) and the best known algorithm solving an m -clause instance has complexity $\tilde{O}(1.3^m)$.

References

- [1] *Ospf version 2, rfc2328*, <http://www.ietf.org/rfc/rfc2328.txt> (1998).
- [2] Ben-Ameur, W. and E. Gourdin, *Internet routing and related topology issues*, SIAM J. Discret. Math. **17** (2004), pp. 18–49.
- [3] Bley, A., *Inapproximability results for the inverse shortest paths problem with integer lengths and unique shortest paths*, in: *Proc. of the Second International Network Optimization Conference*, 2005.
- [4] Bley, A., *On the approximability of the minimum congestion unsplittable shortest path routing problem*, in: *Proceedings of the 11th international conference on Integer Programming and Combinatorial Optimization* (2005), pp. 97–110.
- [5] Bley, A., B. Fortz, E. Gourdin, K. Holmberg, O. Klopfenstein, M. Pioro, A. Tomaszewski and H. Umit, “Optimization of OSPF routing in IP networks,” Springer, 2009 pp. 199–240, in *Graphs and Algorithms in Communication Networks: Studies in Broadband, Optical, Wireless and Ad Hoc Networks*.
- [6] Cook, W. J., W. H. Cunningham, W. R. Pulleyblank and A. Schrijver, “Combinatorial optimization,” Wiley, 1998.
- [7] Fortz, B. and M. Thorup, *Optimizing ospf/is-is weights in a changing world*, IEEE Journal on Selected Areas in Communications **20** (2002), pp. 756–767.
- [8] Fortz, B. and M. Thorup, *Increasing internet capacity using local search*, Computational Optimization and Applications **29** (2004), pp. 13–48.
- [9] Håstad, J., *Some optimal inapproximability results*, Journal of the ACM **48** (2001), pp. 798–859.
- [10] Issam, T., S. Pérennes and F. Giroire, *On the Hardness of Equal Shortest Path Routing*, Rapport de recherche RR-8175, INRIA (2012). URL <http://hal.inria.fr/hal-00763239>
- [11] Izmailov, R., B. Sengupta and A. Iwata, *Administrative weight allocation for pnni routing algorithms*, in: *IEEE Workshop on High Performance Switching and Routing*, 2001, pp. 347–352.
- [12] Pióro, M., Á. Szentesi, J. Harmatos, A. Jüttner, P. Gajowniczek and S. Kozdorowski, *On open shortest path first related network optimisation problems*, J. of Performance Evaluation **48** (2002), pp. 201–223.

(k, c) – coloring via Column Generation

E. Malaguti¹

*DEI, University of Bologna
Bologna, Italy*

I. Méndez-Díaz^{2,3}

*Computer Science Department, University of Buenos Aires
Buenos Aires, Argentina*

J. J. Miranda-Bront and P. Zabala^{2,3}

*Computer Science Department, University of Buenos Aires
Consejo Nacional de Investigaciones Científicas y Técnicas
Buenos Aires, Argentina*

Abstract

In this paper we study the (k, c) -coloring problem, a generalization of the well known *Vertex Coloring Problem* (VCP). We propose a new formulation and compare it computationally with another formulation from the literature. We also develop a diving heuristic that provides with good quality results at a reasonable computational effort.

Keywords: (k, c) -coloring, column generation, diving heuristic

1 Introduction

In the Vertex Coloring Problem (VCP), one is required to assign a color to each vertex of an indirected graph in such a way that adjacent vertices re-

¹ Email: enrico.malaguti@unibo.it

² Email: [\(imendez,jmiranda,pzabala\)@cdc.uba.ar](mailto:(imendez,jmiranda,pzabala)@cdc.uba.ar)

³ This work was partially supported by UBACYT 20020100100666, PICT 304 and 817

ceive different colors, and the objective is to minimize the number of the used colors. Several problems where a resource (color) has to be shared among conflicting users (vertices connected by an edge in the graph) can be modeled as VCPs. Some problems can be represented as generalizations of the VCP. In the multicoloring problem, for example, users require more than one copy of the resource (see, e.g., [4]). In some applications the resource cannot be duplicated more than a fixed amount of times, and it is acceptable to have the resource partially shared among conflicting users (this happens in frequency assignment, see, e.g., Aardal et al., [1]). In this paper we consider a generalization of the VCP where each vertex has to receive more than one color, and each pair of conflicting vertices can share a limited number of colors.

Formally, the (k, c) -coloring problem is defined as follows. Let $G = (V, E)$ be an undirected graph, with $V = \{1, \dots, n\}$ the set of vertices and $E = \{uv : u, v \in V, u \neq v\}$ the set of edges, and $R = \{1, \dots, |R|\}$ the set of available colors. Each vertex $v \in V$ is required to be assigned exactly k different colors and each pair of adjacent vertices u, v cannot share more than c colors. The objective is to minimize the total number of colors used. The problem is $NP-hard$ and reduces to the VCP when $k = 1$ and $c = 0$ (see Garey and Johnson [2] for complexity results on VCP, and Malaguti [6] and Malaguti and Toth [8] for other $NP-hard$ generalizations of the VCP).

In this paper, we propose a new formulation for (k, c) -coloring problem. Based on the results recently achieved by Malaguti et al. [7], Gualandi and Malucelli [3] and Held et al. [5] for the VCP using set covering formulations, we propose a model for the (k, c) -coloring following these ideas. We experimentally evaluate and compare it with another model from the related literature, and develop a diving heuristic, which produces good quality solutions.

The rest of the paper is organized as follows. In Section 2 we describe the model from Méndez-Díaz and Zabala [9] and propose a new formulation for the (k, c) -coloring. In Section 3 we describe the diving heuristic and in Section 4 we present some preliminary computational results, making a comparison of both formulations and evaluating the diving heuristic. Finally, in Section 5 we present some conclusions and propose future lines of research regarding the (k, c) -coloring.

2 Models

We begin by showing the model proposed by Méndez-Díaz and Zabala [9], slightly modified for the particular case of the (k, c) -coloring problem. They consider three different sets of variables. The first one regards binary variables

x_{vj} , for $v \in V$ and $j \in R$, taking value 1 if and only if color j is assigned to vertex v . Secondly, for each arc $u, v \in E$ and a color $j \in R$, binary variable $y_{uvj} = 1$ iff color j is assigned to both u and v . Finally, binary variables w_j , $j \in R$, take value 1 iff color j is used by some vertex. The (k, c) -coloring problem is formulated as:

$$\begin{aligned}
(1) \quad & \min \sum_{j \in R} w_j \\
(2) \quad & \text{s.t.: } \sum_{j \in R} x_{vj} = k \quad v \in V \\
(3) \quad & \sum_{j \in R} y_{uvj} \leq c \quad uv \in E \\
(4) \quad & x_{uj} + x_{vj} - y_{uvj} \leq 1 \quad uv \in E, j \in R \\
(5) \quad & x_{vj} \leq w_j \quad v \in V, j \in R \\
(6) \quad & x_{vj} \in \{0, 1\} \quad v \in V, j \in R \\
(7) \quad & y_{uvj} \in \{0, 1\} \quad uv \in E, j \in R \\
(8) \quad & w_j \in \{0, 1\} \quad j \in R
\end{aligned}$$

The objective function (1) minimizes the number of colors used. Constraints (2) establish that exactly k colors are assigned to vertex v and constraints (3) restrict the number of colors that can be shared by two adjacent vertices. Constraints (4) impose $y_{uvj} = 1$ if color j is assigned to both u and v , and constraints (5) set $w_j = 1$ if color j is used by some vertex. Finally, constraints (6) - (8) establish that all variables must be binary.

Similarly as with the VCP, this formulation allows symmetric solutions. To eliminate some symmetries, the authors propose to include in the formulation inequalities $w_j \leq w_{j+1}$, for $1 \leq j \leq |R| - 1$.

In this paper, we propose a new formulation which does not suffer from symmetry issues. Let $S = 2^V$ be the power set of V , and define integer variables x_s , $s \in S$, representing the number of colors assigned to all vertices in s . Regarding the standard formulation for the VCP, it is important to note that in our model any subset of vertices can be feasibly colored by one color when $c > 0$. The (k, c) -coloring problem can be formulated as follows:

$$\begin{aligned}
(9) \quad & \min \sum_{s \in S} x_s \\
(10) \quad & \text{s.t. } \sum_{s \in S: v \in s} x_s \geq k, \quad v \in V
\end{aligned}$$

$$(11) \quad \sum_{s \in S: uv \in s} x_s \leq c, \quad uv \in E$$

$$(12) \quad x_s \in \mathbb{Z}^+, \quad s \in S$$

The objective function (9) minimizes the number of colors used. Constraints (10) establish that a vertex must receive at least k colors. It is important to note that these constraints modify slightly the definition (k, c) -coloring, where each vertex is required to have assigned exactly k colors, in order to be able to consider maximal sets $s \in S$ and therefore speed up the column generation algorithm. In this context, we let a set $s \in S$ to be maximal if no vertex v can be included into the set without adding new edges. Furthermore, any solution obtained by this formulation can be easily transformed into one having exactly k colors. Finally, constraints (11) restrict the number of colors assigned to pairs of adjacent vertices in G and constraints (12) require variables x_s , $s \in S$, to be nonnegative integers.

Model (9) - (12) has an exponential number of variables and therefore it cannot be formulated explicitly, even for medium size instances (i.e., having a few hundreds of nodes). Branch-and-Price algorithms have been proven to be quite effective in solving this kind of models, using column generation to solve the LP relaxation (usually called *master problem*). The main idea behind this technique is to start with a restricted set of columns, obtaining as a result a restricted master problem, and iteratively add columns with negative reduced cost until the master problem is solved to optimality. For this purpose, let (π, ρ) be the optimal values of the dual variables associated with constraints (10) and (11), respectively. To account for variables x_s with negative reduced costs, we formulate the following slave problem where binary variables z_v , $v \in V$, take value one if and only if vertex v is in the set and, for each $uv \in E$, y_{uv} is forced to 1 when both u and v belong to the set.

$$(13) \quad \max \sum_{v \in V} \pi_v z_v - \sum_{uv \in E} \rho_{uv} y_{uv}$$

$$(14) \quad \text{s.t.} \quad z_u + z_v - y_{uv} \leq 1, \quad uv \in E$$

$$(15) \quad z_v \in \{0, 1\}, \quad i \in V$$

$$(16) \quad y_{uv} \in \{0, 1\}, \quad uv \in E$$

If the objective value of the optimal solution of the slave problem is less than or equal to one, then the master problem has been solved to optimality. Otherwise, the optimal solution represents a feasible set with negative reduced cost, we add the column to the restricted master problem and iterate again.

3 Diving heuristic

Aiming to obtain good solutions for the (k, c) -coloring, in this section we present a diving heuristic developed using model (9) - (12). The algorithm iteratively tries to fix variables and updates the lower and upper bounds using the information provided by the LP relaxation, expecting to obtain at the end a feasible solution for the problem.

In order to solve the LP relaxation of model (9) - (12), we initialize the restricted master problem with the following columns:

- Sets $s = \{i\}$, for $i = 1, \dots, n$, and
- sets s represented by the solution obtained by the greedy heuristic proposed in Méndez-Díaz and Zabala [9]. This heuristic starts with an initial color list $L = \{1, \dots, k\}$ and iteratively selects a vertex to be colored with the first k compatible colors in L . In case this is not possible, L is expanded by including new colors.

Another relevant aspect of the column generation algorithm is to be able to determine whether the master problem has been solved to optimality or if there exists a column with negative reduced cost. For this purpose, we solve the slave problem (13) - (16) using a general purpose MIP algorithm. The sketch of the diving heuristic is described in Algorithm 1.

- Algorithm 1**
- (i) Initialize the master problem setting as columns sets $s = \{i\}$, for $i = 1, \dots, n$ and the solution provided by the heuristic. Set bestUB with the number of colors used.
 - (ii) Solve the master problem of model (9) - (12). Let x^* be the optimal solution and z^* the objective value. If $\lceil z^* \rceil \geq \text{bestUB}$, then exit the algorithm and return bestUB .
 - (iii) If x^* is integer, exit the algorithm and return z^* . Otherwise, set in the master problem lower bounds $x_s = x_s^*$ for all variables x_s having integer values in x^* .
 - (iv) Let \bar{s} be the index of the fractional variable which is closest to its rounded-up value. Set $x_{\bar{s}} \geq \lceil x_{\bar{s}}^* \rceil$ and solve the modified restricted master problem. If the problem is feasible, update the master problem with $x_{\bar{s}} \geq \lceil x_{\bar{s}}^* \rceil$ and go to step (ii). Otherwise, Set $x_{\bar{s}} \leq \lfloor x_{\bar{s}}^* \rfloor$ in the master problem and go to step (ii)

4 Computational results

We conducted experiments in order to evaluate the quality of the lower bounds obtained by the linear relaxation of model (9) - (12) as well as the solutions obtained by the diving heuristic.

The experiments were run on a workstation with an Intel(R) Core(TM) i7 CPU (3.40GHz) and 16 Gb of RAM. The algorithms are coded in C++ using CPLEX 12.1 Callable Library as LP and MIP solver. We consider a set of instances with random generated graphs having 20 vertices and varying the density in terms of the number of edges, considering values of 10, 20, ..., 90 percent of the edges. For each of these values we consider 10 instances, and we group them in Low (10% - 30%), Medium (40% - 60%) and High (70% - 90%) density, considering increasing values for this parameter.

We evaluated 3 different methods: (i) the Branch and Cut algorithm from Méndez-Díaz and Zabala [9] (BC), (ii) the diving heuristic described in Section 3 (DH-GH), and (iii) the diving heuristic but without including the columns obtained from the greedy heuristic (DH-noGH). We impose a time limit of 600 seconds for the three algorithms.

In Table 1 we present the average optimality gaps ($100*(UB - LB)/LB$) and computational times obtained by each algorithm. The average computational times are calculated only over instances for which the corresponding algorithm finishes before the time limit. A cell filled with *** means that the algorithm was not able to solve any of the instances, and a number between parenthesis stands for the number of instances effectively solved within the time limit. We first observe that the diving heuristic obtains better results in all cases when the columns provided by the greedy heuristic are considered to initialize the master problem. As regards the comparison between BC and DH-GH, the results are somehow mixed. BC produces the best results when k and c are close, but when this difference tends to grow DH-GH outperforms BC (see, e.g., (4, 1), (5, 1) and (5, 2)) finding the optimal solution in many cases. Furthermore, it also improves the initial solution provided by the greedy heuristic in more cases than BC.

In terms of computational times, although not reported due to space limitations, the behavior of DH-GH is reasonable and could be improved by developing an effective heuristic algorithm for finding columns with negative reduced costs instead of solving the slave problem to optimality. Furthermore, in several cases we observed that the quality of the lower bounds of the LP relaxation of model (9) - (12) tends to be better than the one produced by BC without a particular strengthening obtained by adding valid inequalities (see [9]) and, on some instances, it also obtains similar values even considering it.

(k, c)	Density	BC		DH-GH		DH-noGH	
		Time	%gap	Time	%gap	Time	%gap
(3, 1)	Low	0.28	0.00	3.03	1.11	3.04	0.56
	Med.	(29) 7.70	0.56	25.63	0.95	28.80	0.95
	High	(11) 138.53	12.14	(26) 177.75	22.92	(26) 181.10	104.58
(3, 2)	Low	0.04	0.00	0.87	0.00	1.11	0.00
	Med.	0.40	0.00	5.93	2.67	6.96	4.67
	High	40.30	0.00	108.83	22.00	112.38	27.33
(4, 1)	Low	(29) 0.68	0.33	2.50	0.00	3.02	0.33
	Med.	(10) 79.27	11.33	39.34	1.80	40.87	2.71
	High	***	39.00	(28) 273.85	4.43	(28) 271.49	37.72
(4, 2)	Low	0.60	0.00	1.55	0.00	1.38	0.48
	Med.	58.67	0.00	26.59	2.38	28.37	2.38
	High	(14) 165.26	7.86	(23) 194.69	16.67	(25) 225.22	151.67
(4, 3)	Low	0.12	0.00	0.62	0.00	0.88	0.67
	Med.	23.15	0.00	5.38	0.00	5.89	0.00
	High	12.76	0.00	54.74	11.11	55.92	15.00
(5, 1)	Low	(28) 0.89	0.52	1.02	0.00	1.33	0.24
	Med.	(6) 15.26	12.62	39.81	0.43	41.61	0.82
	High	***	44.64	(27) 223.63	4.67	(28) 243.99	30.59
(5, 2)	Low	(29) 1.94	0.37	2.31	0.33	2.32	0.67
	Med.	(13) 74.01	6.67	44.04	5.58	42.45	5.88
	High	***	33.00	(25) 239.38	10.05	(25) 234.75	129.39

Table 1
Percentage optimality gap and computational times (averages).

5 Conclusions

In this paper, we present a new formulation for the (k, c) -coloring problem and a diving heuristic which obtains good quality results in a reasonable amount of time. Furthermore, the diving heuristic outperforms one of the recent algorithms proposed in the literature on instances where the difference between k and c tends to grow.

As future research, and aiming to obtain a Branch-and-Price algorithm, it is necessary to focus on two major aspects of the problem. Firstly, it is important to develop a heuristic procedure for solving the slave problem (13) - (16) efficiently in order to accelerate the overall time required by the column generation algorithm.

Secondly, it would be interesting to obtain a deeper insight on the structure of the (k, c) -coloring problem in order to derive a *robust* branching rule. For the case where $c = k - 1$, one possibility is to consider a generalization of the idea used for the VCP (see, e.g., [7], [3], [5]). Let $i, j \in V$ be two non-adjacent vertices, and consider splitting the original problem into two new subproblems

in the following fashion:

- i and j have share exactly k colors, so we can collapse them into one new vertex, or
- i and j share at most $k - 1$ colors, so we can add an edge between i and j .

However, when $c \neq k - 1$ this rule cannot be applied directly, since we obtain as a result a subproblem where the number of maximum colors that two adjacent vertices can share is not fixed. In this regard, further theoretical and practical developments are required.

References

- [1] Aardal, K., S. van Hoesel, A. Koster, C. Mannino and A. Sassano, *Models and solution techniques for the frequency assignment problem*, 4OR **1** (2003), pp. 261–317.
- [2] Garey, M. and D. Johnson, “Computers and Intractability: A Guide to the Theory of NP-Completeness,” DIMACS Series in Discrete Mathematics and Theoretical Computer Science, Freedman, New York, 1979.
- [3] Gualandi, S. and F. Malucelli, *Exact solution of graph coloring problems via constraint programming and column generation*, INFORMS Journal on Computing **24** (2012), pp. 81–100.
- [4] Halldórsson, M. and G. Kortsarz, *Multicoloring: Problems and techniques*, in: *Mathematical Foundations of Computer Science 2004*, Lecture Notes in Computer Science **3153**, 2004, pp. 25–41.
- [5] Held, S., W. Cook and E. Sewell, *Maximum-weight stable sets and safe lower bounds for graph coloring*, Mathematical Programming Computation **4** (2012), pp. 363–381.
- [6] Malaguti, E., *The vertex coloring problem and its generalizations*, 4OR **7** (2009), pp. 101–104.
- [7] Malaguti, E., M. Monaci and P. Toth, *An exact approach for the vertex coloring problem*, Discrete Optimization **8** (2011), pp. 174–190.
- [8] Malaguti, E. and P. Toth, *A survey on vertex coloring problems*, International Transactions in Operational Research **17** (2010), pp. 1–34.
- [9] Méndez-Díaz, I. and P. Zabala, *Solving a multicoloring problem with overlaps using integer programming*, Discrete Applied Mathematics **158** (2010), pp. 349–354.

The final answer to the complexity of a basic problem in resilient network design¹

Artur Tomaszewski²

*Institute of Telecommunications
Warsaw University of Technology, Warszawa, Poland*

Abstract

To provide resilience to failures of the multi-commodity flow network, either in the failure-free state flows can be routed along multiple paths and over-dimensioned, or whenever a failure occurs flows can be restored along unaffected paths. The complexity of the network design depends on the selected method of providing resilience and on a number of design options—whether single or multiple commodities and single- or multi-element failures are considered, if the reaction to failures is dependent or independent on the failure, which mechanism of capacity release and reuse is applied, etc. For almost all combinations of those choices either the corresponding design problem has already been shown to be NP-hard or a compact linear programming formulation of the problem has been provided. The only case that has resisted an answer is when flows are restored in a state-dependent manner using the stub release mechanism. In this paper it is proved that the corresponding network design problem is NP-hard even for a single commodity and for single-element failures. The proof is based on the reduction of the Hamiltonian path problem.

Keywords: complexity theory, multi-commodity flow networks, resilient design

¹ The author was supported by National Science Center (NCN, Poland) under grant 2011/01/B/ST7/02967 Integer programming models for joint optimization of link capacity assignment, transmission scheduling, and routing in fair multicommodity flow networks

² Email: artur@tele.pw.edu.pl

1 Introduction

The multi-commodity flow network can be made resilient to link and node failures (assuring that for the nominal, failure-free state and for every failure state the required total flow of every commodity is preserved) using one of the two mechanisms: with path diversity the commodity flows are over-dimensioned in the nominal state and routed along multiple paths so that for any failure the total unaffected flow is not less than the demand volume; with flow restoration when failures occur flows are restored or rerouted along unaffected paths.

With the design of optimal multi-commodity flow networks resilient to failures being a practically important issue, e.g., in the telecommunications network planning (cf. [8,11,6]), the question about the tractability and complexity ([1]) of the design problem is of primary concern. That complexity is heavily dependent upon both the adopted mechanism of providing resilience and a number of design options. The most straightforward one is whether single- or multi-element failures are taken into account and if these are failures of links or nodes. The additional, less obvious options pertain to flow restoration: they specify if the reaction to failures is dependent or independent on the failure that occurs, and which mechanism of capacity release and reuse is applied. The distinction between the state-independent and state-dependent restoration is based on whether it is required or not that the paths used for restoring the affected flows should be the same for every network state. The options for the capacity release and reuse are as follows: restricted restoration—affected flows can be restored using only unused (redundant) capacity of links; restricted restoration with stub release—the capacity of healthy links used by the affected flows in the nominal state (stubs) can be released and then reused when restoring flows; full restoration—all flows, both affected and unaffected, can be first disconnected to release all capacity and then restored.

For almost every combination of the design choices either a compact linear programming formulation of the corresponding design problem has already been given providing for solving the problem in polynomial time, or the design problem has been shown to be \mathcal{NP} -hard; [7] provides the corresponding survey. The simplest case of resilient network design, and the most classical one, arises when state-dependent full restoration is considered (cf., e.g., [8]). Even when multi-element failures are assumed, the node-link formulation can be used to express the flows, so that the resulting design problem formulation is a compact linear programme. Similar compact formulations have also been proposed for path diversity and for state-dependent flow restoration without

capacity release and reuse (cf. [10]). However, the latter are valid only when single-element failures are assumed; otherwise, the problems have been shown to become \mathcal{NP} -hard even for a single commodity (cf. [10]), which means, in particular, that compact linear formulations of the problems do not exist. For state-independent restoration of flows, the problems are always \mathcal{NP} -hard, even for a single commodity and for single-element failures (cf. [12]).

Thus, the only problem that has yet lacked a decisive answer is state-dependent restricted flow restoration with stub release (RR). For a single commodity and for single-link failures that problem can be stated as follows:

Given a directed network $\mathcal{G}(\mathcal{V}, \mathcal{E})$ with set \mathcal{V} of vertices (nodes) and set \mathcal{E} of arcs (links), a per-unit reservation cost $\alpha : \mathcal{E} \rightarrow \mathcal{R}_+$, two nodes s and t , a set $\mathcal{S} \subset \mathcal{E}$ of arcs that can fail with $|S|$ unbounded, and a set \mathcal{P} of, possibly non-simple, $s - t$ directed paths, **find** a minimum-cost capacity reservation $c : \mathcal{E} \mapsto \mathcal{R}_+$ and flow assignments $g : \mathcal{P} \rightarrow \mathcal{R}_+$ and $h : \mathcal{P} \times \mathcal{S} \mapsto \mathcal{R}_+$, **such that** for any arc $s \in \mathcal{S}$ the following holds: $\sum_{p \in \mathcal{P}} g(p) \geq 1$ and $\sum_{p \in \mathcal{P}: s \notin p} g(p) + \sum_{p \in \mathcal{P}: s \notin p} h(p, s) \geq 1$; and $\sum_{p \in \mathcal{P}: e \in p} g(p) \leq c(e)$ and $\sum_{p \in \mathcal{P}: s \notin p \wedge e \in p} g(p) + \sum_{p \in \mathcal{P}: s \notin p \wedge e \in p} h(p, s) \leq c(e)$ for each $e \in \mathcal{E}$.

The simplest instance of the problem is presented in Figure 1a. If failure f_2 of link b occurs, the capacity of link a is released, and flow g is restored as flow h_2 using the released capacity of link a and the redundant capacity of link b .

A common linear programming formulation of the RR problem (cf. [5]) is non-compact as it assumes a given set of network paths for the nominal and for all failure states. Although, in theory the problem could be effectively solved by column generation, the problem of generating nominal paths is \mathcal{NP} -complete (the proof is due to J-F. Maurras and A. Bley; cf. [2], [5]).

The most recent results of the analysis of the RR problem are contained in [3], [4] and [9]. They pertain to two specific cases when the number of failures is fixed and only simple paths are used. In [3], for a particular case of just one single-link failure, the proof is provided that, on the one hand, the non-compact formulation resulting from the general linear programming formulation of the RR problem still exhibits \mathcal{NP} -hard dual separation, but on the other hand, the RR problem is polynomial—both a compact and a non-compact linear formulation with polynomial path generation exist. In [4] it is shown that for any fixed number of single-link failures the complexity of the RR problem is still polynomial. And finally, in [9] it is proved that if the flows are required to use simple paths the RR problem is \mathcal{NP} -hard even for one commodity and for just two single-link failures.

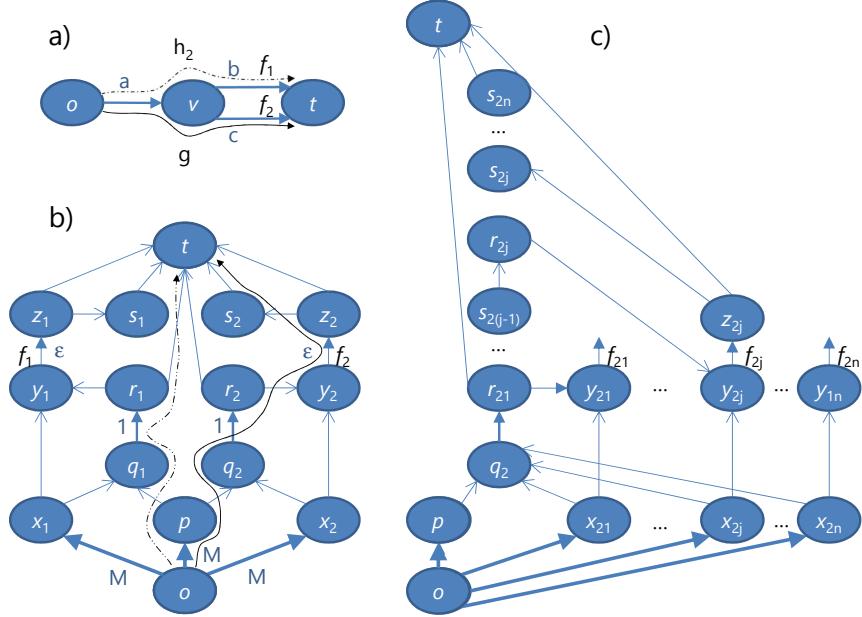


Fig. 1. RR problem instances

In this paper the general problem of state-dependent restricted restoration design is revisited, assuming that the arcs can fail one at a time but the number of failures is not fixed, and paths need not be simple (in [4] examples are provided that in general better optimal solutions can be obtained if loops are allowed, even for a single-commodity and two single-link failures). First, a problem instance that assumes a single commodity and two single-link failures is discussed. Then, it is generalized so as to allow for any number of single-link failures. Finally, the generalized instance is used to prove that the RR problem is \mathcal{NP} -hard by a reduction from the Hamiltonian path (HP) problem.

2 Result

Though simple, the instance of the RR problem illustrated in Figure 1b builds intuition on the state-dependent restricted restoration and on the sources of the mechanism’s effectiveness. The network graph is symmetrical—the left- and the right-hand parts of the graph are identical. The capacity-unit costs of links are equal to: $M \gg 1$ for links (o, p) and (o, x_i) ; 1 for (q_i, r_i) ; $\varepsilon \ll 1$ for (y_i, z_i) ; 0 for other links (marked with non-solid arrowheads). There are two single-link failures $f_i = (y_i, z_i)$, $i = 1, 2$, and there is one commodity with source node o , destination node t , and flow demand volume equal to 1.

It can be checked that the total network cost of $M + \frac{2}{3} + \varepsilon$ is achieved if the following routing of flows is adopted: in the nominal state the total flow of $\frac{1}{3}$ along two paths $\mathcal{P}_i = (o, p, q_i, r_i, y_i, z_i, s_i, t)$, $i = 1, 2$, and the flows of $\frac{1}{3}$ along each path $\mathcal{Q}_i = (o, x_i, y_i, z_i, t)$, $i = 1, 2$; at failure f_i the flow of path \mathcal{P}_i is restored along path $\mathcal{R}_j = (o, p, q_j, r_j, t)$, $j = 3 - i$ and the flow of path \mathcal{Q}_i along path $\mathcal{S}_i = (o, x_i, q_i, r_i, t)$. The purpose of using failing link (y_2, z_2) in path \mathcal{P}_2 is to allow disconnecting the respective flow when failure f_2 occurs and restoring it along path \mathcal{R}_1 (the nominal flow is shown with the solid line, and the restored flow with dotted one) in order to release link (q_2, r_2) for the failing flow of \mathcal{Q}_2 ; this provides for effective sharing of the capacity of links (q_1, r_1) and (q_2, r_2) . Due to the above cost, if in the nominal state a path was used that does not contain failing links (e.g., path (o, p, q_i, r_i, t)) the network cost cannot be minimal as the capacity-unit cost of any such path is $M + 1$.

Proposition 2.1 *The resilient network design problem assuming state-dependent flow restoration with stub release is NP-hard even for a single demand and single-link failures.*

Proof.

The proposition is proved in two steps using the generalization of the problem instance from Figure 1b illustrated in Figure 1c: for $i = 1, 2$, there are n triples of nodes x_{ij} , y_{ij} and z_{ij} , $j = 1, 2, \dots, n$, connected in parallel, and n pairs of nodes r_{ij} and s_{ij} , $j = 1, 2, \dots, n$, connected in series, distinguished with additional index; thus, there are $2n$ single-link failures in total. The first step is to prove that the cost of an optimal solution of that RR problem instance is equal $\pi^* = M + \frac{2}{2n+1} + \frac{3n}{2n+1}\varepsilon$, and that this cost is attained solely for the following routing of flows in the nominal state: the total flow of $\frac{1}{2n+1}$ along two paths $\mathcal{P}_i = (o, p, q_i, r_{i1}, y_{i1}, z_{i1}, s_{i1}, r_{i2}, y_{i2}, z_{i2}, s_{i2}, \dots, r_{in}, y_{in}, z_{in}, s_{in}, t)$, $i = 1, 2$, and the flows of $\frac{1}{2n+1}$ along each path $\mathcal{Q}_{ij} = (o, x_{ij}, y_{ij}, z_{ij}, t)$, $i = 1, 2$, $j = 1, 2, \dots, n$. The second step is to show that an arbitrary HP problem instance can be reduced to the RR problem instance from Figure 1c.

For the given routing of flows cost π^* is attained if at failure f_{ij} the flow of path \mathcal{P}_i is restored along path $\mathcal{R}_j = (o, p, q_j, r_{j1}, t)$, $j = 3 - i$, and the flow of path \mathcal{Q}_{ij} along path $\mathcal{S}_i = (o, p, q_i, r_{i1}, t)$. Now, consider an arbitrary solution for which the cost is not greater than π^* . Let $c(a, b)$ and $f(a, b)$ denote, respectively, the capacity and the nominal flow of edge (a, b) . Obviously,

$$f(o, p) + \sum_{i,j} f(o, x_{ij}) = 1 \quad (2.1)$$

and hence, capacity $c(o, p) + \sum_{i,j} c(o, x_{ij})$ must be at least 1, but considering

the total cost it must be equal to 1. Thus, no redundant capacity should be used on links (o, p) and (o, x_{ij}) . Therefore, for any failure failed flows that use those links must still use them, and can only be rerouted to links (q_1, r_{11}) and (q_2, r_{21}) . So the following must hold:

$$c(q_i, r_{i1}) \geq f(o, x_{ij}) \quad i = 1, 2; j = 1, 2, \dots, n \quad (2.2a)$$

$$c(q_1, r_{11}) + c(q_2, r_{21}) \geq f(o, p) + f(o, x_{ij}) \quad i = 1, 2; j = 1, 2, \dots, n. \quad (2.2b)$$

Additionally, $c(q_1, r_{11}) + c(q_2, r_{21}) \leq \frac{2}{2n+1}$, otherwise the total cost would be greater than π^* . Then, by (2.1) and (2.2a): $f(o, p) = 1 - \sum_{i,j} f(o, x_{ij}) \geq 1 - n[c(q_1, r_{11}) + c(q_2, r_{21})] \geq 1 - n\frac{2}{2n+1} = \frac{1}{2n+1}$. And by (2.2b) and (2.1): $\frac{2}{2n+1} \geq c(q_1, r_{11}) + c(q_2, r_{21}) \geq f(o, p) + \max_{i,j} f(o, x_{ij}) \geq f(o, p) + \frac{1}{2n} \cdot (1 - f(o, p)) \geq f(o, p) + \frac{1}{2n+1}$. So $f(o, p) = \frac{1}{2n+1}$, and also $f(o, x_{ij}) = \frac{1}{2n+1}$, $i = 1, 2, j = 1, 2, \dots, n$, as $f(o, x_{ij}) \leq \frac{1}{2n+1}$ by (2.2b) and $\sum_{i,j} f(o, x_{ij}) = \frac{2n}{2n+1}$ by (2.1).

In the nominal state each flow must use at least one failing link, otherwise its capacity-unit cost would be $M + 1$, which is greater than π^* , so the total network cost would not be minimal. Additionally, since $c(q_1, r_{11}) = c(q_2, r_{21}) = \frac{1}{2n+1}$ by (2.2a), for all $i = 1, 2, j = 1, 2, \dots, n$, at failure f_{ij} link (q_i, r_{i1}) must be totally freed to accommodate flow $f(o, x_{ij}) = \frac{1}{2n+1}$ (it has to be rerouted to that link), and thus flow $f(p, q_i)$ must go through link (y_{ij}, z_{ij}) in the nominal state to allow for stub release. Therefore, $\sum_{i,j} c(y_{ij}, z_{ij}) \geq \sum_{i,j} f(o, x_{ij}) + \sum_j f(p, q_1) + \sum_j f(p, q_2) = \sum_{i,j} f(o, x_{ij}) + \sum_j f(o, p) = \frac{2n}{2n+1} + \frac{n}{2n+1} = \frac{3n}{2n+1}$. Thus π^* is the optimal cost which is attained solely for the described routing.

Next, a reduction from the Hamiltonian path (HP) problem is shown. The directed version of the HP problem consists in answering the question whether for a given directed graph \mathcal{G} and a given pair of nodes (s, t) , \mathcal{G} contains a simple $s-t$ path that traverses each node from \mathcal{G} . Tightly related to the Hamiltonian cycle problem, the HP problem is \mathcal{NP} -complete.

Consider a general instance of the HP problem as presented in Figure 2a: in graph \mathcal{G} with n nodes v_1, v_2, \dots, v_n it is required to find a Hamiltonian path from v_1 to v_n . To construct the corresponding instance of the RR problem the following transformation of graph \mathcal{G} into graph \mathcal{H} presented to the right will be used: each node v_j , $j = 1, 2, \dots, n$, is replaced with a pair of nodes r_j and s_j ; r_j terminates the incoming edges of v_j and s_j terminates the outgoing edges of v_j ; all those edges have the cost equal to 0. The instance of the RR problem (the right half of it) is constructed by embedding two copies of graph \mathcal{H} into the problem instance from Figure 1c (cf. Figure 2b).

It can be shown that the HP problem instance has a solution, i.e., a Hamil-

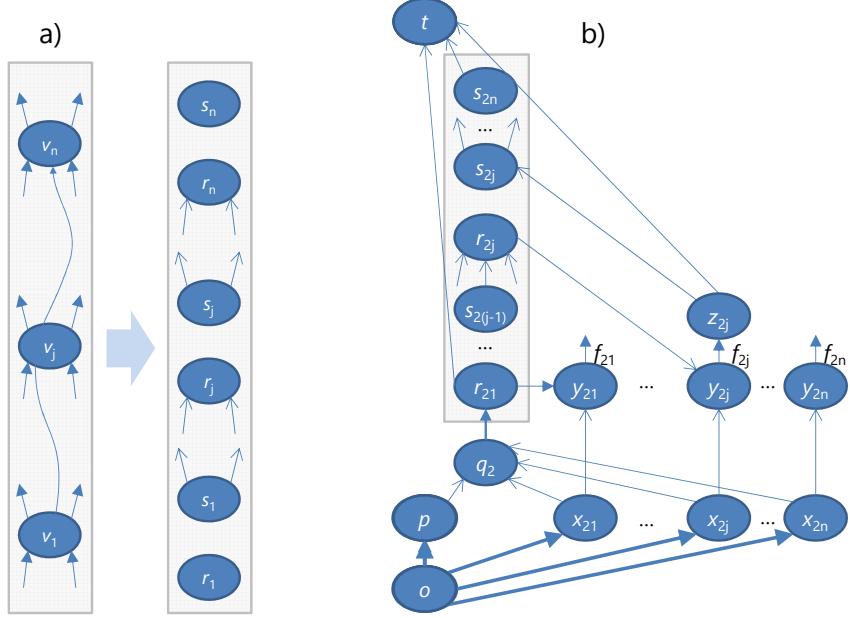


Fig. 2. HP to RR problem reduction

tonian path exists, if and only if, the cost of an optimal solution of the RR problem instance is equal to $M + \frac{2}{2n+1} + \frac{3n}{2n+1}\varepsilon$. Following the analysis presented above, the cost of an optimal solution of that RR problem instance is equal to $M + \frac{2}{2n+1} + \frac{3n}{2n+1}\varepsilon$, if and only if, there exists a path from q_1 to t that traverses all links (y_{1j}, z_{1j}) , $j = 1, 2, \dots, n$, and a path from q_2 to t that traverses all links (y_{2j}, z_{2j}) , $j = 1, 2, \dots, n$; the order of link traversal is irrelevant, however, each of the links has to be traversed exactly once, otherwise the last element of the cost expression will be greater than ε . And, by construction, such paths exist, if and only if, there exists a Hamiltonian path in graph \mathcal{G} .

■

3 Conclusion

Showing the transformation from the Hamiltonian path problem, it is proved that the problem of state-dependent restricted restoration design of resilient multi-commodity flow networks is \mathcal{NP} -hard even for a single commodity and for single-element failures. That way, finally, for every combination of choices of the network protection method and the resilient network design options either the corresponding design problem is known to be \mathcal{NP} -hard or a compact linear programming formulation is available.

References

- [1] M.R. Garey and D.R. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W.H. Freeman & Co, New York, 1979.
- [2] J-F. Maurras and S. Vanier. Network synthesis under survivability constraints. *4OR – A Quarterly Journal of Operations Research*, 2(1):53–67, 2004.
- [3] D. Nace, M. Pióro, A. Tomaszewski, and M. Żotkiewicz. A polynomial multicommodity flow problem with difficult path generation. In *Proceedings of the 3rd International Workshop on Reliable Networks Design and Modeling*, pages 125–129, Budapest, 2011.
- [4] D. Nace, M. Pióro, A. Tomaszewski, and M. Żotkiewicz. Complexity issues for a multicommodity flow problem with difficult dual separation. *Networks: an International Journal*, to appear in 2012.
- [5] S. Orlowski. Local and global restoration of node and link failures in telecommunication networks. Master’s thesis, Technische Universität Berlin, 2003.
- [6] S. Orlowski. *Optimal Design of Survivable Multi-layer Telecommunication Networks*. PhD thesis, Technische Universität Berlin, 2009.
- [7] S. Orlowski and M. Pióro. On the complexity of column generation in network design with path-based survivability mechanisms. *Networks: an International Journal*, to appear in 2011.
- [8] M. Pióro and D. Medhi. *Routing, Flow, and Capacity Design in Communication and Computer Networks*. Morgan-Kaufmann Publishers, San Francisco, CA, USA, 2004.
- [9] M. Pióro, D. Nace, A. Tomaszewski, and M. Żotkiewicz. On a survivable network design problem with one or two failing links and elementary paths. In *Proceedings of the 21st International Symposium on Mathematical Programming*, Berlin, 2012.
- [10] A. Tomaszewski, M. Pióro, and M. Żotkiewicz. On the complexity of resilient network design. *Networks: an International Journal*, 55(2):109–118, 2010.
- [11] R. Wessäly. *Dimensioning Survivable Capacitated NETworks*. PhD thesis, Technische Universität Berlin, 2000.
- [12] M. Żotkiewicz, M. Pióro, and A. Tomaszewski. Complexity of resilient network optimization. *European Transactions on Telecommunications*, 20(7):701–709, 2009.

IP modeling of the survivable hop constrained connected facility location problem

A. Bley^{a,1}, S. M. Hashemi^b, and M. Rezapour^{a,2}

^a Institute for Mathematics, TU Berlin, Germany

^b Department of Computer Science, Amirkabir University of Technology, Iran

Abstract

We consider a generalized version of the rooted connected facility location problem which occurs in planning of telecommunication networks with both survivability and hop-length constraints. Given a set of client nodes, a set of potential facility nodes including one predetermined root facility, a set of optional Steiner nodes, and the set of the potential connections among these nodes, that task is to decide which facilities to open, how to assign the clients to the open facilities, and how to interconnect the open facilities in such a way, that the resulting network contains at least λ edge-disjoint paths, each containing at most H edges, between the root and each open facility and that the total cost for opening facilities and installing connections is minimal. We study two IP models for this problem and present a branch-and-cut algorithm based on Benders decomposition for finding its solution. Finally, we report computational results.

Keywords: Connected facility location, survivability, integer programming.

1 Introduction

A typical metropolitan telecommunication network consists of several local access networks, that are connected by a (regional) core network to a central

¹ Supported by the DFG research center MATHEON–‘Mathematics for key technologies’.

² Supported by the DFG research training group ‘Methods for Discrete Structures’.

³ Emails: {bley, rezapour}@math.tu-berlin.de and hashemi@aut.ac.ir.

hub node, that provides connectivity to the national or international backbone. The traffic originating at the clients is sent through the access networks to the (regional) core nodes. From there, it traverses the core network(s) to reach the national core or the access network of its destination. Routing functionalities are typically only available at the regional or central core nodes. As the core networks play a primary role for the service availability and the service quality in such networks, it is common to require that these networks are fault-tolerant and lead to short routing paths. To model such a network planning scenario, we introduce a generalized version of the rooted connected facility location problem considering both survivability and hop-length constraints.

In the *Survivable Hop-constrained Connected Facility Location problem (SHConFL)*, we are given an undirected graph $G = (V, E)$ with $V = R \cup S$, where R is the set of clients and S is the set of (potential) core nodes. The set $F \subseteq S$ is the set of potential facilities and $r \in S \setminus F$ is the root facility opened in advance, which corresponds to the central hub node. Only edges in $E(S) := \{uv \in E : u, v \in S\}$ may be used in the (regional) core. Furthermore, we are given the costs $f_i \in \mathbb{Z}_{\geq 0}$ for opening facility $i \in F$, the costs $a_{ij} \in \mathbb{Z}_{\geq 0}$ for assigning customer $j \in R$ to facility $i \in F \cup \{r\}$, and the costs $c_e \in \mathbb{Z}_{\geq 0}$ for installing edge $e \in E(S)$ in the (regional) core. Finally, we have an integer hop limit $H \geq 1$, and an integer connectivity requirement $\lambda \geq 1$. We seek for a subset $I \subseteq F$ of facilities to open, a function $\sigma(j) : R \rightarrow I \cup \{r\}$ assigning the clients to the open facilities, and an edge set $E' \subseteq E(S)$ such that (S, E') contains, for each facility $i \in I$, at least λ edge-disjoint (r, i) -paths of length at most H . The objective is to minimize the total cost $\sum_{i \in I} f_i + \sum_{j \in R} a_{\sigma(j)j} + \sum_{e \in E'} c_e$.

The SHConFL has not yet been studied in the literature. However, it is related to two other problems, namely the *Hop-Constrained Connected Facility Location (HConFL)* [4] and the *Hop-Constrained Survivable Network Design Problem (HSND)* [3,5]. The HConFL problem is a special case of the SHConFL with $\lambda = 1$. An overview of formulations and polyhedral results for HConFL can be found in [4]. The HSND problem also can be viewed as a special case of the problem where the set of facilities given in advance. Models for HSND can be found in [3,5].

In this paper we present two strong extended formulations for the SHConFL, inspired by the known formulations for HSND, and present computational results obtained with our implementation of a branch-and-cut algorithm. To make the stronger model computationally tractable, we applied Benders decomposition approach, projecting out the extended flow variables. To speed up the algorithm, we primarily separate connectivity inequalities stemming from (and valid for) the corresponding problem with connectivity constraints but without hop constraints. We only solve the computationally expensive separation problem for Benders cuts if the (computationally much easier) separation of connectivity inequalities fails.

2 IP Formulations

2.1 Multi-Commodity Flow Based Formulation.

For each facility $i \in F$, we create a directed layered graph $G^i = (S^i, A^i)$ with

$$S^i = \{r_0\} \cup \{u_h : u \in S, 1 \leq h < H, u \neq r\} \cup \{i_H\} \text{ and}$$

$$A^i = \{(u_{h-1}, v_h) : u_{h-1}, v_h \in S^i, uv \in E(S), 1 \leq h \leq H\}.$$

The copy of node u in layer h of G^i is denoted by u_h and each edge uv has two directed copies (u_{h-1}, v_h) and (v_{h-1}, u_h) between layers $h-1$ and h of G^i .

Clearly, any (r, i) path of length $h \leq H$ in the (core) graph corresponds to a directed path of the same length from r_0 to i_h in G^i .

We introduce binary variables x_{ij} , y_i , and z_{uv} indicating whether customer j is assigned to facility i , facility i is opened, and edge $uv \in E(S)$ is selected, respectively. Additional binary variables f_{uv}^{ih} indicate if a path from r to facility i contains arc (u, v) in position h . Using these variables, a compact flow-based formulation for the problem is obtained as follows:

$$(MCF) \quad \min \sum_{i \in F} f_i y_i + \sum_{j \in R} \sum_{i \in F \cup \{r\}} a_{ij} x_{ij} + \sum_{e \in E(S)} c_e z_e$$

$$\sum_{i \in F \cup \{r\}} x_{ij} = 1 \quad \forall j \in R \quad (1)$$

$$y_i - x_{ij} \geq 0 \quad \forall i \in F, j \in R \quad (2)$$

$$\sum_{(v_{h-1}, u_h) \in A^i} f_{vu}^{i(h-1)} - \sum_{(u_h, v_{h+1}) \in A^i} f_{uv}^{ih} = 0 \quad \forall i \in F, u_h \in S^i; u \notin \{r, i\} \quad (3)$$

$$\sum_{h=0}^{H-1} \sum_{(u_h, i_{h+1}) \in A^i} f_{ui}^{ih} = \lambda y_i \quad \forall i \in F \quad (4)$$

$$f_{rv}^{i0} \leq z_{rv} \quad \forall i \in F, rv \in E(S) \quad (5)$$

$$\sum_{h=1}^{H-2} (f_{uv}^{ih} + f_{vu}^{ih}) \leq z_{uv} \quad \forall i \in F, uv \in E(S), u, v \notin \{r, i\} \quad (6)$$

$$\sum_{h=1}^{H-1} f_{ui}^{ih} \leq z_{ui} \quad \forall i \in F, ui \in E(S), u \neq r \quad (7)$$

$$x, y, z, f \in \{0, 1\} \quad (8)$$

Constraints (1) and (2) state that any customer has to be assigned to an open facility. Constraints (3) and (4) ensure flow conservation in each layered graph G^i , which guarantees λ length bounded flow paths between r and each open facility. Constraints (5)-(7) finally guarantee the edge-disjointness of these paths and the correct setting of the edge variables z_e .

2.2 Hop-Level Multi-Commodity Flow Based Formulation.

Recently, Mahjoub et al. [5] introduced a new idea to extend the general hop constrained formulation using additional variables indicating the distance of each core node from r in the solution. The formulation presented above can be improved by applying this idea. Given a solution (I, σ, E') , one can partition S into $H+2$ levels according to their distance from r : Level 0 only contains r ; level l , $1 \leq l \leq H$, contains nodes with distance l from r ; and level $H+1$ contains the nodes that are not connected to r in E' . We introduce binary variables w_u^l indicating if vertex u is in level l (according to its distance from r in the solution) and a_{uv}^{lk} indicating if edge uv belongs to the solution with u in level l and v in level k , respectively. Remark that $|k-l| \leq 1$. Together with the variables x, y , and z , we obtain a new formulation HL-MCF as follows.

Let $E'_S = E(S) \setminus \delta(r)$. With the following constraints, any binary vector (y, z) defines binary values (w, a) where each node is assigned to a single level.

$$\sum_{l=1}^{H+1} w_u^l = 1 \quad \forall u \in S \setminus \{r\} \quad (9)$$

$$w_i^{H+1} \leq 1 - y_i \quad \forall i \in F \setminus \{r\} \quad (10)$$

$$\sum_{l=1}^{H-1} a_{uv}^{ll} + \sum_{l=1}^{H-1} (a_{uv}^{l(l+1)} + a_{vu}^{l(l+1)}) = z_{uv} \quad \forall uv \in E'_S \quad (11)$$

$$a_{uv}^{11} + a_{uv}^{12} \leq w_u^1 \quad \forall uv \in E'_S \quad (12)$$

$$a_{uv}^{11} + a_{vu}^{12} \leq w_v^1 \quad \forall uv \in E'_S \quad (13)$$

$$a_{uv}^{ll} + a_{uv}^{l(l+1)} + a_{vu}^{l(l-1)l} \leq w_u^l \quad \forall uv \in E'_S, 2 \leq l \leq H-1 \quad (14)$$

$$a_{uv}^{ll} + a_{vu}^{l(l+1)} + a_{uv}^{l(l-1)l} \leq w_v^l \quad \forall uv \in E'_S, 2 \leq l \leq H-1 \quad (15)$$

$$a_{vu}^{(H-1)H} \leq w_u^H \quad \forall uv \in E'_S \quad (16)$$

$$a_{uv}^{(H-1)H} \leq w_v^H \quad \forall uv \in E'_S \quad (17)$$

$$w_v^l - \sum_{(u,v) \in \delta(v), u \neq r} a_{uv}^{(l-1)l} \leq 0 \quad \forall v \in S, 2 \leq l \leq H \quad (18)$$

$$w, a, y, z \in \{0, 1\} \quad (20)$$

Constraints (9)-(10) state that each node should be in exactly one of the possible levels and chosen facilities must be reachable. Constraints (11) and (12) make the connection between (w, a) variables and variables z . Constraints (13)-(18) ensure that a variable a_{uv}^{lk} can only be 1 if both w_u^l and w_v^k are one. Constraints (19) state that a node can only be in level l if it is reached by at least one edge from level $l-1$.

Like in [5], it can be shown that a fractional (w, a) splits nodes and edges into different levels, which might increase the length of paths between r and open facilities in the level-expanded network induced by (w, a) . Enforcing the

existence of hop-limited arc-disjoint paths between r and open facilities in the level-expanded network, we can thus strengthen the formulation as follows.

For each facility $i \in F$, we create a directed layered graph $G_H^i = (S_H^i, A_H^i)$, where $S_H^i = \{r_0^0\} \cup \{u_h^l : u \in S, 1 \leq l \leq h \leq H-1, u \neq r\} \cup \{i_h^l : 1 \leq l \leq H\}$ and $A_H^i = \{(u_{h-1}^l, v_h^k) : u_{h-1}^l, v_h^k \in S_H^i, uv \in E(S), 1 \leq h \leq H, |l-k| \leq 1\} \cup \{(i_h^l, i_{h+1}^l) : 1 \leq l \leq h \leq H-1\}$. The copy of node u in layer h and level l is denoted by u_h^l , while (u_{h-1}^l, v_h^k) denotes the directed arc corresponding to the copy of edge uv with node u in layer $h-1$ and level l and node v in layer h and level k . For each such arc, we have a binary flow variable g_{uv}^{ihlk} indicating that a path from r to facility i goes from u at level l to v at level k in hop h .

The following constraints ensure the existence of the λ arc-disjoint paths.

$$\sum_{(v_{h-1}^k, u_h^l) \in A_H^i} g_{vu}^{i(h-1)kl} - \sum_{(u_h^l, v_{h+1}^k) \in A_H^i} g_{uv}^{ihlk} = 0 \quad \forall i \in F, u_h^l \in S_H^i, u \neq r, h \leq H-1 \quad (21)$$

$$\sum_{l=1}^H \sum_{(v_{H-1}^k, i_H^l) \in A_H^i} g_{vi}^{i(H-1)l} = \lambda y_i \quad \forall i \in F \quad (22)$$

$$g_{rv}^{i001} \leq a_{rv}^{01} \quad \forall i \in F, rv \in E(S) \quad (23)$$

$$\sum_{h=l}^{H-2} (g_{uv}^{ihll} + g_{vu}^{ihll}) \leq a_{uv}^{ll} \quad \forall i \in F, uv \in E'_S \setminus \delta(i), 1 \leq l \leq H-2 \quad (24)$$

$$\sum_{h=l}^{H-2} g_{uv}^{ihl(l+1)} + \sum_{h=l+1}^{H-2} g_{vu}^{ih(l+1)l} \leq a_{uv}^{l(l+1)} \quad \forall i \in F, uv \in E'_S \setminus \delta(i), 1 \leq l \leq H-2 \quad (25)$$

$$\sum_{h=l}^{H-1} g_{ui}^{ihll} \leq a_{ui}^{ll} \quad \forall i \in F, ui \in \delta(i) \setminus \delta(r), 1 \leq l \leq H-1 \quad (26)$$

$$\sum_{h=l}^{H-1} g_{ui}^{ihl(l+1)} \leq a_{ui}^{l(l+1)} \quad \forall i \in F, ui \in \delta(i) \setminus \delta(r), 1 \leq l \leq H-1 \quad (27)$$

Constraints (21)-(22) are the flow conservation constraints at every node of the layer- and level-extended graphs, guaranteeing λ units of flow from r to each open facility. Constraints (23)-(27) link the flow to the a variables.

The complete HL-MCF formulation is given by the objective function (MCF) subject to constraints (1)-(2) and (8)-(27).

Lemma 2.1 *Formulation HL-MCF is at least as strong as formulation MCF.*

3 Benders Decomposition for HL-MCF

In our implementation, we use a Benders decomposition approach to efficiently handle the huge number of variables and constraints of HL-MCF. However, we cannot directly apply Benders decomposition method, because all variables of

HL-MCF are integer and classical duality theory does not allow to project out integer variables. Similar to Botton et al. [3], we therefor relax the integrality restrictions of the flow variables in HL-MCF and apply Benders decomposition to this relaxation, called R-HL-MCF. We shall discuss whether R-HL-MCF provides the same optimal design variables x , y , and z as HL-MCF.

The master problem is given by the objective function (HL-MCF) subject to constraints (1)-(2), and (8)-(20). A solution $\bar{y}, \bar{x}, \bar{z}, \bar{w}, \bar{a}$ of the master problem defines a feasible solution for the R-HL-MCF if and only if for each $i \in F$ there exist fractional flow values satisfying (21)-(27) with $y_i = \bar{y}_i$ and $a = \bar{a}$.

To apply Farkas lemma to this linear system called SUB i , we define dual variables $\pi_{u,l}^h, \pi, \sigma_{rv}^{0(1)}, \sigma_{uv}^{l(l)}, \sigma_{uv}^{l(l+1)}, \sigma_{ui}^{l(l)}, \sigma_{ui}^{l(l+1)}$ associated to the constraints (21)-(27), respectively. Let $\Pi(i)$ be the set of extreme rays of the corresponding dual system. It can be shown that the Benders reformulation of R-HL-MCF is given by adding the following Benders cuts to the master problem:

$$\lambda y_i \bar{\pi} - \sum_{(u,v) \in E(S), 0 \leq l, k \leq H-1} a_{uv}^{lk} \bar{\sigma}_{uv}^{l(k)} \leq 0 \quad \forall (\bar{\pi}, \bar{\sigma}) \in \cup_{i \in F} \Pi(i) . \quad (28)$$

Let ξ_{int} and ξ_{frac} denote the set of all binary vectors $(\bar{x}, \bar{y}, \bar{z})$ for which there exists an integral or a fractional solution for (SUB i), $\forall i \in F$, respectively. Since $\xi_{int} \subseteq \xi_{frac}$, any Benders cut (28) is a valid inequality for $\text{conv}(\xi_{int})$, as well. Following Lemma shows that these Benders cuts are sufficient to describe ξ_{int} in some special cases, but not in general.

Lemma 3.1 ξ_{int} and ξ_{frac} are equal for $H=2, 3$ with any λ , and $H=4$ with $\lambda=2$. For $H \geq 4$, there exist a $\lambda \geq 2$ for which $\xi_{int} \subsetneq \xi_{frac}$, unless P=NP.

The results follow immediately from the corresponding results for the HSND problem given by Botton et al. [3] (see also [2]).

For general λ and H , we retreat to the generation of *Benders feasibility cuts* to cut off infeasible integer solutions $(\bar{x}, \bar{y}, \bar{z}) \in \xi_{frac} \setminus \xi_{int}$. For this, we solve an integer programming formulation to check if there exist integral flows satisfying (21)-(27) with $y = \bar{y}$ and $a = \bar{a}$; see [3] for more details.

4 Branch-and-cut algorithm

We implemented a branch-and-cut algorithm based on the Benders decomposition discussed above. In every node of the branch-and-bound tree, we first separate (non-hop-restricted) connectivity inequalities of the form $z(\delta(U)) \geq \lambda y_i$ with $U \subseteq S$, $r \notin U$, $i \in U$ for some facility $i \in F$ via max-flow min-cut computations. Only if no violated connectivity cuts are found, we check for violated Benders cuts, which is computationally far more expensive. Formally, in every node of the (B&B) tree the algorithm works as follows:

- (i) Solve the master problem; take the optimal (fractional) solution

- (ii) Check if the current solution satisfies the connectivity requirements between root and facilities. As long as there are violated connectivity cuts, add them to the master and goto (i).
- (iii) Solve the linear Benders subsystems (SUB^i) for the current solution. If this results in a violated Benders cut, add it to the master and goto (i).
- (iv) Only at integer nodes of tree (and if needed; see Lemma 3.1): Solve the integer Benders subsystems (SUB^i) for the current solution. If infeasible, add corresponding feasibility cut to the master and goto (i).

Our computational experiences (see Section 5) show that connectivity cuts generated early in step (ii) are very important to avoid the exploration of too many infeasible nodes and to reduce the time spent in the computationally expensive generation of Benders cuts.

5 Computational Study

The algorithm has been implemented in C++, using SCIP [1] as a framework and CPLEX 12.4 as a LP solver, and run on a machine with a AMD Phenom(tm) II X6 1090T 3GHz and 8 GiB RAM. To generate our instances, we combine benchmarks from the HSND problem and benchmarks from uncapacitated facility location problem (UFL) for the core and access part of our instances, respectively. We follow Botton et al. [3] and generate the core graph as follows. Instances are complete graphs whose node set is of size 60 randomly placed among integer points of a grid 100×100 . The first $|F| \in \{20, 30, 40\}$ nodes are selected to be potential facilities and the node with index 1 is selected as the root node. The edge costs are set to be the Euclidean distance between any two points. Now given an instance of UFL, $|F| \in \{20, 30, 40\}$ facilities are randomly selected. The number of customers, facility opening costs and assignment costs are provided in UFL instance. We consider a set of 6 instances obtained by combining two UFL instances $mp1$ and $mq1$ ⁴ (of size 200×200 and 300×300 , respectively) with the three generated core instances.

Table 1 reports number of cuts of types “connectivity cut” and “Benders cut” generated throughout the execution of the algorithm (no feasibility cuts have been generated in any instance), denoted by “C(C)”, and “C(B)”, respectively, the number of branch and bound nodes visited, denoted by “B&B”, the total time to solve the instances in seconds, denoted by “T(s)”. Results in Table 1 show that easily computable connectivity cuts are very important to avoid generating many expensive Benders cuts.

For the algorithm without Step (ii), Table 2 reports the number of generated Benders cuts, the number of “B&B” nodes, and the running times. The results in Table 2 (compare to same results in Table 1) show the practical

⁴ Available at <http://www.mpi-inf.mpg.de/departments/d1/projects/benchmarks/UflLib>.

Instances	H	$\lambda = 3$				$\lambda = 5$			
		C(C)	C(B)	B&B	T(s)	C(C)	C(B)	B&B	T(s)
MP1-20	3	148	7	54	84	286	9	56	209
MP1-20	5	199	19	101	478	231	2	32	448
MP1-30	3	67	0	5	25	40	1	1	17
MP1-30	5	69	0	7	78	55	4	1	39
MP1-40	3	575	14	136	457	1483	374	620	950
MP1-40	5	301	69	215	819	379	186	528	1092
MQ1-20	3	73	2	29	83	99	0	14	57
MQ1-20	5	91	0	33	163	95	2	8	198
MQ1-30	3	231	16	87	210	159	31	54	601
MQ1-30	5	146	50	92	476	120	27	91	969
MQ1-40	3	986	458	679	568	813	5	352	683
MQ1-40	5	440	268	935	1426	560	16	241	980

Table 1
Results of our algorithm on the set of instances

improvement of the algorithm due to the addition of cuts stemming from the plain (non-length-restricted) connectivity problem.

Instances	$H = 3, \lambda = 3$			$H = 3, \lambda = 5$		
	Cut	B&B	T(s)	Cut	B&B	T(s)
MP1-20	654	216	283	642	107	366
MP1-30	144	9	93	58	12	47
MQ1-20	178	6	163	185	4	194
MQ1-30	466	111	385	454	67	872

Table 2
Results of our algorithm without Step (ii) on a subset of the instances

References

- [1] Achterberg, T. Constraint Integer Programming. PhD Thesis, TU Berlin, 2007.
- [2] Bley, A., J. Neto. Approximability of 3- and 4-Hop Bounded Disjoint Paths Problems. In Proc. of IPCO 2010, pages 205-218.
- [3] Botton,Q., B. Fortz, L. Gouveia, M. Poss. Benders Decomposition for the Hop-Constrained Survivable Network Design Problem. INFORMS Journal on Computing 25, pages 13-26, 2013.
- [4] Ljubic, I., S. Gollowitzer. Layered Graph Approaches to the Hop Constrained Connected Facility Location Problem, INFORMS Journal on Computing, DOI: 10.1287/ijoc.1120.0500, 2012
- [5] Mahjoub, R., L. Simonetti, E. Uchoa. Hop-Level Flow Formulation for the Hop Constrained Survivable Network Design Problem. Networks 61, pages 171-179, 2013.

A matheuristic approach for solving a home health care problem

Hanane Allaoua¹ Sylvie Borne¹ Lucas Létocart¹ and
Roberto Wolfler Calvo¹

*Université Paris 13, Sorbonne Paris Cité, LIPN, CNRS, (UMR 7030)
93430 Villetaneuse, France*

Abstract

We deal with a Home Health Care Problem (HHCP) which objective consists in constructing the optimal routes and rosters for the health care staffs. The challenge lies in combining aspects of vehicle routing and staff rostering which are two well known hard combinatorial optimization problems. To solve this problem, we initially propose an integer linear programming formulation (ILP) and we tested this model on small instances. To deal with larger instances we develop a matheuristic based on the decomposition of the ILP formulation into two problems. The first one is a set partitioning like problem and it represents the rostering part. The second problem consists in the routing part. This latter is equivalent to a Multi-depot Traveling Salesman Problem with Time Windows (MTSPTW).

Keywords: Home health care, rostering, vehicle routing with time windows.

¹ Email: (allaoua, borne, letocart, wolfler)@lipn.univ-paris13.fr

1 Introduction

Home health care (HHC), i.e., visiting and nursing patients at home, is a growing sector in the medical care system. Therefore, the optimal scheduling of the health care staffs arises. The objective of this problem consists in constructing routes and rosters for the health care staffs while optimizing costs. The challenge lies in combining aspects of vehicle routing and staff rostering which are two well known hard combinatorial optimization problems. The routing part of HHCP can be modeled as a Vehicle Routing Problem with Time Windows (VRPTW) with multiple depots and some specific constraints. It has multiple depots since each staff's home acts as a depot housing a single vehicle where a route must start and end. Several references dealing with the routing and rostering of the HHCP exist in the literature, we report here only two references which are closer to our approach. In [1] the authors use a combination of linear programming, constraint programming and (meta-)heuristic. In [5] the authors model the problem as a set partitioning problem with side constraints and develop a branch-and-price algorithm. For papers dealing with only one of the two aspects of HHCP see for example [2,3]. Last, but not least, our problem is similar to the technician routing and scheduling problem studied in [4].

2 Problem and proposed method

We consider a set of patients needing health cares at home, and each patient may require several cares. Therefore, we consider a set of services. Each one composed by a patient, a skill, the duration of the care and a time window during which the service to perform must begin. Moreover, a set of health care staffs is given. Each staff is characterized by a skill and a time window representing when he/she is available. The set of services must be assigned to the staffs, we also assign staffs to shifts. This latter is characterized by a time window. All staffs which are assigned to the shift must respect the time window. Specific constraints regarding the assignment of staffs to shifts according to their types (for example: a set of skills needed, overnight and week-end jobs, split shifts) could be considered.

2.1 Problem formulation

Let S be the set of services, P be the set of care staffs, and J be the set of shifts. The problem is defined on a graph $G = (V, E)$ where $V = P \cup S \cup J$ and the edge set $E = \{(i, j) : i \in V, j \in V, i \neq j\}$. Each edge $(i, j), i, j \in V \setminus \{J\}$

has a cost d_{ij} which is the distance between i and j . D_i is the duration of service i , Q_i and $[e_i, l_i]$ are respectively the skill and the time window of node $i \in V$. y_i is equal 1 if staff i is in the solution and 0 otherwise and $v_{i,j}$ (resp $w_{i,j}$) is equal 1 if staff i (resp staff or service i) is assigned to service (resp shift) j , 0 otherwise. The variable $\xi_{i,k,h}$ is equal 1 if staff i do service k before service h , 0 otherwise and $p_{i,k}$ is a variable representing the arrival time of staff i in service k . The objective function of our formulation given as below is to minimize the number of staffs.

$$\begin{aligned}
& \min \sum_{i=1}^{|P|} y_i \\
& \sum_{i=1}^{|P|} v_{i,k} = 1 && \forall k \in S \\
& \sum_{j=1}^{|J|} w_{j,k} = 1 && \forall k \in S \\
& v_{i,|S|+i} = y_i && \forall i \in P \\
& v_{i,|S|+|P|+i} = y_i && \forall i \in P \\
& \sum_{j=1}^{|J|} w_{j,|S|+i} = y_i && \forall i \in P \\
& \sum_{j=1}^{|J|} w_{j,|S|+|P|+i} = y_i && \forall i \in P \\
& v_{i,k} \leq y_i && \forall i \in P, \forall k \in S \\
& \sum_{h=1}^{|S|} \xi_{i,|S|+i,h} = y_i && \forall i \in P \\
& \sum_{h=1}^{|S|} \xi_{i,h,|S|+|P|+i} = y_i && \forall i \in P \\
& \sum_{k=1, h \neq k}^{|S|} \xi_{i,k,h} + \xi_{i,|S|+i,h} = v_{i,h} && \forall i \in P, \forall h \in S \\
& \sum_{h=1, h \neq k}^{|S|} \xi_{i,k,h} + \xi_{i,k,|S|+|P|+i} = v_{i,k} && \forall i \in P, \forall k \in S \\
& v_{i,k} e_i \leq p_{i,k} \leq v_{i,k}(l_i - D_k) && \forall i \in P, \forall k \in \{1, \dots, |S| + 2|P|\}
\end{aligned}$$

$$\begin{aligned}
\sum_{i=1}^{|P|} v_{i,k} e_k &\leq \sum_{i=1}^{|P|} p_{i,k} \leq \sum_{i=1}^{|P|} v_{i,k} l_k & \forall k \in \{1, \dots, |S| + 2|P|\} \\
\sum_{i=1}^{|P|} p_{i,k} - e_j &\geq (w_{j,k} - 1)M & \forall j \in J, k \in \{1, \dots, |S| + 2|P|\} \\
\sum_{i=1}^{|P|} p_{i,k} + D_k - l_j &\leq (1 - w_{j,k})M & \forall j \in J, k \in \{1, \dots, |S| + 2|P|\} \\
v_{i,k}(Qp_i - Qs_k) &= 0 & \forall i \in P, \forall k \in S \\
w_{j,k}(Qpo_j - Qs_k) &= 0 & \forall j \in J, \forall k \in S \\
p_{i,h} &\geq p_{i,k} + D_k + d_{kh} + (\xi_{i,k,h} - 1)M & \forall i \in P, k, h \in \{1, \dots, |S| + 2|P|\} \\
(1 - v_{i,k}) + w_{j,k} &\geq w_{j,|S|+i} & \forall i \in P, \forall j \in J, \forall k \in S \\
(v_{i,k} - 1) + w_{j,k} &\leq w_{j,|S|+i} & \forall i \in P, \forall j \in J, \forall k \in S \\
w_{j,|S|+i} &= w_{j,|S|+|P|+i} & \forall i \in P, \forall j \in J
\end{aligned}$$

We solved this model with Cplex-IBM which is able to solve small/medium instances.

2.2 Matheuristic

To deal with larger instances we develop a matheuristic based on a decomposition of the previous formulation into two problems. The first one is modeled as a set partitioning-like problem and it represents the rostering part. The second problem consists in the routing part which is equivalent to a Multi-Depot Traveling Salesman Problem with Time Windows (MTSPTW). The algorithm works as follows: first we perform the pre-assignment of staffs to shifts and we cluster the set of services using a first method. Then, we solve the TSPTW using an exact method for each cluster. The result is a set of feasible routes used as columns of the rostering problem. If there is at least one uncovered service, we add a new set of clusters (i.e., columns) by using a second method and we obtain a new set of routes thanks to the solution of TSPTW for each new cluster obtained. Then, we solve the rostering problem by using the whole set of routes generated. The remaining unserved services are inserted one by one in the routes.

2.2.1 Pre-assignment

To perform the assignment of staffs to shifts, we consider a new set of staffs, noted P' , as each staff i in P is replaced in P' by new staffs with different

time windows according with time windows of shifts. For example we consider staff i in P which has time window equals $[8, 20]$. Let $[7, 12]$ and $[13, 19]$ be respectively the time windows of the morning shift and the afternoon shift. In P' , i is represented by two staffs i' and i'' which have time windows equal to $[8, 12]$ and $[13, 19]$. Then if staff i' is in the solution, then staff i is assigned to the morning shift no overlap shifts are considered.

2.2.2 Clusters of services

The purpose of this step is to define subsets $S^1, S^2, S^3, \dots \subseteq S$ to get feasible tours when we solve a TSPTW having as input $S^i \cup \{k\}, k \in P'$. Two methods are proposed.

First method

This method is inspired by the m-max-cut. Let S_l be a subset of services which have the same skill l . We define an oriented graph $G = (S_l, A)$, where A is the set of arcs defined as $A = \{(i, j) : i \in S_l, j \in S_l, i \neq j\}$. Each arc (i, j) has a cost P_{ij} which is the overlap between the two time windows of i and j . Let $(i', j') \in A$ be an arc such as $P_{i'j'} = \min(P_{ij}, i \in S_l, j \in S_l)$. To cluster the subset S_l we proceed to the coloring of nodes such as, colors represent different clusters. The coloring of i' and j' is as follows:

- if both i' and j' are not colored. We color both i' and j' with a new color.
- if i' or j' is already colored. We color the non colored node with the same color of the other.

Then we collapse nodes i' and j' into one node. We replace all parallel arcs by one arc with cost equals to the sum of costs of those arcs. The heuristic terminates when all the nodes have been colored.

Second method

This method alternates between computing a lower bound of a bin-packing problem and solving an assignment problem.

Valid lower bounds. Since we want to minimize the number of staffs used for each skill, we can easily define a lower bound. Note that for each skill, the solution of a bin-packing problem, defined as follows, is a lower bound for our problem. Each service could be seen as an object to pack whose volume is $D_i + \min_{j \in S_l} (d_{i,j})$ and the capacity of the bins are the time windows of health

care staffs. To compute this lower bound we use the following formula:

$$lb1_l = \left\lceil \frac{\sum_{i \in S_l} (D_i + \min_{j \in S_l} (d_{i,j}))}{l_k - e_k - 2 \times \min_{j \in S_l} (d_{k,j})} \right\rceil$$

Note that k is the staff which has the skill l and the largest time window.

To improve this lower bound, we relaxed the time windows and precedence constraints of the MTSPTW and we partially add them in the objective function. Therefore we solve an assignment problem (see [7]) on a new adapted graph. Let I be a set of care staffs which skills equal l with $|I| = lb1_l$, and $S'' = S_l \cup I$. Therefore, the new graph is $G = (S'', A)$ where the arc set is $A = \{(i, j) : i \in S'', j \in S'', i \neq j\}$ and $W_{ij} = \max(e_j - d_{ij} - D_i - e_i, 0)$ is the waiting time between i and j . The assignment problem we solve is defined as follows:

$$(P) \quad \begin{aligned} & \min \sum_{i,j \in S''} (D_i + d_{ij} + W_{ij}) x_{ij} \\ & \text{s.t. } \sum_{i \in S''} x_{ij} = 1 \quad \forall j \in S'' \\ & \quad \sum_{j \in S''} x_{ij} = 1 \quad \forall i \in S'' \\ & \quad x_{ij} \in \{0, 1\} \quad \forall i, j \in S'' \end{aligned}$$

Given the optimal solution X^* of (P) we compute $lb2_l$ by using the following formula:

$$lb2_l = \left\lceil \frac{\sum_{i,j \in S''} (D_i + x_{ij}^* \times d_{ij})}{l_k - e_k} \right\rceil.$$

$lb2_l$ is an improvement of $lb1_l$. Then we solve the assignment problem for the second time with $|I| = lb2_l$. The solution of this second assignment problem can be decomposed in clusters of services S^i and subtours. Therefore we solve a TSPTW for each cluster and we insert subtours (i.e., the unserved customers) into clusters.

2.2.3 Building tours (TSPTW)

Let $S^k, k \in P'$ be a subset of services which can be served by the staff k (thanks to the solution obtained in Section 2.2.2) and $S' = S^k \cup \{k\}, k \in P'$.

We define the TSPTW on an oriented graph $G = (S', A)$, where the arc set is $A = \{(i, j) : i \in S', j \in S', i \neq j\}$. We use the dynamic programming method proposed in [6] to solve the TSPTW for each cluster generated.

2.2.4 Planning (set partitioning like problem)

Let R be a subset of feasible routes, R_i represents the set of routes passing through service i , R_k represents the set of routes passing through staff k . Note that we allow empty routes for each staff. The variable y_l is equal 1 if route l is in the solution, 0 otherwise. Then we can write the following formulation:

$$\begin{aligned} \min \quad & \sum_{l \in R} y_l \\ \text{s.t.} \quad & \sum_{l \in R_i} y_l = 1 \quad \forall i \in S \\ & \sum_{l \in R_k} y_l \leq 1 \quad \forall k \in P \\ & y_l \in \{0, 1\} \quad \forall l \in R \end{aligned}$$

That we solve with Cplex IBM solver.

3 Computational results

The various algorithms were coded in C++ and Cplex IBM solver on a machine having the following characteristics: Bi Xeon quad core 2,8 Ghz - 16 Go RAM - 8 coeurs machine. We tested the algorithms on 190 instances proposed in [3]. All instances have 30 services, number of staffs between 7 and 9 and number of skills equal to 3. The rows of Table 1 report the results obtained by the different algorithms used i.e., opt (ILP,Cplex), $lb1 = \sum_{l \in Q} lb1_l$ (bin-packing), $lb2 = \sum_{l \in Q} lb2_l$ where Q is the set of skills (assignment + bin-packing) and ub (matheuristic). The results are between 3 and 9 staffs. Each cell in the N-staffs= n ($n \in \{3, \dots, 9\}$) column represents the number of instances that have solution equal to n , for each one of the four algorithms. Note that none of the instances has $opt = 3$ while 53 instances have $lb1 = 3$. We also give the average values and standard deviation for what concerns computational time and gap for each one of the algorithms. Note that the gap of the exact method is given in relation to the solution value at the root node. In 50.52% of instances the solution found by the matheuristic has the same value of the one found with the exact method; in 35.26% of the cases, the former solution value exceeds the latter by one unit, while in the remaining 14.22% by two.

	N-staffs									Time(sec)		Gap(%)	
	3	4	5	6	7	8	9	avg	sd	avg	sd	avg	sd
<i>opt</i>	0	3	59	104	22	1	1	3383	18927	93.33	23.98		
<i>lb1</i>	53	112	23	2	0	0	0	0.001	0.0008	54.18	31.19		
<i>lb2</i>	7	86	75	22	0	0	0	0.018	0.013	29.16	24.29		
<i>ub</i>	0	1	33	68	61	22	5	0.9	1.1	9.05	10		

Table 1
Computational results

References

- [1] Bertels, S., and T. Fahle. *A hybrid setup for a hybrid scenario: combining heuristics for the home health care problem*, Computers & Operations Research. **33**(10) (2006), 2866–2890.
- [2] Eveborn, P., P. Flisberg, and M. Ronnqvist. *Laps carean operational system for staff planning of home care*, European Journal of Operational Research. **171**(3) (2006), 962–976.
- [3] Kergosien, Y., C. Lent, and J. C. Billaut. *Home health care problem: An extended multiple traveling salesman problem*, 4th Multidisciplinary International Conference on Scheduling: Theory and Applications (MISTA09), Dublin (Irlande), (2009), 85–92.
- [4] Pillac, V., C. Guéret and A.L. Medaglia. *A parallel matheuristic for the technician routing and scheduling problem*, Optimization Letters, (2012), 1–11.
- [5] Rasmussen, M.S, T. Justesen, A. Dohn, and J. Larsen. *The home care crew scheduling problem: Preference-based visit clustering and temporal dependencies*, European Journal of Operational Research, **219**(3) (2011), 598–610.
- [6] Righini, G., M. Salani. *Decremental state space relaxation strategies and initialization heuristics for solving the orienteering problem with time windows with dynamic programming*, Computers and Operations Research, **36** (2009), 1191–1203.
- [7] Wolfler Calvo, R., *A New Heuristic for the Traveling Salesman Problem with Time Windows*, Transportation Science. **34** (2000), 113–124.

Capacitated Network Design using Bin-Packing

Amal Benhamiche, A. Ridha Mahjoub¹

LAMSADE, Université Paris-Dauphine, Paris, France.

Nancy Perrot²

Orange Labs, Issy-Les-Moulineaux, France.

Eduardo Uchoa³

Universidade Federal Fluminense, Niterói, Brazil.

Abstract

In this paper, we consider the *Capacitated Network Design* (CND) problem. We investigate the relationship between CND and the *Bin-Packing* problem. This is exploited for identifying new classes of valid inequalities for the CND problem and developing a branch-and-cut algorithm to solve it efficiently.

Keywords: Capacitated Network Design, Bin-Packing, Facets, Branch-and-Cut.

1 Introduction

Given an optical network having a set of optical devices interconnected by optical fibers, and a set of traffic demands, this problem consists in finding

¹ Email: {benhamiche,mahjoub}@lamsade.dauphine.fr

² Email: nancy.perrot@orange.com

³ Email: uchoa@producao.uff.br

the number of modular capacities (modules) that have to be installed over each fiber so that the traffic is routed and the cost is minimum. In practice, several modules can be installed on one fiber. Each one can carry many traffic demands but one demand has to be routed on a unique path.

More formally, the problem can be presented as follows. Consider a bidirected graph $G = (V, A)$ that represents an optical network. Each node $v \in V$ corresponds to an optical device and each arc $a = ij \in A$ corresponds to an optical link. Let K be a set of commodities or traffic demands. Each commodity $k \in K$ has an origin node $o_k \in V$, a destination node $d_k \in V$ and a traffic D^k that has to be routed between o_k and d_k . We suppose that we can install a set of modules on each link. The set of available modules is denoted by W and a module $w \in W$ installed between nodes i and j is a copy of the arc $a = ij \in A$. Every module w can carry one or many commodities, but a commodity can not be splitted on many modules. We denote by C and c_{ij} , the capacity and the cost of each module, respectively. The CND problem consists in determining the number of modules to install on each arc of G so that the commodities can be routed and the total cost is minimum.

In this work, we mainly focus on the restriction of CND to one edge. This approach is similar to one presented by Bienstock and Muratore in [2], for survivable network design problems restricted to a cut. In [1], authors have exploited the relationship of Network Design problem with several problems studied earlier, like the binary knapsack problem.

We are interested in the relationship between CND and Bin-Packing problem [3]. Our primary motivation comes from the structure of CND whose restriction on one edge reduces to study a variant of the Bin-Packing problem. In particular, our contribution concerns this relationship and how it can be exploited to identify valid inequalities for CND problem and develop a branch-and-cut algorithm to solve it efficiently. The paper is organized as follows. In section 2, we give a compact formulation for the CND problem. We introduce an *aggregated* model for the problem in section 3, and highlight the relationship between CND and the Bin-Packing problem. We study the basic properties of the polyhedron associated with the latter problem in section 4. Finally, we show some experiments in section 5.

2 Compact formulation for CND

In this section, we give a compact flow-based formulation for the CND problem. Let y_{ij}^w be a variable that takes value 1 if the module $w \in W$ is installed on the arc $ij \in A$, and 0 otherwise. And let x_{ij}^{kw} be a variable that takes value

1 if the commodity $k \in K$ is routed on the module $w \in W$ of the arc $ij \in A$, and 0 otherwise. The CND problem is then equivalent to the following ILP:

$$\min \sum_{ij \in A} \sum_{w \in W} c_{ij} y_{ij}^w$$

$$\sum_{w \in W} \sum_{j \in V} x_{ji}^{kw} - \sum_{w \in W} \sum_{j \in V} x_{ij}^{kw} = \begin{cases} 1, & \text{if } i = d_k, \\ -1, & \text{if } i = o_k, \\ 0, & \text{otherwise,} \end{cases} \quad \begin{array}{l} \forall k \in K, \\ \forall i \in V, \end{array} \quad (1)$$

$$\sum_{k \in K} D^k x_{ij}^{kw} \leq Cy_{ij}^w, \quad \forall w \in W, \forall ij \in A, \quad (2)$$

$$x_{ij}^{kw} \in \{0, 1\}, y_{ij}^w \in \{0, 1\}, \quad \forall k \in K, \forall w \in W, \forall ij \in A. \quad (3)$$

Equalities (1) are the flow conservation constraints, they require that a unique path between o_k and d_k is associated with each commodity k . Inequalities (2) are the capacity constraints for each installed module.

3 Aggregated formulation and Bin-Packing

Suppose now that G consists of nodes i, j connected by a single edge ij . Then the CND problem here, is to determine the number of modules to install over ij , in such a way that each commodity using ij is assigned to at most one module and the total cost is minimum. Consider the polyhedron:

$$P_{ij} := \text{conv}\{(x, y) \in \{0, 1\}^{|K| \times |W|} \times \{0, 1\}^{|W|} :$$

$$\sum_{k \in K} D^k x_{ij}^{kw} \leq Cy_{ij}^w \quad \forall w \in W, \quad \sum_{w \in W} x_{ij}^{kw} \leq 1 \quad \forall k \in K\}$$

P_{ij} is the convex hull of CND problem restricted to ij . If K and W are a set of objects and a set of bins, respectively, then P_{ij} corresponds to the Bin-Packing polytope. Note that the polyhedron P_{ij} has many symmetric solutions. To overcome this difficulty, we will introduce a new *aggregated* model that does not specify which copy of the arc ij is used for the routing of a commodity k . Indeed, the idea is just to determine the number of modules that have to be installed, so that each commodity can be assigned to one of these modules. We will define the additional integer design variable y_{ij} as the number of modules installed on ij . We also define the variables x_{ij}^k that takes the value 1, if k uses some copy of the arc ij for its routing and 0 otherwise. The CND problem

can then be formulated using the following ILP:

$$\min \sum_{ij \in A} c_{ij} y_{ij}$$

$$\sum_{j \in V} x_{ji}^k - \sum_{j \in V} x_{ij}^k = \begin{cases} 1, & \text{if } i = d_k, \\ -1, & \text{if } i = o_k, \\ 0, & \text{otherwise,} \end{cases} \quad \begin{array}{l} \forall k \in K, \\ \forall i \in V, \end{array} \quad (4)$$

$$\sum_{k \in K} D^k x_{ij}^k \leq C y_{ij}, \quad \forall ij \in A, \quad (5)$$

$$(x_{ij}^k, y_{ij}) \in Q_{ij}, \quad \forall ij \in A, \forall k \in K. \quad (6)$$

where

$$Q_{ij} := \text{conv}\{(x, y) \in \{0, 1\}^{|K|} \times \mathbb{Z}^+ : x_{ij}^k = \sum_{w \in W} x_{ij}^{kw}, y_{ij} \geq \sum_{w \in W} y_{ij}^w,$$

$$\sum_{k \in K} D^k x_{ij}^{kw} \leq C y_{ij}^w \quad \forall w \in W, x_{ij}^{kw} \in \{0, 1\}, y_{ij}^w \in \{0, 1\}, \forall k \in K, \forall w \in W\}$$

As in formulation (1)-(3), equalities (4) are the flow conservation constraints for each commodity of K . Constraints (5) and (6) express the capacity constraints over the polyhedron Q_{ij} .

Q_{ij} is the projection on (x_{ij}^k, y_{ij}) of the polyhedron P_{ij} . Observe that the symmetric solutions of P_{ij} will project on a single point. We denote by $BP(S)$ the solution of the Bin-Packing problem for a subset S of K . In other words, $BP(S)$ is the minimum number of bins needed to carry the objects of S . We also introduce $S(x)$ that denotes the subset of objects corresponding to incidence vector x . Then we provide an alternative definition of Q_{ij} :

$$Q_{ij} := \text{conv}\{(x, y) \in \{0, 1\}^{|K|} \times \mathbb{Z}^+ : y_{ij} \geq BP(S(x))\}$$

Q_{ij} would then be more suitable to investigate. This polyhedron is associated with a problem that will be referred to as *Bin-Pack Function* (BPF). Since polyhedra Q_{ij} are identical for every $ij \in A$, in the remaining of this article, we omit the indices ij . We then refer to Q_{ij} as Q , x_{ij}^k as x^k and y_{ij} as y .

4 Polyhedral analysis and valid inequalities

The purpose of this section is to discuss the polyhedron Q . We will describe its dimension, identify some valid inequalities and give necessary and sufficient

conditions for these inequalities to be facet defining.

Proposition 4.1 *Q is full dimensionnal.*

Proof. We will exhibit $|K|+2$ solutions whose incidence vectors are affinely independent. Let us introduce the $|K|$ solutions S_k , $k \in K$, defined such that one module is used to satisfy the commodity k , while the other commodities are not satisfied. Consider the incidence vector associated with each S_k , given by $(0, \dots, x^k = 1, 0, \dots, y = 1)$. We denote by S_{k_1, k_2} , the solution defined as follows. Suppose that three modules are installed and two commodities, say k_1 and k_2 , are satisfied. The incidence vector of S_{k_1, k_2} is given by $(0, \dots, x^{k_1} = 1, x^{k_2} = 1, 0, \dots, y = 3)$. Consider now the solution S_0 where no commodity is satisfied and no module is installed. The associated incidence vector is then given by $(0, \dots, 0)$. It is clear that S_0 , S_{k_1, k_2} , and S_k , $k \in K$, are feasible solutions and their incidence vectors are affinely independent. \square

Theorem 4.2 *$x^k \geq 0$ and $x^k \leq 1$ define facets for Q .*

We will not give the proofs for these inequalities as they are very similar to proof of Proposition 4.1. We will however focus in the following sections on introducing new classes of facet defining inequalities.

4.1 Valid inequalities

Proposition 4.3 *For each subset $S \subseteq K$ and a non negative integer $p \in \mathbb{Z}^+$, inequality*

$$\sum_{k \in S} x^k \leq y + p, \quad (7)$$

is valid for Q if and only if $BP(S) \geq |S| - p$.

Proof. (\Leftarrow) Suppose that the solution of the BPF problem for a subset $S \subseteq K$ verifies $BP(S) \geq |S| - p$, for some $p \in \mathbb{Z}^+$. Then by definition of the polyhedron Q , we have $y \geq BP(S) \geq |S| - p$. Hence, we have $|S| \leq y + p$. In consequence $\sum_{k \in S} x^k \leq |S| \leq y + p$. Thus inequality (7) is valid for Q ,
(\Rightarrow) Suppose now that $BP(S) < |S| - p$, then the solution $(1, 1, 1, \dots, x^{|S|} = 1, 0, \dots, y = BP(S))$ is cut off by (7). \square

Theorem 4.4 *For each $S \subseteq K$ and a non negative integer p , inequalities (7) define facets for Q if and only if the following conditions hold*

- (i) $BP(S) = |S| - p$,
- (ii) $BP(S \cup \{\tilde{s}\}) = |S| - p$, where \tilde{s} is the largest element of $K \setminus S$,
- (iii) $BP(S \setminus \{\bar{s}\}) < |S| - p$, where \bar{s} is the smallest element of S .

Proof. (\Rightarrow) We show that (i), (ii) and (iii) are necessary conditions for (7) to define facets.

- (i) Consider a subset S and a non negative integer p such that the inequality (7) induced by S and p defines a facet. Then, there exists at least one solution (x^*, y^*) such that $\sum_{k \in S} x^{*k} = y^* + p$. We have by definition of polyhedron Q that $y^* \geq BP(S)$. Thus, $BP(S) \leq \sum_{k \in S} x^{*k} - p$, and then

$$BP(S) \leq |S| - p \quad (8)$$

We also have by the validity condition of (7) that

$$BP(S) \geq |S| - p \quad (9)$$

Hence, by (8) and (9), we conclude that $BP(S) = |S| - p$,

- (ii) Suppose that there exists an element \tilde{s} of $K \setminus S$ such that $BP(S \cup \{\tilde{s}\}) \leq |S| - p$. Then the inequality (7) is dominated by another constraint

$$\sum_{k \in S \cup \{\tilde{s}\}} x^k \leq y + p$$

In consequence, (7) can not be a facet of Q .

- (iii) If $BP(S \setminus \{\bar{s}\}) \geq |S| - p$, we can see that (7) is dominated by

$$\sum_{k \in S \setminus \{\bar{s}\}} x^k \leq y + p$$

and $x^k \leq 1$, for $k = \bar{s}$, Thus (7) can not define facets for Q .

- (\Leftarrow) Let \bar{S} be a subset of K and \bar{p} a given non negative integer. Suppose that inequality (7) induced by \bar{S} and \bar{p} , is valid for Q . We will exhibit $|K| + 1$ solutions denoted by S_k , $k \in \{1, \dots, |K| + 1\}$ of BPF that satisfy this constraint with equality. M_1 denotes a $(|K| + 1) \times (|K| + 1)$ matrix containing the incidence

vectors of solutions S_k , $k \in \{1, \dots, |K| + 1\}$.

$$M_1 = \begin{array}{c} \begin{matrix} & x^1 & x^2 & x^3 & \dots & x^{|\bar{S}|} & x^{|\bar{S}|+1} & \dots & x^{|K|} & y \\ S_1 & \left(\begin{matrix} 1 & 1 & 1 & \dots & 1 & 0 & \dots & 0 & |\bar{S}| - \bar{p} \\ S_2 & \begin{matrix} 1 & 1 & 1 & \dots & 1 & 1 & \dots & 0 & |\bar{S}| - \bar{p} \end{matrix} \\ \vdots & \end{matrix} \right) \\ M_1 = & S_{|K \setminus \bar{S}|+1} \\ & \begin{matrix} 1 & 1 & 1 & \dots & 1 & 0 & \dots & 1 & |\bar{S}| - \bar{p} \\ S_{|K \setminus \bar{S}|+2} & \begin{matrix} 1 & 1 & 1 & \dots & 0 & 0 & \dots & 0 & |\bar{S}| - \bar{p} - 1 \end{matrix} \\ \vdots & \end{matrix} \\ S_{|K|+1} & \begin{matrix} 0 & 1 & 1 & \dots & 1 & 0 & \dots & 0 & |\bar{S}| - \bar{p} - 1 \end{matrix} \end{matrix} \end{array}$$

We can easily check that the incidence vectors of S_k , $k \in \{1, \dots, |K \setminus \bar{S}| + 1\}$ verify conditions (i) and (ii). Indeed, the value of y in incidence vector of S_1 ensures $BP(\bar{S}) = |\bar{S}| - \bar{p}$, and adding to \bar{S} the greatest commodity of $K \setminus \bar{S}$ does not change the value of $BP(\bar{S})$. On the other hand, incidence vectors of $S_{|K \setminus \bar{S}|+2}$ to $S_{|K|+1}$ verify the condition (iii), as the removal of any commodity of \bar{S} yields the decreasing of $BP(\bar{S})$ value. Moreover, the incidence vectors of S_k , $k \in \{1, 2, \dots, |K| + 1\}$ are affinely independent. \square

Proposition 4.5 *For $S \subseteq K$ and a parameter $q \in \mathbb{Z}_*^+$, inequality*

$$\sum_{k \in S} x^k \leq qy, \quad (10)$$

is valid for Q if and only if $BP(S') \geq \lceil \frac{|S'|}{q} \rceil$, $\forall S' \subseteq S$.

Theorem 4.6 *For $S \subseteq K$ and a given parameter $q \in \mathbb{Z}_*^+, q \geq 2$, inequalities (10) are facets defining for Q if and only if $BP(S') \geq \lceil \frac{|S'|}{q} \rceil$, $\forall S' \subseteq S$.*

5 Computational results

Based on the theoretical results described above, we devised a branch-and-cut algorithm that has been implemented in C++ using CPLEX 12.0 with the default settings. We have proposed a heuristic separation procedure for inequalities (7), that uses a greedy algorithm to find S , strengthened by an exact evaluation of $BP(S)$. We have tested our approach on several instances derived from SNDlib topologies (<http://sndlib.zib.de>), with a restricted subset of commodities. The obtained results are presented in Table 1 with the following entries. The first three columns contain the size of each instance.

In column four, we can find the value of the solution given by the aggregated formulation Z_{AF} , by only considering constraints (4) and (5). Those solutions ignore the Bin-Packing structure of the problem and usually are not feasible. Z_{BC} is an upper bound found by the branch-and-cut over the aggregated formulation including the cuts from the Bin-Pack Polyhedron, optimum if the instance is solved to optimality. Z_{CF} is an upper bound found by the compact formulation, optimum if the instance is solved to optimality. The remaining columns are the number of constraints (7) separated, and the CPU time for branch-and-cut and the compact formulation, denoted T_{BC} and T_{CF} , respectively (given in days:hours:min:sec). In Z_{CF} column, values were written in

Table 1
Results for instances with $|W|=4$

$ V $	$ A $	$ K $	Z_{AF}	Z_{BC}	Z_{CF}	#CutsI	T_{BC}	T_{CF}
12	36	20	24000.00	25000.00	25000.00	3	0:00:02.27	0:00:32.31
15	44	10	17256.00	17720.00	17720.00	3	0:00:00.97	0:00:12.00
15	44	20	32806.00	32806.00	32806.00	1	0:00:11.41	0:10:22.31
17	52	15	4692.10	5105.90	5105.90	11	0:00:21.99	0:01:30.05
17	52	20	6165.00	6416.40	6603.80	15	0:06:47.49	1:00:00.00
22	72	15	35332.00	35942.00	35942.00	4	0:01:26.16	0:49:06.23
22	72	20	38172.00	38172.00	38481.00	0	0:04:06.69	1:00:00.00
28	82	15	10528.00	10528.00	10528.00	0	0:00:24.70	0:08:27.72
51	160	20	5700.00	5800.00	5800.00	1	0:02:37.35	1:00:00.00

italics if the optimal solution could not be found within 1 hour. Notice that for most of the instances, adding only few cuts helped to find the optimal solution in a short time. These results are very promising and show the efficiency of facets (7). We are now implementing the separation of (10) and some other facets that were not presented in this paper.

References

- [1] A. Atamtürk, O. Günlük. *Network design arc set with variable upper bounds*. Networks, vol. 50, pages 17-28, 2007.
- [2] D. Bienstock, G. Muratore. *Strong inequalities for capacitated survivable network design problems*. Mathematical Programming, vol. 89, issue 1, pages 127-147, November 2000.
- [3] S. Martello, P. Toth. *Knapsack Problems : Algorithms and Computer Implementations*. Wiley, New York, 1990.

A branch-and-cut algorithm for the Multiple Steiner TSP with Order constraints

S. Borne¹

LIPN, Paris 13 University, France

A.R. Mahjoub, R. Taktak²

LAMSADE, Paris Dauphine University, France

Abstract

The paper deals with a problem motivated by survivability issues in multilayer IP-over-WDM telecommunication networks. Given a set of traffic demands for which we know a survivable routing in the IP layer, our purpose is to look for the corresponding survivable topology in the WDM layer. The problem amounts to Multiple Steiner TSPs with order constraints. We propose an integer linear programming formulation for the problem and investigate the associated polytope. We also present new valid inequalities and discuss their facial aspect. Based on this, we devise a Branch-and-cut algorithm and present preliminary computational results.

Keywords: IP-over-WDM networks, Steiner TSP, order constraint, Branch-and-cut algorithm.

¹ Email: sylvie.borne@lipn.univ-paris13.fr

² Email: {mahjoub,taktak}@lamsade.dauphine.fr

1 Introduction

Multilayer Network Design and survivability problems have recently seen a particular attention [1,2,3]. The problem that we are studying in this paper deals with survivability in multilayer telecommunication networks. Consider an IP-over-WDM network consisting of a logical IP layer over an optical WDM layer. The IP layer is composed of IP routers interconnected by logical links and the WDM layer consists of optical switches interconnected by optical links. To each router in the IP layer corresponds an optical switch in the WDM layer. The links between the IP routers are logical and are ensured by paths in the optical layer. We suppose given a set K of demands such that for each demand we know two node-disjoint paths routing it in the IP layer. Besides, with each optical link in the WDM layer is associated a positive cost for its installation.

The *Multilayer Survivable Optical Network Design* problem (MSOND problem) is to find, for each demand, two node-disjoint optical paths routing it in the WDM layer. These paths must go in the same order through the optical switches corresponding to the routers visited in the logical paths of the IP layer, and such that the total cost is minimum. The optical switches that must be visited for a demand $k \in K$ are called *terminals* and the other nodes are called *Steiner nodes* for this demand. Consider a demand $k \in K$, looking for two paths respecting some order through the terminals of demand k amounts to looking for a cycle visiting these terminals in a predefined order. This is in a close relationship with the Steiner Cycle problem or the Steiner TSP [4]. However, an additional constraint related to the order between the terminals is here considered. The MSOND problem is hence nothing but a succession of Steiner TSPs with a specific order on the terminals for each demand $k \in K$.

The paper is organized as follows. In section 2, we introduce further notations and propose a linear integer programming formulation for the problem. In section 3, we investigate the associated polytope. In section 4, we give new families of valid inequalities and discuss the facial aspects for some constraints. Finally, section 5 will be devoted to present preliminary computational results.

2 Notations and formulation

As previously mentioned, in the MSOND problem, optimization concerns only the WDM layer. Let us associate with this layer an undirected graph $G = (V, E)$ where V corresponds to the optical switches and E to the optical links between these switches. For each demand $k \in K$, let T_k represent the set of its

terminals and S_k the set of its Steiner nodes. As the terminals of each demand must be visited in a predefined order, demand $k \in K$ can also be represented by a sequence of terminals $(w_1^k, w_2^k, \dots, w_{l_k}^k)$. Two successive terminals, w_j^k and w_{j+1}^k , where $j = 1, \dots, l_k$ and $w_{l_k+1}^k = w_1^k$, define a *section* q_j^k of demand k . For $W \subset V$, we denote by $\delta_G(W)$ (or if the context is clear $\delta(W)$) the set of edges in G having exactly one node in W . $\delta(W)$ is called a *cut*. For $W \subset V$ such that $w_j^k \in W$ and $w_{j+1}^k \in V \setminus W$, $\delta_{G^{k,j}}(W)$ will denote the cut separating the terminals w_j^k and w_{j+1}^k in the graph $G^{k,j}$. Here, $G^{k,j}$ is the graph obtained from graph G by deleting all the terminals of demand k excepted w_j^k and w_{j+1}^k .

Let y_e , $e \in E$ be a variable such that y_e is equal to 1 if edge e is taken and 0 otherwise. Given a demand $k \in K$ and an edge $e \in E$, x_e^k will define a variable which takes 1 if demand k is routed using edge e and 0 otherwise.

The MSOND problem is equivalent to the following ILP.

$$\begin{aligned} & \min \sum_{e \in E} c(e) y_e \\ & \sum_{e \in \delta_{G^{k,j}}(W)} x_e^k \geq 1 && \text{for all } k \in K, q_j^k = (w_j^k, w_{j+1}^k), W \subset V, \\ & && w_j^k \in W \text{ and } w_{j+1}^k \in V \setminus W \quad (1) \\ & \sum_{e \in \delta(w)} x_e^k \leq 2 && \text{for all } w \in V, k \in K \quad (2) \\ & x_e^k \leq y_e && \text{for all } e \in E, k \in K \quad (3) \\ & 0 \leq x_e^k, y_e \leq 1 && \text{for all } e \in E, k \in K \quad (4) \\ & x_e^k \in \{0, 1\}, y_e \in \{0, 1\} && \text{for all } e \in E, k \in K \quad (5) \end{aligned}$$

Inequalities (1) ensure for each section q_j^k , $j = 1, \dots, l_k$ of a demand $k \in K$ a path in the reduced graph $G^{k,j}$. This guarantees for each demand two paths passing in order through its terminals. Inequalities (2) ensure the node-disjunction between these paths. Inequalities (3) are the linking constraints. Inequalities (4) and (5) are the trivial and integrity constraints, respectively.

3 Associated polytope

Denote by $\text{MSOND}(G, K, T)$ the convex hull of the incidence vectors of the solutions of (1)-(5) associated with graph G , the demands' set K and the set $T = \bigcup_{k \in K} T_k$ of terminals of the different demands. In the sequel, we suppose that G is complete and that each demand $k \in K$ has at least 2 Steiner nodes ($|S_k| \geq 2$). We have the following remarks.

Remark 3.1 Given a demand $k \in K$ and a terminal node $w_j^k \in T_k$, the following equation is valid for $\text{MSOND}(G, K, T)$.

$$\sum_{e \in \delta(w_j^k)} x_e^k = 2 \quad (6)$$

Remark 3.2 Consider a demand $k \in K$ and two non-successive terminals $w_i^k, w_j^k \in T_k$. Let $e' = w_i^k w_j^k$. We then have

$$x_{e'}^k = 0 \quad (7)$$

Now, we can state the dimension of $\text{MSOND}(G, K, T)$.

$$\text{Theorem 3.3 } \dim(\text{MSOND}(G, K, T)) = (|K| + 1)|E| - \sum_{k \in K} \frac{|T_k|(|T_k| - 1)}{2}$$

Proof (Sketch) Consider an equation $ax + by = \beta$ of $\text{MSOND}(G, K, T)$, we prove that $b = 0$, and $ax = \beta$ is a linear combination of equations (6) and (7), which implies that (6) and (7) are the only equations of $\text{MSOND}(G, K, T)$. Denote by M the matrix of equations of $\text{MSOND}(G, K, T)$. M looks as follows

$$M = \begin{pmatrix} M_1 & & & \\ & M_2 & & \\ & & \ddots & \\ & & & M_K \end{pmatrix} \text{ where } M_k \text{ is the matrix of equations (6) and (7) for demand}$$

$k \in K$. Since for $k \in K$, there are $|T_k|$ equations of (6) and $\frac{|T_k|(|T_k| - 1)}{2} - |T_k|$ equations of (7), it follows that $\text{rank}(M_k) = |T_k| + (\frac{|T_k|(|T_k| - 1)}{2} - |T_k|) = \frac{|T_k|(|T_k| - 1)}{2}$. By construction of M , we deduce that $\text{rank}(M) = \sum_k \frac{|T_k|(|T_k| - 1)}{2}$. As $\dim(\text{MSOND}(G, K, T)) = N - \text{rank}(M)$, the result follows. Here $N = (|K| + 1)|E|$ represents the total number of variables. \square

4 Valid inequalities and facial aspect

In this section, we describe some classes of valid inequalities for $\text{MSOND}(G, K, T)$. These are given in the following theorems.

Theorem 4.1 Consider a demand $k \in K$ and let $W \subset V$ such that $W \cap T_k \neq \emptyset \neq (V \setminus W) \cap T_k$. Then the following inequality is valid for $\text{MSOND}(G, K, T)$.

$$\sum_{e \in \delta(W)} x_e^k \geq 2 \quad (8)$$

Inequalities (8) are a straight consequence related to the connectivity requirements of the problem and will be called the *Steiner 2-connectivity inequalities*.

Theorem 4.2 Consider a demand $k \in K$. Let $w_j \in T_k$ be a terminal node and $S \subseteq S_k$. Denote $E_j = [S, \{w_j\}]$ and $F_j = [S, \{w_{j+2}, \dots, w_{j-2}\}]$. Then

$$\sum_{e \in \delta(S) \setminus \{E_j, F_j\}} x_e^k \geq \sum_{e \in E_j} x_e^k \quad (9)$$

is valid for MSOND(G, K, T).

Inequalities (9) are called the *Steiner non-successive terminals inequalities*. These can be seen as flow inequalities and are saying the following. The flow going from w_j to a subset of Steiner nodes $S \subseteq S_k$ (corresponding to the flow circulating through edges E_j) must be used to route only sections that are adjacent to w_j (corresponding to $\delta(S) \setminus \{E_j, F_j\}$).

Theorem 4.3 Consider a demand $k \in K$ and let V_0, \dots, V_p be a partition of V such that $|V_i \cap T^k| \geq 1, i = 1, \dots, p$ and $V_0 \cap T^k = \emptyset$. Let $F \subseteq \delta(V_0)$ such that $|F|$ is odd. Then

$$x^k(\delta(V_0, \dots, V_p) \setminus F) \geq p - \lfloor \frac{|F|}{2} \rfloor \quad (10)$$

is valid for MSOND(G, K, T).

These inequalities are called the *Steiner F-partition inequalities*.

Proof. Clearly, the following inequalities are valid for MSOND(G, K, T),

$$\begin{aligned} x^k(\delta(V_i)) &\geq 2 && \text{for all } i = 1, \dots, p \\ -x^k(f) &\geq -1 && \text{for all } f \in F \\ x^k(g) &\geq 0 && \text{for all } g \in \delta(V_0) \setminus F \end{aligned}$$

The result follows by summing these inequalities, dividing by 2 and rounding up the right-hand side. \square

Theorem 4.4 Consider a demand $k \in K$ and let V_1, \dots, V_p be a partition of V such that $|V_i \cap T^k| \geq 1, i = 1, \dots, p$. Suppose that $r \leq p$ subsets in the partition contain respectively $q_i, i = 1, \dots, r$ non-successive terminals (or sequences of terminals). Let $S \subseteq S_k$ be a subset of Steiner nodes of demand k . Then

$$x^k(\delta_{G \setminus S}(V_1, V_2, \dots, V_p)) \geq (p + \sum_{i=1}^r q_i - r) - |S| \quad (11)$$

is valid for MSOND(G, K, T).

These inequalities are called the *Steiner partition inequalities*.

Proof (Sketch) The idea of the proof is to replace each subset V_i , $i = 1, \dots, r$ which contains q_i non-successive terminals (or sequences of terminals) by q_i subsets each one containing either a unique terminal or a sequence of successive terminals. The proof is by induction on r . \square

We have investigated the facial aspect for all the above valid inequalities. Because of the space limit of the paper, we will give the results only for inequalities (2) and (8).

Theorem 4.5 *Inequalities (2) define facets for $\text{MSOND}(G, K, T)$ if and only if w is not a terminal for demand $k \in K$.*

Proof. Let $F = \{(x, y) \in \text{MSOND}(G, K, T) : x^k(\delta(w)) = 2\}$ be the facet induced by inequalities (2). If $w \in T_k$ then $F = \text{MSOND}(G, K, T)$. Consequently, F is not a proper face and hence it is not facet defining. Assume now that $w = s \in S_k$. We will exhibit $\dim(\text{MSOND}(G, K, T))$ affinely independent solutions of F . To this end, we first suppose that s behaves like a terminal node for demand k . Assume that s behaves like a terminal between the terminals w_1^k and w_2^k . Consider the new polytope $\text{MSOND}(G, K, T')$ where $T' = (T \setminus T_k) \cup T'_k$ and $T'_k = T_k \cup \{s\}$. By theorem 3.3, there are $(|K| + 1)|E| - \sum_{h \in K} \frac{|T_h|(|T_h|-1)}{2} - |T_k| + 1$ affinely independent solutions in $\text{MSOND}(G, K, T')$. These solutions are in F as well. Now we complete these solutions by $|T_k| - 1$ additional ones obtained as follows. For all the demands $h \neq k \in K$, we route the demand h by considering the edges $\{w_j^h, w_{j+1}^h\}$, where $j = 1, \dots, l_h$ and $w_{l_h+1}^h = w_1^h$, between the terminals of T_h . For demand k , a first solution is obtained by routing on the edges between the successive terminals for all the sections of the demand excepted section (w_2^k, w_3^k) . For this section, the routing is ensured by inserting the Steiner node s between (w_2^k, w_3^k) , which gives a feasible solution for $\text{MSOND}(G, K, T)$ that is in F . The same procedure is applied to sections $(w_3^k, w_4^k), \dots, (w_{l_k}^k, w_1^k)$. And this leads to exactly $|T_k| - 1$ new solutions belonging to F . By construction, all the previous solutions are affinely independent in F and the result follows. \square

Theorem 4.6 *Inequalities (8) define facets for $\text{MSOND}(G, K, T)$ if and only if the two following conditions hold:*

- (i) *either $|W \cap T_k| = 1$ or $W \cap T_k$ is a sequence of successive terminals,*
- (ii) *$W \cap S_k \neq \emptyset$ and $(V \setminus W) \cap S_k \neq \emptyset$.*

Proof (Sketch) Let $F = \{(x, y) \in \text{MSOND}(G, K, T) : x^k(\delta(W)) = 2\}$ be the face induced by inequality (8). If W contains non-successive terminals or non-successive sequences of terminals, then $x^k(\delta(W)) \geq 4$. This implies that $F = \emptyset$

and hence F does not define a facet. Now, assume that W contains either only one terminal or a sequence of successive terminals. In this case, if $W \cap S_k = \emptyset$ then $F = \text{MSOND}(G, K, T)$ and hence F is not a facet defining. Assume now that conditions (1) and (2) are satisfied. Consider a valid inequality $ax + by \leq \beta$ for $\text{MSOND}(G, K, T)$ and let F' be the corresponding induced face $F' = \{(x, y) \in \text{MSOND}(G, K, T) : ax + by = \beta\}$. Suppose that $F \subseteq F'$. The result follows by proving that $ax + by = \beta$ is a linear combination of equations $x^k(\delta(W)) = 2$, (6) and (7). \square

5 Computational results

Using the previous results, we devise a Branch-and-Cut algorithm. This is tested on three SNDlib-based instances (*polska*, *newyork* and *pioro40*) with a number of demands ranging from 8 to 30. The maximum CPU time is fixed to 2 hours. The results are reported in Table 1. The columns of the table represent: the name of the instance, the number of nodes (V), the number of demands (K), the average number of terminals (T), the number of generated cuts for inequalities (1) (#C), (8) (#S2C) and (9) (#SNST) respectively, the relative error between the best upper bound and the lower bound obtained at the root node (gap) and finally the total time of execution in hours:min:sec (CPU). We can observe that all the instances *polska* where solved to optimality within less than 6 minutes, which means that our algorithm performs well for relatively small instances. The resolution of the problem becomes harder when the size of instances grows. In fact, instances *newyork* and *pioro40* took much more time to be solved. In particular, for *pioro40* with 12 demands, no feasible solution could be found within the time limit. We can also note that inequalities (8) and (9) are quite efficient, they permit to strengthen the linear relaxation. In fact, as it can be seen, all the solved instances have a gap not exceeding 10%. However, for bigger instances, the use of further valid inequalities would be necessary. In this perspective, more significant computational results, details about separation routines and a deeper facial investigation will be presented.

6 Concluding remarks

In this paper we studied a problem consisting of multiple Steiner TSPs with order constraints. We proposed an integer linear programming formulation for the problem and studied the associated polytope. We introduced new valid inequalities and discussed some facial aspects. Using this, we devised

Instance	<i>V</i>	<i>K</i>	<i>T</i>	#C	#S2C	#SNST	Gap(%)	CPU
polska	12	8	4,88	329	863	135	4,38	0: 00: 08
polska	12	12	4,25	693	2808	202	8,65	0: 00: 58
polska	12	14	4,29	894	3494	202	8,65	0: 01: 18
polska	12	15	4,47	904	3627	202	8,65	0: 01: 25
polska	12	16	4,44	1029	4135	234	8,38	0: 01: 43
polska	12	18	4,28	723	2319	298	5,83	0: 00: 38
polska	12	20	4,35	764	2561	298	5,83	0: 00: 51
polska	12	25	3,92	1096	4497	330	5,83	0: 01: 44
polska	12	30	3,97	1610	7462	394	8,28	0: 06: 35
newyork	16	12	4,17	1536	6762	96	7,81	0: 14: 11
newyork	16	14	4,36	1372	5833	199	7,14	0: 10: 33
newyork	16	15	4,4	2319	10915	247	8,06	0: 31: 53
newyork	16	16	4,38	2072	11214	247	8,06	0: 36: 47
newyork	16	18	4,44	1111	4198	302	6,19	0: 03: 20
newyork	16	20	4,5	2279	11124	398	8,76	0: 39: 29
pioro40	40	8	4,5	2963	262	576	0,72	0: 03: 17
pioro40	40	10	4,5	2735	1123	720	1,4	0: 10: 52
pioro40	40	12	4,67	7749	6680	864	4,34	2: 00: 00

Table 1
Preliminary results for the Branch-and-cut algorithm

a Branch-and-cut algorithm that has been tested on SNDlib-based instances. The first results show the efficiency of the valid inequalities to improve the linear relaxation of the formulation. A deeper facial investigation and more significant computational results will be further presented.

References

- [1] Borne, S., Gourdin, E., Liau, B. and A.R. Mahjoub, *Design of survivable IP-over-optical networks*, Ann. Oper. Res. **146** (2006), 41–73.
- [2] Gouveia, L., Patrício, P. and A. de Sousa, *Hop-Constrained Node Survivable Network Design: An Application to MPLS over WDM*, Networks and Spatial Economics **8** (2008), 3–21.
- [3] Orlowski, S., Raack, C., Koster, A. M. C. A., Baier, G., Engel, T. and P. Belotti, *Branch-and-Cut Techniques for Solving Realistic Two-Layer Network Design Problems*, In: Graphs and Algorithms in Communication Networks, (2010), 95–118, Springer.
- [4] Salazar-González, J.J., *The steiner cycle polytope*, Eur. J. of Op. Res. **147** (2003), 671–679.

Optimal Client-Server Configuration of Mobile Ad-Hoc Networks

Luigi De Giovanni¹, Claudio E. Palazzi^{2,3}

*Dipartimento di Matematica
Università degli Studi di Padova
Padua, Italy*

Abstract

Interactive online gaming is a widely successful application that is becoming more and more popular even on mobile devices. In this context, an emerging scenario is represented by having players' mobile devices directly connected to each other, without resorting to the Internet. To enable this scenario, we leverage on a hybrid architecture that allows client-server games to be played in ad-hoc mode: a number of players take turn in acting also as the server of the game. However, the problem on how to choose the servers so as to maximize the duration of the network still remains an issue. To this aim, we propose an Integer Programming model and a heuristic based on it: preliminary experimental results show the effectiveness and the potential of the approach.

Keywords: Online Game, Backup Server, Energy, MANET, Mixed Integer Linear Programming, Tabu Search.

¹ Email: luigi@math.unipd.it

² Email: cpalazzi@math.unipd.it

³ Partial financial support for this work is provided by MIUR/PRIN ALTER-NET and the UNIPD/PRAT Web Squared projects.

1 Introduction

The popularity of multiplayer online games and the endemic diffusion of connectivity-enabled mobile devices foster new gaming scenarios based on players' location, proximity and physical interaction [5], as well as on exploiting ad-hoc, direct connectivity [4]. However, critical technical issues emerge. For instance, the main gaming model today is client-server, with the client run by the player's PC and the server located remotely in the Internet [6]. In this model, players' game actions are transmitted toward the server, which updates the global game state and periodically sends it to all its clients. To facilitate the adaptation of existing online games, the client-server model has to be adoptable even in the mobile ad-hoc network (MANET) scenario. As MANETs are not connected to the Internet, one of the mobile devices participating to the game should also act as the game server. Even more critical, energy consumption becomes a major concern, because of the limited energy in a battery. The worst energy consumption is experienced by the server node. Indeed, not only that node has to run the game client to let its user play, but it also has to receive all game actions from the other players, compute the new game state, and forward back to all its clients updated information about their *area-of-interest* within the game arena. To address these problems, a hybrid architecture could be employed (e.g., [4]): more than one node is able to act as server and, in turn, one of these nodes becomes the *active server*, whereas the others remain *backup servers*. In this approach, in addition to the activity described above, with a certain periodicity, the active server forwards the whole game state view to the backup servers, so as to have them ready to take in charge. Having more servers taking turn in becoming active distributes the highest energy consumption burden on more nodes, thus allowing a longer gaming session. Yet, communication among nodes is responsible as well for energy consumption, and the amount of communication depends on routing strategies and on servers' location, which are hence crucial to maximize the time before the first node in the network exhausts its battery, thus interrupting the game.

In this paper we consider the Client-Server MANET Configuration Problem (CSMCP), aiming at finding an optimal configuration of a MANET in terms of servers' location and routing strategies taking into account that: the number of servers is given; each server is active for the same amount of time (we assume a rotating shifts policy with a high enough rotation frequency); communications between nodes are made in unicast, like actual games (the active server sends a copy of a packet for each destination), and can use multi-

path routing; the game is interrupted as soon as the first node in the network exhausts its battery; node positions do not sensibly change during the game; the objective is to maximize the time before the game is interrupted and, this time being equal, to minimize the total energy consumed.

Preliminary approaches based on simulation are presented in [4], while, to the best of our knowledge, the problem is new to optimization literature and is related to both MANET configuration and routing optimization (e.g., [1]).

In this paper we propose a Mixed Integer Linear Programming (MILP) formulation and a simple tabu search heuristic to solve the problem in a centralized manner (for example, after the election of a network coordinator), or to provide lower bounds or suggestions to any distributed approach. Both the model and the heuristic are presented in Section 2. Preliminary tests show that the model is able to solve to proven optimality instances of up to a realistic size of about 30 nodes, while the heuristic provides optimal or near optimal solutions even for larger instances and in fairly less time: results are presented in Section 3, which concludes the paper and suggests future extensions.

2 An Integer Programming formulation and a heuristic

The input data for CSMCP are the following. A number of servers L is given, together with the set N of the network nodes and the set E of communication links. Each link corresponds to an ordered pair $(i, j) \in N \times N$, such that it is possible to send packets from i to j (e.g., if i and j are close enough). Each link (i, j) has a maximum bandwidth U_{ij} . For each node v , we are given the initial battery level (A_v), the average energy consumed per time unit by v as client (P_v^C), and the additional energy consumed by v in case it acts as active (resp. backup) server (P_v^S , resp. P_v^D). Given a node v , data on traffic refers to three types of packets: packets to be routed from v as client to the active server (type C), packets from the active server to v as client (type S), packets from the active server to v as backup server (type D). Depending on the type and on the node v , packets have different size and transmission rate: B_v^C , B_v^S and B_v^D are the given bandwidths required for packets of type C , S and D related to v . Finally, the data $T_{k\tau}^{\tau}$, (resp. $R_{ik\tau}^{\tau}$) refer to the energy per time unit consumed by node k for transmitting (resp. receiving) on link (k, j) (resp. (i, k)) the packets of type $\tau \in \{C, S, D\}$ related to node v .

A MILP model for CSMCP is presented in Figure 1. It is based on the following decision variables: σ_v^l , boolean variables taking value 1 if node v is the active server in the shift $l \in \{1..L\}$, 0 otherwise; x , y and z , determining the routing strategy. In particular, x_{ij}^{lv} (resp. y_{ij}^{lv} and z_{ij}^{lv}) is the percentage of

$$\min \quad \alpha \quad + \quad \epsilon \sum_{v \in N} \zeta_v \quad (1)$$

$$\text{s.t. } A_v \alpha \geq \zeta_v \quad \forall v \in N \quad (2)$$

$$P_v^C + \left(\frac{1}{L} P_v^S + \frac{L-1}{L} P_v^D \right) \sum_{l=1}^L \sigma_v^l + \frac{1}{L} \sum_{l=1}^L (\gamma_{lv}^T + \gamma_{lv}^R) = \zeta_v \quad \forall v \in N \quad (3)$$

$$\sum_{v \in N} \sigma_v^l = 1 \quad \forall l = 1..L \quad (4)$$

$$\sum_{l=1}^L \sigma_v^l \leq 1 \quad \forall v \in N \quad (5)$$

$$\sum_{(k,j) \in \bar{E}} x_{kj}^{lv} - \sum_{(i,k) \in \bar{E}} x_{ik}^{lv} = \begin{cases} 1 & \text{if } k = v \\ -1 & \text{if } k = \bar{s} \\ 0 & \text{otherwise} \end{cases} \quad \forall k \in \bar{N}, v \in N, l = 1..L \quad (6)$$

$$\sum_{(k,j) \in \bar{E}} y_{kj}^{lv} - \sum_{(i,k) \in \bar{E}} y_{ik}^{lv} = \begin{cases} -1 & \text{if } k = v \\ 1 & \text{if } k = \bar{s} \\ 0 & \text{otherwise} \end{cases} \quad \forall k \in \bar{N}, v \in N, l = 1..L \quad (7)$$

$$\sum_{(k,j) \in \bar{E}} z_{kj}^{lv} - \sum_{(i,k) \in \bar{E}} z_{ik}^{lv} = \begin{cases} -\sum_{l=1}^L \sigma_v^l & \text{if } k = v \\ \sum_{l=1}^L \sigma_v^l & \text{if } k = \bar{s} \\ 0 & \text{otherwise} \end{cases} \quad \forall k \in \bar{N}, v \in N, l = 1..L \quad (8)$$

$$x_{k\bar{s}}^{lv} \leq \sigma_k^l, x_{\bar{s}k}^{lv} \leq \sigma_k^l, y_{k\bar{s}}^{lv} \leq \sigma_k^l, y_{\bar{s}k}^{lv} \leq \sigma_k^l, z_{k\bar{s}}^{lv} \leq \sigma_k^l, z_{\bar{s}k}^{lv} \leq \sigma_k^l \quad \forall k, v \in N, l = 1..L \quad (9)$$

$$\sum_{v \in N} B_v^C x_{ij}^{lv} + B_v^S y_{ij}^{lv} + B_v^D z_{ij}^{lv} \leq U_{ij} \quad \forall (i, j) \in E, l = 1..L \quad (10)$$

$$\sum_{v \in N} \sum_{(i,k) \in E} R_{ikv}^C x_{ik}^{lv} + R_{ikv}^S y_{ik}^{lv} + R_{ikv}^D z_{ik}^{lv} = \gamma_{lk}^R \quad \forall k \in N, l = 1..L \quad (11)$$

$$\sum_{v \in N} \sum_{(k,j) \in E} T_{kjv}^C x_{kj}^{lv} + T_{kjv}^S y_{kj}^{lv} + T_{kjv}^D z_{kj}^{lv} = \gamma_{lk}^T \quad \forall k \in N, l = 1..L \quad (12)$$

$$\sigma_j^l \leq 1 - \sigma_i^{l+1} \quad \forall l = 1..L-1, i, j \in N : i \prec j \quad (13)$$

$$\alpha \in \mathbb{R} ; \quad \zeta_v \in \mathbb{R}_+ \quad \forall v \in N \quad (14)$$

$$\gamma_{lv}^T, \gamma_{lv}^R \in \mathbb{R}_+ \quad \forall v \in N, l = 1..L \quad (15)$$

$$\sigma_v^l \in \{0, 1\} \quad \forall v \in N, l = 1..L \quad (16)$$

$$x_{ij}^{lv}, y_{ij}^{lv}, z_{ij}^{lv} \in [0, 1] \quad \forall v \in N, (i, j) \in \bar{E}, l = 1..L \quad (17)$$

Fig. 1. A MILP model for the client-server MANET gaming configuration.

packets of type C (resp. S and D) that, in shift l , are routed on link (i, j) . Other model variables are: ζ_v , the average consumption per time unit of node v ; γ_{lv}^T (resp. γ_{lv}^R), the average consumption of node v for transmitting (resp. receiving) during the shift l ; and α , the inverse of the game duration.

The objective function (1) minimizes a weighted sum of α and the total consumption: a small-enough parameter ϵ gives priority to α minimization (hence game duration maximization). Taking into account that node v exhausts its battery after $\frac{A_v}{\zeta_v}$ time units, constraints (2) bound the inverse of α (i.e., the game duration) by the minimum time at which a node goes off. Constraints (3) set variables ζ_v by summing up the energy per time unit consumed by node v (i) as client, (ii) as active (resp. backup) server during $\frac{1}{L}$ (resp. $\frac{L-1}{L}$) of the game duration, in case v is chosen as server (i.e., $\sum_{l=1}^L \sigma_v^l = 1$), and (iii) for communications, taking into account that each shift takes $\frac{1}{L}$ of the game duration. Constraints (4) provide each shift with exactly one active server, and constraints (5) guarantee that a node will be server in at most one shift. The multi-commodity flow conservation constraints (6), (7) and (8) ensure the multi-path routing of packets. To keep them linear, a dummy node \bar{s} is introduced, together with dummy links (\bar{s}, v) and (v, \bar{s}) , for all $v \in N$: \bar{N} and \bar{E} denote the sets of nodes and links extended to dummy elements, and flow variables x , y and z are extended to \bar{E} . For a node k and a shift l , constraints (6) consider the total outgoing an incoming percentages of packets of type C generated at node v (routing variables x), fixing their difference to 1 for node v itself (all the packets leave v), -1 for \bar{s} (all the packets reach the dummy active server) and 0 otherwise (no packets is kept nor created by transit nodes). Constraints (9) activate the dummy links between \bar{s} and a node i during the shift l , only if i is configured as the active server for the same shift l . As a consequence, all the traffic to/from the dummy server \bar{s} reaches/leaves the actual selected server. Hence, variables x give feasible routings for the packets of type C , differentiated by shifts. Similarly, constraints (7) and (8) ensure that y and z give feasible routings for the packets of type S and D (note that D packets are generated at a node \bar{s} and received by v only if v acts as server). Based on the routing variables giving the percentage of packets involving each link on each shift, constraints (10) compute the total bandwidth required on each link and bound it by the link capacity, while constraints (11) and (12) set variables γ^R and γ^T to the communication consumptions of each node. Finally, technical constraints (13) use any a-priori order \prec between nodes to break symmetries related to equivalent permutation of shift indexes.

As to efficiently solve instances of growing size, we have set up a heuristic algorithm based on tabu search [2] and sketched in Figure 2. It starts from an initial solution and explores the space of σ variables. Indeed, once variables σ (servers' location) are fixed, the model reduces to a Linear Program (LP) and can be efficiently solved to determine the optimal routing and the remaining variables, and evaluate the objective function (1). The initial servers' location

```

1. Find an initial solution  $\bar{S}$  and its value  $\bar{V}$ . Set  $S^* = \bar{S}$  and  $V^* = \bar{V}$ 
2. Initialize the number of iterations  $k = 0$  and, for each node  $v$ ,  $T_v = -T$ 
3. while ( the time limit is not expired ) do {
4.   Set the current solution  $S_0 = \bar{S}$  and initialize the best neighbor solution value  $\bar{V} = +\infty$ 
5.   for each pair of nodes  $(i, j)$  such that  $i$  is a server in  $S_0$  and  $j$  is not do {
6.     if  $k - T_i < T$  and  $k - T_j < T$  then continue with next node pair (tabu solution)
7.     Generate a neighbor solution by fixing variables  $\sigma$  so that  $j$  is a server and  $i$  is not
8.     Solve model (1)–(17) with fixed  $\sigma$  obtaining a new solution  $S'$  and its value  $V'$ 
9.     if  $V' < \bar{V}$  then  $\bar{S} = S'$ ,  $\bar{V} = V'$ ,  $\bar{i} = i$  and  $\bar{j} = j$  (update best neighbor)
10.    }
11.   Set  $T_{\bar{i}} = T_{\bar{j}} = k$  and  $k = k + 1$ 
12.   if  $\bar{V} < V^*$  then  $S^* = S'$  and  $\bar{V} = V'$  (update the incumbent solution)
13. }
14. return the solution  $S^*$  and its value  $V^*$ 

```

Fig. 2. Sketch of the tabu search for CSMCP.

(step 1.) is heuristically obtained by solving the MILP model (1)–(17) after eliminating variable α and related constraints (2), which implies minimizing the total consumption per time unit. This is related but not equivalent to maximizing the game duration, but allows obtaining initial values for σ in a drastically reduced amount of time. The initial solution is evaluated (routing and game duration) by fixing σ in model (1)–(17) and solving the related LP. Then the algorithm iteratively moves from the current solution to another (possibly better) one, until a given time limit expires (step 3.). At each iteration, neighbor solutions are generated by swapping a node configured as server with another node (step 7.). Each neighbor is evaluated by the corresponding LP (step 8.), and the best one is selected as the new current solution (step 9.). Not improving solutions may be accepted and the procedure may cycle over the same configurations. To avoid cycling, a parameter T is defined [2] and the solutions obtained from swapping pairs of nodes involved in the last T moves are discarded (steps 6. and 11.). Neighbors that do not satisfy the symmetry breaking constraints are discarded as well.

3 Experimental Evaluation and Conclusions

Preliminary tests have been conducted and a representative subset of the results is shown in Table 1. Results refer to a prototype implementation of the proposed model and heuristic in OPL [3], running on an Intel Core i7 2.2 GHz CPU with 8 GB RAM and using, as optimization engine, Cplex 12.4 [3] with default settings and a total time limit of one hour.

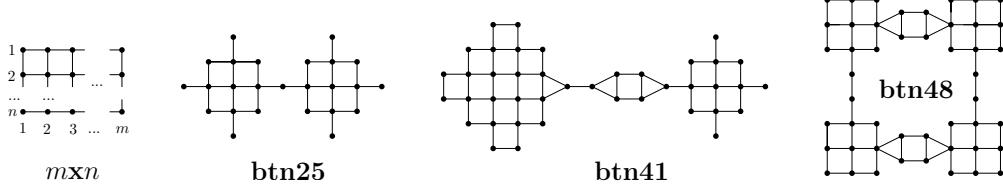


Fig. 3. Sample topologies.

Instance	Cplex 12.4				Init. Heur.			Tabu Search			
	[min.]	[Joule]	Best [s]	Proof [s]	[min.]	[Joule]	Best [s]	[min.]	[Joule]	Best [s]	*
9x3	98.2	35.1	32.4	70.9	92.8	35.0	0.5	98.2	35.1	19.2	*
btn25	68.7	32.7	7.5	11.5	67.2	31.6	0.3	68.7	32.7	11.5	
7x7	82.4	65.4	672.1	2013.6	82.4	66.2	3.7	82.4	65.4	105.8	*
12x4	83.0	66.6	319.3	1553.1	83.0	68.5	3.9	83.0	66.6	111.8	*
btn41	56.3	58.0	2.6	2.6	51.3	57.5	2.0	56.3	58.0	353.7	
btn48	70.2	69.7	155.6	202.4	54.7	73.4	5.5	70.2	69.7	108.7	*
8x8	74.2	88.4	2437.7	16.8%	74.2	90.2	7.9	74.2	88.4	58.2	*
10x10	46.2	167.6	316.1	35.6%	60.0	149.9	35.1	60.0	146.5	2493.7	*
9x3-rA	100.2	36.4	3.0	3.3	83.2	35.1	0.6	100.2	36.4	20.4	
btn25-rA	49.9	32.5	0.8	0.8	47.5	31.6	0.3	49.9	32.5	100.2	
7x7-rA	100.7	68.2	181.0	459.2	88.5	66.7	3.4	100.7	68.2	174.0	
12x4-rA	88.5	68.8	130.0	232.8	80.7	68.1	3.2	88.5	68.8	144.3	*
btn41-rA	41.1	57.6	2.5	2.5	41.1	57.6	1.9	41.1	57.6	1.9	*
btn48-rA	57.8	69.6	77.6	81.7	57.8	70.1	4.3	57.8	69.6	162.0	
8x8-rA	92.5	92.2	1638.7	3205.8	84.0	90.7	9.7	91.2	91.8	733.3	
10x10-rA	48.2	167.6	248.0	35.0%	64.3	151.6	36.0	72.5	148.2	2512.1	*

Table 1
Experimental results: $L = 3$.

Each row reports the name of the instance (see Figure 3), and three groups of columns devoted to: the model directly solved with Cplex, the initial heuristic, and the tabu search with $T = 7$. Column [min.] is the game duration in minutes, [Joule] the total consumption, Best [s] the time in seconds to find the best solution (including, for tabu search, the time for the initial solution), and Proof [s] the time in seconds to prove the optimality of the solution (or the optimality gap after one hour computation, in italicics). The first eight instances have homogeneous settings for each node and link (in particular $A_v = 9000, \forall v \in N$); the remaining instances (suffix “rA”) have the same settings but for A_v , which is uniformly distributed between 6000 and 12000.

The model is able to solve to optimality instances of up to about 30 nodes, while, from 60 nodes on, it fails in finding or proving the optimal solution. Tabu search always finds the best solution (optimal, if available), but in one instance (8x8-rA), and the value may be significantly better (see bold figures). Required running times are often lower than Cplex (see the asterisks), even thanks to the good quality of the initial heuristic solutions, and the distance

seems to grow for larger instances, in particular for the first more “regular” ones, where tabu search seems to better exploit topological symmetries.

From a network configuration point of view, it is interesting to note that, with respect to the case with a single server, the case $L = 3$ allows improving the game duration by 45% on average, at the cost of additional 2% of total consumptions. Further, from the analysis of optimal solutions, it emerges that the servers should be fairly distributed around the “center” of the network (i.e., the node chosen for $L = 1$): in fact, if all servers are too close to this center and, hence, to each other, when they are not active, they would consume energy for both backup-server duties and for message forwarding.

Preliminary results point out that the proposed model for CSMCP can be effectively used to solve small instances to optimality, while the tabu search is able to quickly propose good solutions, often the optimal ones, even for instances of up to 100 nodes. Of course, results should be validated by extensive test (which is an ongoing work) and there is room for improvement. Among others, the initial heuristic may better exploits information on instance settings (e.g., on node battery) and some suggestions emerging from the analysis of optimal servers’ location; also, the very basic tabu search may be implemented more efficiently and completed with more sophisticated moves and intensification/diversification. This is the object of ongoing research.

References

- [1] Ajmone Marsan, M., C. F. Chiasseroni, A. Nucci, G. Carello and L. De Giovanni, *Optimizing the topology of Bluetooth Wireless Personal Area Networks*, in: *Proc. of IEEE Infocom 2002*, New York, NY, USA, Jun 2002.
- [2] Glover, F. and M. Laguna, “Tabu search,” Kluwer Academic, Boston, 1997.
- [3] IBM, *ILOG CPLEX Optimization Studio*, <http://www-01.ibm.com/software/integration/optimization/cplex-optimization-studio/>.
- [4] Palazzi, C. E., A. Kaiser, N. Achir and K. Boussetta, *A Preliminary Evaluation of Backup Servers for Longer Gaming Sessions in MANETs*, in: *Proc. of DISIO 2010 - ICST SIMUTools 2010*, Torremolinos, Spain, Mar 20102.
- [5] Roccetti, M., G. Marfia and A. Semeraro, *Playing into the Wild: A Gesture-based Interface for Gaming in Public Spaces*, Journal of Visual Communication and Image Representation **23** (2012), pp. 426–440.
- [6] Wright, S. and S. Tischer, *Architectural Considerations in Online Game Services over DSL Networks*, in: *IEEE ICC’04*, Paris, France, Jun 2004.

On the Generalized Elementary Shortest Path Problem: A heuristic approach

Guerrero, W.J.^{a,b,1} Velasco, N.^{a,c} Prodhon, C.^b
Amaya, C.A.^a

^a *Pylo, School of Engineering, Universidad de los Andes
Cra 1 Este No 19A - 40, Bogotá, Colombia.*

^b *ICD-LOSI (UMR-STMR CNRS 6279), Université de Technologie de Troyes
12 rue Marie Curie, CS 42060 10004 Troyes Cedex, France*

^c *Invited Professor, ICD-LOSI (CNRS 6279), Université de Technologie de Troyes
12 rue Marie Curie, CS 42060 10004 Troyes Cedex, France*

Abstract

We introduce the generalized elementary shortest path problem (GESPP) where in addition to the features of the shortest path problem, nodes belong to predefined non-disjoint clusters. Each cluster is associated to a profit to the cost function, obtained if at least one element in the cluster appears in the path. Several applications can be considered as school bus routing, pricing problems, or telecommunication network design. Thus, depending on the case, clusters could be interpreted as groups of nodes with linking features as, for example, being easily reachable from each other, or some kind of coverage guarantee. We compare the GESPP to similar problems in the literature and we propose a two-phase heuristic algorithm for graphs including negative cycles. Tests on random instances with up to 100 nodes show an average gap of 0.3% to the best known solutions computed in 2.8s in average.

Keywords: Elementary shortest path problem, labeling algorithm.

1 Introduction

In this paper, a new variant of the elementary shortest path problem (ESPP) is formulated by considering arbitrary arc costs and profits associated to visit predefined clusters of nodes. It is introduced as the generalized elementary shortest path problem (GESPP). As a matter of fact, it reduces to the ESPP with negative weight cycles reachable from the source node, if the profits associated to visit the clusters are equal to 0. Recall that the polynomially bounded algorithms by Dijkstra [5] and Bellman-Ford [2] forbid negative weight cycles reachable from the source node. Then, the GESPP is NP-hard given that the ESPP with negative weight cycles is known to be NP-hard [3]. Further, a heuristic algorithm is proposed inspired by the exact labeling algorithm for the ESPP with resource constraints (ESPPRC) by [7].

To illustrate the GESPP, consider a network in which nodes are grouped into a set of non-disjunctive clusters Ψ . Although, there might be nodes not belonging to any cluster. The objective of the optimization problem is to find the minimum cost path from a defined source node to the sink node. The cost of the path includes the cost c_{ij} of each traversed arc (i, j) and the aggregated profits p_t for each visited cluster $C_t, \forall t \in \Psi$. To collect a particular profit from a cluster, at least one node within the cluster must be in the path. Figure 1 presents the graph for an example of the GESPP considering $n = 9$. Nodes 0 and 10 are the source and sink respectively. Consider four clusters: $C_1 = \{1, 2, 4, 5\}$, $C_2 = \{2, 3\}$, $C_3 = \{7, 8\}$, and $C_4 = \{5, 6, 8, 9\}$, associated to the profits p_1, p_2, p_3 and p_4 respectively. The path $\{0 - 2 - 4 - 7 - 10\}$ would have a cost equal to $c_{0,2} + c_{2,4} + c_{4,7} + c_{7,10} - p_1 - p_2 - p_3$.

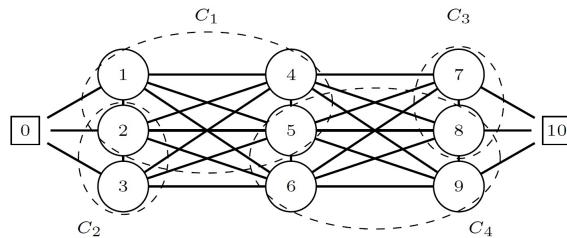


Fig. 1. Graph for the GESPP - an example

Similar problems in the literature have been studied. For example, [4] introduced the median shortest path problem (MSPP). The MSPP is a bi-objective problem trading off the distance of the path with the accessibility of the path. They measure accessibility as the total weighted travel distance

¹ Email: wi-guerr@uniandes.edu.co

that demand must traverse to reach the nearest node on the selected path. This last objective matches with the p -median facility location problem. The median cycle problem (MCP) [8] aims to design a tour to visit a subset of nodes satisfying that the sum of the distance from every unvisited node to the closest node in the tour is not greater than a predefined bound.

The GESPP and the TSP with profits [6] have both in common that a subset of nodes is visited and a notion of profit is proposed. While the TSP with profits associates a benefit to each node, the GESPP associates a profit per cluster of nodes. Still, in the TSP with profits, the model objective is to minimize the distance of a tour that visits a subset of nodes while maximizing or satisfying a minimum collected profit from each visited node. More on the TSP with profits and orienteering problems can be found at [1] and [10].

Application contexts include: 1) Urban transportation network design, to optimize the design of a new bus or metro line; 2) New rail lines design to connect two major cities while deserving smaller villages; and 3) Telecommunications network design with profits for increased reliability when interconnecting hubs. In the following: section 2 presents the GESPP mathematical formulation, section 3 explains the heuristic algorithm, denoted H^* , and its performance is studied in section 4. Conclusions are exposed in section 5.

2 Mathematical Formulation

Let the GESPP be defined over a complete, weighted and directed graph G composed by a set J of n nodes, a source node $\{0\}$ and a sink node $\{n+1\}$. Each arc in $A = \{(i, j), \forall i, j \in J \cup \{0, n+1\}\}$ in G is associated to a cost $c_{ij} \in \mathbb{R}$ (G may contain negative cycles). Also, nodes are aggregated in predefined non-disjoint clusters. Each cluster $t \in \Psi$ is associated with a profit $p_t \geq 0$ to the cost function if at least one node in t is visited. Depending on the application, clusters could be interpreted as groups of nodes with linking features, easily reachable from each other, or some kind of coverage guarantee. Let x_{ij} be a binary decision variable indicating if the arc (i, j) belongs to the path, and y_t be a binary variable equal to 1 iff cluster $t \in \Psi$ is visited at least once. Then:

$$\text{GESPP: } \min \sum_{i \in J \cup \{0\}} \sum_{j \in J \cup \{n+1\}} c_{ij} x_{ij} - \sum_{t \in \Psi} y_t p_t \quad (1)$$

Subject to:

$$\sum_{i \in J} x_{0,i} = 1 \quad (2)$$

$$\sum_{i \in J} x_{i,n+1} = 1 \quad (3)$$

$$\sum_{i \in J \cup \{0\}} x_{ij} - \sum_{i \in J \cup \{n+1\}} x_{ji} = 0, \forall j \in J \quad (4)$$

$$\sum_{i \in S} \sum_{j \in S} x_{ij} \leq |S| - 1, \forall S \subseteq J \quad (5)$$

$$\sum_{i \in S_t} \sum_{j \in J \setminus \{S_t\}} x_{ij} \geq y_t, \forall t \in \Psi \quad (6)$$

$$x_{ij} \in \{0, 1\} \quad \forall i \in J \cup \{0\}, \forall j \in J \cup \{n+1\}, y_t \in \{0, 1\} \quad \forall t \in \Psi \quad (7)$$

The objective is defined by equation (1) aiming to minimize the result of the path length cost after subtracting the cluster profits. Constraints (2), (3), and (4) are traditional to a shortest path problem. The path must start at the source node and end at the sink node as imposed by equations (2) and (3). Constraints (4) are flow conservation constraints. Subtour elimination constraints (5) are required given the potential negative cycles in G . A cluster profit is obtained iff the path visits any node belonging to the cluster as stated by constraints (6). Decision variables are binary as defined by equations (7).

3 Heuristic Procedure H*

To find the path with minimum cost value, a truncated labeling algorithm is proposed in a first phase. Each node j is associated with a set of labels representing paths from node 0 to j . Each label L'_j keeps track of all visited clusters in the set $S_{L'_j} \subseteq \Psi$, the visited nodes in the set $V_{L'_j} \subseteq J$, and the path cost $C(L'_j)$ including both, traveled cost and collected profits. One label L'_j dominates another label L''_j ($L'_j \prec L''_j$), where L'_j and L''_j represent paths from node 0 to the same node j if 1) $C(L'_j) < C(L''_j)$, $S_{L'_j} \subseteq S_{L''_j}$ implying that L'_j visits at least all the clusters visited by L''_j , and $V_{L'_j} \subseteq V_{L''_j}$ meaning L'_j visits at least all the nodes visited by L''_j ; or 2) if $C(L'_j) = C(L''_j)$, and $S_{L'_j} \not\subseteq S_{L''_j}$ implying that L'_j visits at least all the clusters visited by L''_j or $S_{L'_j} \neq S_{L''_j}$, and $V_{L'_j} \not\subseteq V_{L''_j}$. The first phase enumerates paths from 0 up to every node by keeping only non-dominated labels. Extensions for label L'_j are limited towards nodes in the set $J \cup \{n+1\} / V_{L'_j}$ to guarantee elementary paths only. It stops when all the existing labels have been extended to unvisited nodes. A

limit of K non-dominated labels is imposed per node after extending labels, in order to speed up the search. Two different rules are analyzed: Whether the K non-dominated lowest cost labels are kept or the first computed K non-dominated labels. Further, a local search procedure is performed as post-optimization with the following traditional neighborhoods:

- EXCHANGE: Modifies the position of a node in the path.
- SWAP: Interchanges the position of two nodes in the path.
- 2-Opt: Erases two arcs in the path and reconnects it with two different arcs.
- 3-Opt: Erases 3 arcs in the path and reconnects it with three different ones in the best possible way.
- INSERT: Insert an unvisited node into the path.

4 Computational Experiments

Tests for a set of 100 random instances with up to 100 nodes are run on an Intel Xeon with 2.80Ghz processor and 12 GB of RAM. As expected, for low values of K , H^* requires extensive intensification in the post-optimization phase while larger values makes H^* to perform slowly in the first phase. The best found solution (UB) is obtained whether by running H^* with different K values up to 100 or by solving the IP model using Xpress-IVE with a time limit of 4 hours per instance and replacing constraints (5) by the MTZ subtour elimination constraints [9]. A quick and simple lower bound (LB) is computed using a MIP solver if subtour elimination constraints (5) are relaxed and replaced by constraints (8) which are valid for an ESPP.

$$\sum_{i \in J \cup \{0\}} x_{i,j} \leq 1, \quad j \in J \tag{8}$$

Table 1 presents the average results for the test instance sets composed by 20 instances with $n = 20$, 30 instances with $n = 50$, and 50 instances with $n = 100$. The coordinates (x_i, y_i) of each node i are randomly generated over a grid of 100×100 together with a random value $\delta_i \sim \text{Normal}(50, 20), \forall i \in J$. Arc costs are computed as: $c_{ij} = \left[100 \cdot \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} - \delta_i/2 - \delta_j/2 \right]$. The number of clusters is generated using a uniform distribution between the range $[n, 2 \cdot n]$. Each node i belongs to cluster t with a probability of 0.5.

Tests are performed for K values from 2 to 100. The average gap between H^* and UB (gap_{UB}), between H^* and LB (gap_{LB}), and the average computation time in seconds (cpu) is reported for the most relevant K values. Super

Table 1 Computational results: (a) Truncating labels by Lowest cost rule, (b) Random start heuristic

Test Set	H* (K = 2)				H* (K = 4)				H* (K = 6)				H* (K = 20)			
	gap _U ¹ _B	gap _U ² _B	gap _L ² _B	cpu	gap _U ¹ _B	gap _U ² _B	gap _L ² _B	cpu	gap _U ¹ _B	gap _U ² _B	gap _L ² _B	cpu	gap _U ¹ _B	gap _U ² _B	gap _L ² _B	cpu
20 nodes	23.32	1.27	8.92	1.8	20.05	0.84	8.54	1.9	16.54	1.09	8.78	1.9	13.28	1.21	8.88	2.2
50 nodes	22.17	0.40	3.34	1.8	20.66	0.34	3.28	2.1	19.61	0.35	3.30	2.2	16.99	0.35	3.29	4.8
100 nodes	16.03	0.18	1.59	2.9	14.96	0.18	1.60	4.6	14.61	0.18	1.60	6.2	13.99	0.16	1.58	25.2
Average	19.33	0.46	3.58	2.4	17.69	0.36	3.49	3.3	16.50	0.41	3.54	4.1	14.74	0.43	3.55	14.5

Test Set	H* (K = 35)				H* (K = 50)				H* (K = 100)				Random Start			
	gap _U ¹ _B	gap _U ² _B	gap _L ² _B	cpu	gap _U ¹ _B	gap _U ² _B	gap _L ² _B	cpu	gap _U ¹ _B	gap _U ² _B	gap _L ² _B	cpu	gap _U ¹ _B	gap _U ² _B	gap _L ² _B	cpu
20 nodes	11.46	1.42	9.07	2.5	10.45	1.01	8.69	2.9	8.56	1.00	8.69	4.9	2.52	10.02	0.0	0.0
50 nodes	17.20	0.35	3.29	8.7	17.08	0.37	3.31	14.0	16.29	0.44	3.38	40.2	0.36	3.31	0.0	0.0
100 nodes	13.57	0.18	1.59	62.4	13.29	0.19	1.61	116.7	13.08	0.19	1.61	401.7	0.15	1.59	0.2	0.2
Average	14.24	0.48	3.60	34.3	13.86	0.41	3.53	63.1	13.14	0.43	3.56	213.9	0.69	3.79	0.1	0.1

(a) (b)

(c) Truncating labels by first computed rule, (d) IP solver results

Test Set	H* (K = 2)				H* (K = 4)				H* (K = 6)				H* (K = 20)			
	gap _U ¹ _B	gap _U ² _B	gap _L ² _B	cpu	gap _U ¹ _B	gap _U ² _B	gap _L ² _B	cpu	gap _U ¹ _B	gap _U ² _B	gap _L ² _B	cpu	gap _U ¹ _B	gap _U ² _B	gap _L ² _B	cpu
20 nodes	68.62	2.32	9.83	1.7	63.72	2.30	9.81	1.8	60.63	0.74	8.43	1.9	51.59	0.71	8.40	2.1
50 nodes	60.95	0.33	3.27	1.9	56.07	0.36	3.30	2.1	55.28	0.36	3.30	2.3	51.28	0.31	3.25	3.6
100 nodes	48.47	0.18	1.59	2.3	42.06	0.17	1.59	3.0	41.58	0.18	1.60	3.6	37.21	0.16	1.58	10.5
Average	56.24	0.65	3.74	2.1	50.60	0.65	3.75	2.5	49.50	0.35	3.47	2.8	44.31	0.32	3.45	6.8

Test Set	H* (K = 35)				H* (K = 50)				H* (K = 100)				IP Solver			
	gap _U ¹ _B	gap _U ² _B	gap _L ² _B	cpu	gap _U ¹ _B	gap _U ² _B	gap _L ² _B	cpu	gap _U ¹ _B	gap _U ² _B	gap _L ² _B	cpu	LB cpu	IP cpu	#opt	
20 nodes	45.90	2.40	9.91	2.3	43.80	2.62	10.11	2.6	39.31	3.39	10.81	4.0	1.8	2.9	20/20	
50 nodes	49.96	0.39	3.34	5.8	49.12	0.35	3.29	8.3	47.87	0.34	3.28	20.3	2.7	1.485	25/30	
100 nodes	36.40	0.17	1.58	20.0	36.14	0.17	1.58	31.1	35.66	0.16	1.58	82.9	8.2	-	0/50	
Average	42.37	0.68	3.78	12.2	41.57	0.71	3.80	18.6	40.05	0.86	3.94	48.4	5.3	826	45/100	(d)

(c) (d)

index 1 is for the solution of the first phase, and 2 for the solutions after local search. Note that increasing K improves the quality of the solution before local search. But even when $K = 100$, the first phase is about 8% larger than the optimal solution (known for the small instances with $n = 20$). The average results for cpu required to compute LB and a feasible solution using the IP solver are also presented (columns LB cpu and IP cpu) together with the number of instances solved to optimality by the solver (#opt). Note that every instance with $n = 20$ is solved to optimality in about 2.9s by the IP solver, while LB (elementary solutions allowing subtours) are computed in 1.8s with an average gap of 8%. 25 out of 30 optimal solutions are known for instances with $n = 50$ and no instance with $n = 100$ is solved to optimality. Average results for 5 executions of a random start heuristic with the local search operator are also presented. While it is fast, its performance is volatile and it is outperformed by the proposed heuristic. Nevertheless, the revealed contribution of the local search operator is important.

The rule for truncating the labels has an impact on the performance of H^* . The lowest cost rule provides better results before local search while the first computed rule provides paths with fewer visited nodes. Given that the local search does not evaluate the removal of nodes, the local search combined with the second rule and a K value sufficiently large could have better exploration of solution space. In fact, the best results are shown for $K=6$ using the first computed K non-dominated labels. A cpu of 2.8s and a gap to UB of 0.35% is the best trade-off between solution quality and cpu.

5 Conclusions

The GESPP is introduced as a variant of the elementary shortest path problem. Since the problem is not restrained to networks without negative cost cycles, the problem is computationally challenging. A two-phase heuristic algorithm, denoted H^* , is proposed. In the first phase, a truncated labeling algorithm is performed. A limit on the number of labels per node is forced to speed up the search of an initial solution. In a second phase, a local search operator is applied. Naturally, if the imposed limit in the first phase is sufficiently large, the optimal solution is guaranteed but the computational burden is higher. Experiments for different K values and different strategies for truncating labels were performed. Tests on random instances with up to 100 nodes show the best results when $K = 6$, with an average gap of 0.35% to the best known solutions computed in 2.8s. Future research involves exact methods and improved lower bounds.

Acknowledgement

This research is partially supported by Champagne-Ardenne Regional Council (France), Centro de Estudios Interdisciplinarios Básicos y Aplicados - CEIBA (Colombia) and Université de Technologie de Troyes. We thank the two anonymous referees for their valuable comments.

References

- [1] R. Baldacci, E. Bartolini, and G. Laporte. Some applications of the generalized vehicle routing problem. *Journal of the Operational Research Society*, 61(7):1072–1077, 2010.
- [2] R. Bellman. On a routing problem. *Quarterly of Applied Mathematics*, 16(1):89–90, 1958.
- [3] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to algorithms*. MIT Press, Cambridge, MA, USA, 2nd edition, 2001.
- [4] J.R. Current, C.S. Revelle, and J.L. Cohon. The median shortest path problem: A multiobjective approach to analyze cost vs. accessibility in the design of transportation networks. *Transportation Science*, 21(3):188–197, 1987.
- [5] E. W. Dijkstra. A note on two problems in connexion with graphs. *NUMERISCHE MATHEMATIK*, 1(1):269–271, 1959.
- [6] D. Feillet, P. Dejax, and M. Gendreau. Traveling salesman problems with profits. *Transportation Science*, 39(2):188–205, 2005.
- [7] D. Feillet, P. Dejax, M. Gendreau, and C. Gueguen. An exact algorithm for the elementary shortest path problem with resource constraints: Application to some vehicle routing problems. *Networks*, 44(3):216–229, 2004.
- [8] M. Labb  , G. Laporte, I. Rodr  guez-Mart  n, and J.J Salazar Gonz  lez. Locating median cycles in networks. *European Journal of Operational Research*, 160(2):457–470, 2005.
- [9] C. E. Miller, A. W. Tucker, and R. A. Zemlin. Integer programming formulation of traveling salesman problems. *J. ACM*, 7(4):326–329, 1960.
- [10] F. Tricoire, M. Romauch, K.F. Doerner, and R.F. Hartl. Heuristics for the multi-period orienteering problem with multiple time windows. *Computers & Operations Research*, 37(2):351–367, 2010.

Distributionally robust stochastic shortest path problem

Jianqiang Cheng¹

Laboratoire de Recherche en Informatique, Université Paris Sud, Orsay, France

Abdel Lisser and Marc Letournel²

Laboratoire de Recherche en Informatique, Université Paris Sud, Orsay, France

Abstract

This paper considers a stochastic version of the shortest path problem, the Distributionally Robust Stochastic Shortest Path Problem(DRSSPP) on directed graphs. In this model, the arc costs are deterministic, while each arc has a random delay. The mean vector and the second-moment matrix of the uncertain data are assumed known, but the exact information of the distribution is unknown. A penalty occurs when the given delay constraint is not satisfied. The objective is to minimize the sum of the path cost and the expected path delay penalty. As it is NP-hard, we approximate the DRSSPP with a semidefinite programming (SDP for short) problem, which is solvable in polynomial time and provides tight lower bounds.

Keywords: Stochastic programming, Shortest path, Distributionally robust, Semidefinite programming.

¹ Email: cheng@lri.fr

² Email: {lisper, marc.letournel}@lri.fr

1 Introduction

The Shortest Path (SP) problem is a well-known combinatorial optimization problem and has been extensively studied for the last decades [2,5,7]. Due to different kinds of real life uncertainties, there have been many papers presenting the Stochastic Shortest Path Problem (SSPP)[8,9,10,11]. One of the fundamental assumptions underlying SSPP is that the probability distributions for the random data are known. However, it is not often the case. In this paper, we adopt the distributional robustness approach where only part of information on random data are assumed to be known, i.e., the first and second moment of the random variables. Due to requiring a mild information on the random data, robust optimization has received a great interest during the past decades and becomes a popular technique to find bounds in optimization problems [4,6].

In this paper, we study a special Distributionally Robust Stochastic Shortest Path Problem (DRSSPP). In this model, each arc has a deterministic cost and a random delay. Furthermore, for the random data, we assume that its first and the second moment information are known. The problem has a simple recourse formulation. That means we deal with the delays of the path by introducing a penalty which occurs in the case where the delay constraint is not satisfied. The objective is to minimize the sum of the path cost and the expected path delay penalty. As the deterministic shortest path problem with delay is NP-hard [13], it follows that the DRSSPP is NP-hard by choosing all variance of the arcs equal to 0. The paper is organized as follows. In Section 2, we give the mathematical formulation of the DRSSPP and its equivalent deterministic formulation. In Section 3, a semidefinite programming problem is given to approximate the problem. Then, in Section 4, preliminary numerical results are provided to evaluate the approximation. The conclusions are given in the last section.

2 DRSSPP Formulation

Let $\mathcal{G} = (V, A)$ be a digraph with $n = |V|$ nodes and $m = |A|$ arcs. Each arc $a \in A$ has an associated cost $c(a) > 0$ as well as a random delay represented by the random variable $\tilde{\delta}(a)$. Without loss of generality, we assume that $\tilde{\delta}_1, \dots, \tilde{\delta}_m$ are m random delays. Let $\tilde{\delta} = \{\tilde{\delta}_1, \dots, \tilde{\delta}_m\}$.

When we know the exact probability distribution of $\tilde{\delta}$ which is denoted by $\tilde{\mathcal{F}}$, the *Stochastic Shortest Path Problem*(SSPP) consists in finding a directed path between two given vertices s and t such that the sum of the cost and

the expected delay cost is minimal. The delay cost is based on a penalty per time unit $d > 0$ that has to be paid whenever the total delay exceeds a given threshold $D > 0$.

Then, SSPP can be mathematically formulated as follows([3]):

$$(\text{SSPP}) \quad \min \sum_{a \in A} c(a)x_a + d \cdot \mathbb{E}_{\tilde{\mathcal{F}}} [\sum_{a \in A} \tilde{\delta}(a)x_a - D]^+ \quad (1a)$$

$$\text{s.t. } Mx = b \quad (1b)$$

$$x \in \{0, 1\}^m \quad (1c)$$

where $[\cdot]^+ = \max\{0, \cdot\}$, $\mathbb{E}[X]$ denotes the expectation of a random variable X , $M \in \mathbb{R}^{n \times |A|}$ is the *node-arc incidence matrix* (see [1]) and $b \in \mathbb{R}^n$, where all elements are 0 except the s -th and t -th elements, which are 1 and -1, respectively.

2.1 Distributionally Robust

SSPP requires a strong assumption which is the knowledge of the exact information about the distribution $\tilde{\mathcal{F}}$. This is not the case in practical situations. This naturally leads to a robust optimization which requires only a mild assumption on the probability distribution such as known supports and covariances. Correspondingly, we can write a distributionally robust SSPP as follows:

$$(\text{DRSSPP}) \quad \min \sum_{a \in A} c(a)x_a + d \cdot \sup_{\tilde{\mathcal{F}} \in \tilde{\mathcal{S}}} \mathbb{E}_{\tilde{\mathcal{F}}} [\sum_{a \in A} \tilde{\delta}(a)x_a - D]^+ \quad (2a)$$

$$\text{s.t. } Mx = b \quad (2b)$$

$$x \in \{0, 1\}^m \quad (2c)$$

The key assumptions under which the problem is analyzed are as follows:
Assumptions. (A1) There exists a deterministic Matrix B , a known vector μ and random vector δ such that

$$\tilde{\delta} = B\delta + \mu$$

(A2) The distributional uncertainty set accounts for information about the support \mathcal{S} , mean 0, and an upper bound Σ on the covariance matrix of the

random vector δ

$$\mathcal{D}(\mathcal{S}, 0, \Sigma) = \left\{ \mathcal{F} \middle| \begin{array}{l} \text{Prob}(x \in \mathcal{S}) = 1 \\ \mathbb{E}_{\mathcal{F}}[\delta] = 0 \\ \mathbb{E}_{\mathcal{F}}[\xi \xi^T] \preceq \Sigma \end{array} \right\}.$$

Theorem 2.1 *Under assumptions (A1) and (A2), the problem (2) is equivalent to the following deterministic problem*

$$\min \quad \sum_{a \in A} c(a)x_a + d \cdot (\Sigma \bullet \mathbf{Q} + t) \quad (3a)$$

$$\begin{bmatrix} t + D - \mu^T x & \frac{(\mathbf{q} - Bx)^T}{2} \\ \frac{\mathbf{q} - Bx}{2} & \mathbf{Q} \end{bmatrix} \succeq 0 \quad (3b)$$

$$\begin{bmatrix} t & \frac{\mathbf{q}^T}{2} \\ \frac{\mathbf{q}}{2} & \mathbf{Q} \end{bmatrix} \succeq 0, \quad \mathbf{Q} \succeq 0, \quad \mathbf{q} \in \mathbb{R}^m \quad (3c)$$

$$Mx = b \quad (3d)$$

$$x \in \{0, 1\}^m \quad (3e)$$

Proof. The main idea of the proof relies on applying the theory presented in [6] to transform the distributionally robust objective function into its deterministic equivalent. \square

3 Semidefinite approximation

Problem (3) is an SDP problem with binary variables. Therefore, we apply semidefinite relaxation to deal with the binary constraints. By introducing redundant constraints, we get the following SDP approximation of DRSSPP:

$$\min \sum_{a \in A} c(a)x_a + d \cdot (\Sigma \bullet \mathbf{Q} + t) \quad (4a)$$

$$\begin{bmatrix} t + D - \mu^T x & \frac{(\mathbf{q} - Bx)^T}{2} \\ \frac{\mathbf{q} - Bx}{2} & \mathbf{Q} \end{bmatrix} \succeq 0 \quad (4b)$$

$$\begin{bmatrix} t & \frac{\mathbf{q}^T}{2} \\ \frac{\mathbf{q}}{2} & \mathbf{Q} \end{bmatrix} \succeq 0 \quad (4c)$$

$$M_i^T x = b_i, i = 1, \dots, n \quad (4d)$$

$$M_i^T \mathbf{X} M_i = b_i^2, \quad X_{ii} = x_i, i = 1, \dots, n \quad (4e)$$

$$\begin{bmatrix} 1 & x^T \\ x & \mathbf{X} \end{bmatrix} \succeq 0, \quad \mathbf{Q} \succeq 0, \quad \mathbf{q} \in \mathbb{R}^m \quad (4f)$$

where M_i is the i -th row vector of the matrix M .

4 Numerical study

The SDP relaxations as well as a branch-and-bound algorithm [3] were implemented in Matlab and solved by Sedumi [12] with default parameters on an Intel(R)D @ 2.00 GHz with 4.0 GB RAM. We considered two directed graphs for our tests with $(|V|, |A|)$ equal to (21, 39) and (23, 40) respectively.

The input data for the models is randomly generated as follows. The cost c is uniformly generated from $[0, 10]$. The mean μ is uniformly generated from $[5, 10]$ interval and the covariance matrix Σ is generated by MATLAB function “gallery(‘randcorr’,n)*2”. The penalty d is 5 and D is set to the mean of the delay of the shortest path. Finally, the matrix B is uniformly generated from $[0, 5]$ interval.

To measure the quality of the results of the SDP relaxations designed hereafter by V^{SDP} , we apply a branch-and-bound method to get an optimal value designed hereafter by V^{OPT} . Numerical results are given by Table 1 and Table ??, where column one gives the name of the instances and columns two and three present the optimal value of the SDP relaxation and the corresponding CPU time respectively. Columns four and five give the optimal value of the branch-and-bound and the relative CPU time respectively. The last column gives the gap defined by $\text{Gap} = \frac{V^{OPT} - V^{SDP}}{V^{OPT}} \cdot 100\%$.

From Table 1 and Table 2, we observe that the SDP approximation is very competitive for two reasons: the first one is that the largest gap between the

Instance	V^{SDP}	CPU(s)	V^{OPT}	CPU(s)	Gap(%)
Inst1	296.25	179.38	296.25	1623.31	0.00
Inst2	283.85	180.74	286.66	1505.53	0.98
Inst3	314.37	197.95	315.04	1405.27	0.21
Inst4	320.96	191.59	322.86	1548.92	0.59
Inst5	291.55	174.44	295.82	1292.81	1.44

Table 1
Computational results of DRSSPP as $(|V|, |A|) = (21, 39)$

approximation and the optimal value is less than 1.6% for the whole instances of the two graphs. The other reason is that for all the instances of the two graphs, the maximum CPU time of SDP approximation does not exceed 260 seconds, while the maximum CPU time of the branch-and-bound is more than 3400 seconds .

5 Conclusions

In this paper, we consider a distributionally robust version of the shortest path problem, the Distributionally Robust Stochastic Shortest Path Problem(DRSSPP) on directed graphs. In this model, the arc costs are deterministic, while each arc has a random delay. The mean vector and the second-moment matrix of the uncertain data are assumed known, but the exact information of the distribution is unknown. A penalty occurs when the given delay constraint is not satisfied. The objective is to minimize the sum of the path cost and the expected path delay penalty. As this problem is NP-hard, we approximate the DRSSPP by a semidefinite programming relaxation, which is tractable in polynomial time and provide tight lower bounds. Finally, ex-

Instance	V^{SDP}	CPU(s)	V^{OPT}	CPU(s)	Gap(%)
Inst1	288.92	237.59	289.34	1693.99	0.15
Inst2	256.57	209.71	260.63	3445.97	1.56
Inst3	240.01	232.20	240.59	2550.83	0.24
Inst4	293.17	255.03	294.13	1781.36	0.33
Inst5	284.55	229.31	286.30	1820.68	0.61

Table 2
 Computational results of DRSSPP as $(|V|, |A|) = (23, 40)$
 samples randomly generated highlight the efficiency of our approach.

References

- [1] Ahuja, R. K., T. L. Magnanti and J. B. Orlin, “Network Flows: Theory, Algorithms and Applications”, Prentice Hall, New Jersey, 1993.
- [2] Bellman, R. E., *On a routing problem*, Quarterly of Applied Mathematic 16 (1958), 87–90.
- [3] Cheng, J., S. Kosuch and A. Lisser, *Stochastic Shortest Path Problem with Uncertain Delays*, ICORES (2012), 256–264.
- [4] Cheng, J., E. Delage and A. Lisser, *Distributionally robust stochastic knapsack problem authors*, Working paper, No.1556 (2012).
- [5] Dijkstra, E. W., *A note on two problems in connection with graphs*, Numerische Mathematik 1 (1959), 269–271.
- [6] Delage, E. and Y. Ye, *Distributionally Robust Optimization Under Moment Uncertainty with Application to Data-Driven Problems*, Operations Research 58(3) (2010), 595–612.

- [7] Ford, L. R., and D. R. Fulkerson, “Flows in Networks”, Princeton University Press, Princeton, 1962.
- [8] Hutson, K. R., and D. R. Shierb, *Extended dominance and a stochastic shortest path problem*, Computers and Operations Research 36 (2009), 584–596.
- [9] Mirchandani, P. B., and H. Soroush, *Optimal paths in probabilistic networks: a case with temporary preferences*, Computers and Operations Research 12 (1985), 365–81.
- [10] Murthy, I., and S. Sarkar, *A relaxation-based pruning technique for a class of stochastic shortest path problems*, Transportation Science 30 (1996), 220–236.
- [11] Ohtsubo, Y., *Minimization risk models in stochastic shortest path problems*, Mathematical Methods of Operations Research 57 (2003), 79–88.
- [12] Sturm, J. F., *Using SeDuMi 1.02, a Matlab toolbox for optimization over symmetric cones*, Optimization Methods and Software 11-12 (1999), 625–653.
- [13] Verweij, B., S. Ahmed, A. J. Kleywegt, G. Nemhauser and A. Shapiro, *The sample average approximation method applied to stochastic routing problems: A computational study*, Computational Optimization and Applications 24(2-3) (2003), 289–333.

On the approximability of two capacitated vehicle routing problems

Adrian Bock¹

EPFL, Lausanne, Switzerland

Laura Sanit  ²

University of Waterloo, Canada

Abstract

Vehicle routing is an important and active research topic in computer science and operations research. In this paper, we give some approximation results for two well-known capacitated vehicle routing problems.

Our first result concerns the Capacitated Orienteering problem in Euclidean graphs. We are here given an Euclidean graph G , where each node has a profit value and a demand value, starting and end nodes s, t , a length bound D and a capacity bound C . The goal is to find an $s-t$ -path of length at most D that collects maximum profit from nodes whose total demand does not exceed the capacity bound C . We give a PTAS for this problem, extending the corresponding known result given by Chen and Har-Peled [7] for the uncapacitated version.

Our second result concerns the School Bus problem with regret minimization, where we are given a general metric graph, and the task is to design the routes for a given set of buses of limited capacity to transport a set of children to a school, while minimizing a certain regret threshold. Under the standard hypothesis $P \neq NP$, we show that this problem cannot be approximated.

Keywords: Vehicle routing, Approximation algorithms, Orienteering.

1 Introduction

A fundamental and important topic in computer science and operations research is *Vehicle routing*. In fact, vehicle routing problems frequently arise in areas such as logistics, distribution systems, and public transportation (we refer to the survey in [12]). In this paper, we give some approximation results for two known *capacitated* vehicle routing problems.

Our first result concerns the capacitated version of the Orienteering Problem. In the Orienteering Problem (OP), we are given a complete graph $G = (V, E)$ with metric distances $d : V \times V \rightarrow \mathbb{N}$, starting and end nodes $s, t \in V$, node profits $\pi : V \rightarrow \mathbb{N}$ and a length bound D . The goal is to find an $s-t$ path P , subject to the length constraint $\sum_{\{u,v\} \in E(P)} d(u, v) \leq D$, that collects maximum profit $\sum_{v \in V(P)} \pi(v)$ from the nodes it covers. Orienteering is widely studied in the literature in terms of approximation, exact, and heuristic algorithms (see e.g. [6, 7, 14]). Indeed, there are many algorithms for various vehicle routing problems that rely on applying algorithms for Orienteering as a subroutine (e.g. Deadline-TSP [2], Distance-constrained vehicle routing [10], School Bus problem [4]). Orienteering is NP-hard, but it admits a $(2 + \varepsilon)$ -approximation algorithm for general metrics [6], a Fully Polynomial Time Approximation Scheme (FPTAS) for tree metrics [5], and a Polynomial Time Approximation Scheme (PTAS) for Euclidean metrics [7].³

The *capacitated* Orienteering problem (COP) is a natural generalization of Orienteering introduced in [5], in which the selected path has to satisfy a second budget constraint besides the length bound. Namely, here we are also given node demands $r : V \rightarrow \mathbb{N}$ and a capacity bound $C \in \mathbb{N}$, and then we look for an $s-t$ -path P that maximizes $\sum_{v \in V(P)} \pi(v)$, while satisfying both constraints $\sum_{\{u,v\} \in E(P)} d(u, v) \leq D$ and $\sum_{v \in V(P)} r(v) \leq C$. There is a number of capacitated vehicle routing problems where the capacitated orienteering problem appears as subroutine (e.g. Capacitated Team Orienteering [1] and related variants). In [5], it is shown that COP admits a constant approximation for general metrics, and a PTAS for tree metrics. In this paper, we give a PTAS for Euclidean metrics, building upon the corresponding result given by [7] for the uncapacitated version.

Our second result concerns the School Bus Problem with Regret Minimization

¹ Email: adrianaloysius.bock@epfl.ch

² Email: lisanita@uwaterloo.ca

³ In fact, the result in [7] is given for unit profit values, but it can be generalized to arbitrary profits by simply scaling the profits (cf. [3]).

tion (SBP-R). SBP-R is an NP-hard capacitated vehicle routing problem in which the routes for a given set of buses of limited capacity are to be designed to transport a set of children to a school. The goal is to minimize the maximum *regret* of a child, that is the difference between the distance travelled on the bus and the shortest distance from the child’s home to the school (a formal definition is given in Section 3). For the SBP-R, heuristic methods for practical applications have been studied (see the survey of [11]), while [4] gives a $O(1)$ -approximation for tree metrics, assuming infinite capacity. Indeed, for general metric graphs there is no literature concerning approximability and inapproximability results.

In this paper, we show that unless $P = NP$, SBP-R is inapproximable, since it is NP-hard to check whether an instance of SBP-R has regret 0 or strictly positive.

2 A PTAS for capacitated Orienteering in the plane

The aim of this section is to give a PTAS for the capacitated Orienteering, extending the result by [7] who considered the case $\pi \equiv 1, r \equiv 1$, and $C = |V|$. We recall here that a PTAS for a maximization problem is an algorithm that takes as input an instance of the problem and a *fixed* $\bar{\varepsilon} > 0$, and outputs in polynomial time a solution of value at least $(1 - \bar{\varepsilon})$ times the value of an optimal solution.

In the remainder, $d(v, w)$ refers to the Euclidean distance between the nodes $v, w \in V$. We denote $n = |V|$ and we will focus in the following on points in the plane, i.e. each point is characterized by an x - and a y -coordinate.

2.1 The PTAS for uncapacitated Orienteering

In this section we describe the algorithm of Chen and Har-Peled [7]. Their result is based on a dynamic programming approach closely related to the one by Mitchell for k -TSP [9]. In particular, their algorithm relies on a dynamic program for the following k -path problem: Given an Euclidean graph G , nodes $s, t \in V$ and an integer $k \leq n$, find an $s - t$ path of minimum length visiting at least k nodes.

We sketch the dynamic program in more detail, since it will be crucial for our result. Assume that all points are contained in an axis-parallel square \mathcal{R} . We use the notion of a *window* for a closed axis-parallel rectangle $w \subseteq \mathcal{R}$. With this notion and given a fixed $\varepsilon > 0$, one can define table entries for the dynamic program, each characterized by:

- (i) a (minimal) window w that contains at least one point of V , with its boundaries determined by (up to) four points of V ,
- (ii) integer $h \in \{0, 1, \dots, |V|\}$, indicating the number of nodes that should be visited within w ,
- (iii) boundary information specifying at most $m = O(\frac{1}{\varepsilon})$ crossing segments (each determined by a pair of points of V , one inside or on the boundary of w , another outside w) for each of the four sides of the boundary of w ,
- (iv) connectivity constraints indicating which pairs of crossing segments are required to be connected within w .

Each table entry stores a set of short paths inside w meeting both the boundary and the connectivity constraints, visiting at least h nodes. The stored information can be combined together in a recursive manner, by subdividing each window into smaller windows by cutting w along a horizontal or vertical line (we refer to [7,9] for details). This procedure gives a PTAS for the k -path problem.

How is the k -path problem related to Orienteering? We can find a feasible solution for an Orienteering instance (G, s, t, D) as follows: starting from $k = 0$ up to $k = n$, apply the dynamic program above for the k -path problem on G and starting and ending nodes s and t . Let \tilde{P} be the best path (in terms of number of visited points) with length $\leq D$ found with this procedure. \tilde{P} is then a feasible path for the given Orienteering instance. If k_{opt} is the number of points visited by an optimal solution, the authors in [7] show that

Lemma 2.1 [7] *The path \tilde{P} visits at least $(1 - \varepsilon)k_{opt}$ nodes.*

2.2 Our modifications

Similarly to the previous algorithm for uncapacitated Orienteering, our result relies on a dynamic program for a more general capacitated *quota-path* problem that is defined as follows: We are given an Euclidean graph G , nodes $s, t \in V$, node profits $\pi : V \rightarrow \mathbb{N}$, demands $r : V \rightarrow \mathbb{N}$, a capacity bound C and an integer $Q \leq \sum_{v \in V} \pi(v)$. Moreover, we are also given a special set of vertices $S \subset V$ with $|S| = O(1)$. We want to find an $s - t$ path of minimum length that contains S (among possibly some other nodes), and collects at least Q profit by serving a total demand $\leq C$. Clearly, the capacitated *quota-path* problem is a generalization of the k -path problem that can be obtained by setting $\pi \equiv 1$, $r \equiv 1$, $C = |V|$ and $S = \emptyset$.

We extend the dynamic program of the previous subsection as follows. First, instead of remembering an integer indicating the number of nodes that

should be visited within a window w , we should rather consider in (ii) a profit threshold to be collected, that is, we should consider an integer h in the range $\{0, 1, \dots, Q\}$.

Second, we need to handle the demands and the capacity bound. To this end, we additionally specify in a table entry of the dynamic program

- (v) an integer j in the range $\{0, \dots, C\}$ indicating that the total demand of the nodes covered inside the window w should not exceed j .

Furthermore, we enforce that the dynamic program considers only solutions for a window w that visit all nodes of S that fall into w .

We can show that the stored information can be combined in a recursive manner, following exactly the same steps in [7,9], to compute an $s - t$ path visiting the nodes in S , that collects an amount of profit $\geq Q$ from a subset of nodes of total demand $\leq C$. We can therefore prove that

Lemma 2.2 *There is a PTAS for the quota-path problem if the capacity C and the quota Q are polynomially bounded in the input size.*

How can we use the quota-path problem to solve capacitated Orienteering?

In general, the profits might not be polynomially bounded in the input size. However, by preprocessing our instance using standard scaling and rounding techniques, we can reduce to the case where the profits are polynomially bounded in the input size. Since the argument is pretty standard, we omit the details (see e.g. [15]), and assume this is the case.

Again, the capacity bound C might not polynomially bounded in the input size. This would result in a dynamic program whose run time is not polynomial in the input size. Differently from the profit values, we cannot simply apply standard scaling and rounding techniques. To overcome this difficulty, we rely on a feasibilization method introduced by Grandoni and Zenklusen [8]. This is the reason why we have to introduce the special set S of constant size.

Given a capacitated Orienteering instance (G, s, t, π, r, C, D) and a fixed $\bar{\varepsilon} > 0$ small enough, we do the following steps.

First, we set $\varepsilon := \frac{\bar{\varepsilon}}{2}$ and we guess the $1/\varepsilon$ nodes with largest profit in the optimal solution by enumerating all possibilities in time $O(n^{\frac{1}{\varepsilon}})$. This yields the set S . Knowing S , we construct a modified instance $(G, s, t, \pi', r', C', D)$, as follows. For the profits π' , we first set to zero the profit values of all nodes that are not in S and have a profit $> \min_{v \in S} \pi(v)$. The profits of the other nodes remain unchanged. The demands and capacity bound are now rounded and scaled as follows. Let $\tilde{C} := C - \sum_{v \in S} r(v)$ and $c_{max} := \max_{v \notin S: r(v) \leq \tilde{C}} r(v)$. For $K := \frac{\varepsilon}{n} c_{max}$, we set $C' := \lfloor (1 - \varepsilon) \frac{\tilde{C}}{K} \rfloor$ and $r'(v) = 0$ for $v \in S$ and

$r'(v) = \lfloor \frac{r(v)}{K} \rfloor$ otherwise. For all values of Q from 0 to $\sum_{v \in V} \pi'(v)$, we apply the dynamic program above for the capacitated quota-path problem on the instance defined by $(G, s, t, \pi', r', C', Q, S')$ for each subset $S' \subseteq S$. We output the path P' of length $\leq D$ that collects the maximum amount of profit among all feasible paths found with this procedure. Note that C' is polynomially bounded in the input size (and so is Q , by assumption) and therefore the running time of the dynamic program is polynomial. The crucial lemma is:

Lemma 2.3 *The path P' is feasible for (G, s, t, π, r, C, D) and collects profit $\geq (1 - \bar{\varepsilon})\pi(P^*)$, where P^* is the optimal solution of the instance (G, s, t, π, r, C, D) .*

Sketch of the proof. First of all, we observe that the path P' is feasible for the instance (G, s, t, π, r, C, D) , since:

$$\begin{aligned} r(P') &\leq r(S) + r(P' \setminus S) \leq r(S) + Kr'(P \setminus S) + Kn \\ &\leq r(S) + KC' + \varepsilon c_{max} \leq r(S) + (1 - \varepsilon)(C - r(S)) + \varepsilon c_{max} \leq C \end{aligned}$$

Then, we argue the bound about the profit threshold by combining the analysis of [7] and the technique of [8].

In more details, we first show that the optimum path P^* can be modified as to obtain a path P that is a feasible solution for the modified capacitated orienteering instance $(G, s, t, \pi', r', C', D)$ and still visits all nodes in S , without losing too much in term of profit: namely, $\pi(P) \geq (1 - \varepsilon)\pi(P^*)$. This can be done as in [8], by applying a greedy discarding strategy similar to the greedy strategy for the Knapsack problem.

Second, we prove that the path P' output by our algorithm has a profit value $\pi(P') \geq (1 - \varepsilon)\pi(P)$. To this end, we use a similar analysis as in [7] for Lemma 2.1 to show that, for the quota value $(1 - \varepsilon)\pi(P)$, the dynamic program will actually find a solution of length $\leq D$ and thus a feasible path to our original instance (G, s, t, π, r, C, D) . Eventually, we obtain $\pi(P') \geq (1 - \varepsilon)^2\pi(P^*) \geq (1 - \bar{\varepsilon})\pi(P^*)$. \square

Putting all together, we get

Theorem 2.4 *There is a PTAS for Capacitated Orienteering in the plane.*

3 Inapproximability of the School Bus Problem

In the SBP-R, we are given a complete graph $G = (V, E)$ with general metric distances $d : V \times V \rightarrow \mathbb{N}$, a node s representing the school, and a set $W \subseteq V$ representing the houses of children. Additionally, we are given a bus capacity

$C \in \mathbb{N}$, and a number $N \in \mathbb{N}$ of available buses. We want to select N paths ending at s (that correspond to bus routes) such that each child is covered by at least one path and the total number of children covered by each path is at most C . The goal is to minimize the *regret* value R , defined as follows.

For a given set of paths covering all children, denote by P^v the path covering a child v and by $d^{P^v}(v, s)$ the distance between v and s along the path P^v . Then $R := \max_{v \in W} \{d^{P^v}(v, s) - d(v, s)\}$.

Next theorem gives an inapproximability result for the SBP-R.

Theorem 3.1 *It is NP-hard to distinguish whether an instance of SBP-R has optimum regret value 0 or strictly positive.*

Sketch of the proof. The reduction is from a variant of the satisfiability problem, namely (3, 4)-SAT. Formally, we are given n variables x_1, \dots, x_n and m clauses c_1, \dots, c_m , where each variable appears in at most 4 clauses and each clause contains exactly 3 literals. The task of deciding if a truth assignment exists that satisfies all clauses is NP-complete [13].

The main idea of the reduction is to construct an instance of SBP-R where every node in W can be covered via a path that is a shortest path from its location to the school (and therefore we can have a solution with maximum regret 0) if and only if there is a satisfying truth assignment for (3, 4)-SAT.

In our construction, the capacity bound C will play a crucial role: we ensure that $N \cdot C = |W|$, and therefore each bus must pick exactly C children. This implies that the problem reduces to understanding how to split the children among the buses. In particular, we will introduce a set of children for each clause, and two sets of children for each variable, namely a “true” set and a “false” set. A bus route from the “true” set via a clause set to the school for example corresponds to setting the variable to true to satisfy the clause.

By carefully placing the children, we can ensure that, in order to have regret zero, the set of children corresponding to a clause must be picked by a bus together with one set of children corresponding to one of its literals. Furthermore, by introducing a special gadget graph, we can ensure that for every variable, either the “true” set or the “false” set can be picked together with sets corresponding to clauses, but not both. This implies that each child can be picked by a bus that never deviates from the shortest path to the school if and only if there is a truth assignment satisfying the given (3, 4)-SAT formula. \square

We remark here that the above reduction does not hold in case of infinite capacity. Indeed, obtaining a non-trivial approximation for the uncapacitated version of the SBP-R in general metric graphs is an interesting open question.

References

- [1] Archetti, C., Feillet, D., Hertz, A., and M.G. Speranza, *The capacitated team orienteering and profitable tour problem*. Journal of the Operational Research Society 60, pp. 831–842, 2009.
- [2] Bansal, N., Blum, A., Chawla, S., and A. Meyerson, *Approximation Algorithms for Deadline-TSP and Vehicle Routing with Time Windows*. Proc. of STOC, pp. 166–174, 2004.
- [3] Blum, A., Chawla, S., Karger, D. R., Lane, T., Meyerson, A., and M. Minkoff, *Approximation Algorithms for Orienteering and Discounted-Reward TSP*. SIAM Journal on Computing, vol. 37, no. 2, pp. 653–670, 2007.
- [4] Bock, A., Grant, E., Könemann, J., and L. Sanità, *The School Bus Problem on Trees*. Proc. of ISAAC, pp. 10–19, 2011.
- [5] Bock, A., and L. Sanità, *Capacitated Orienteering*. submitted, 2012.
- [6] Chekuri, C., Korula, N., and M. Pal, *Improved Algorithms for Orienteering and Related Problems*. Proc. of SODA, pp. 661–670, 2008.
- [7] Chen, K., and S. Har-Peled, *The Euclidean orienteering problem revisited*. SIAM Journal on Computing, 2007.
- [8] Grandoni, F., and R. Zenklusen, *Approximation Schemes for Multi-Budgeted Independence Systems*. Proc. of ESA, pp. 536–548, 2010.
- [9] J.S.B. Mitchell, *Guillotine subdivisions approximate polygonal subdivisions: A simple polynomial-time approximation scheme for geometric TSP, k-MST, and related problems*. SIAM Journal of Computing, vol. 28, pp. 1298–1309, 1999.
- [10] Nagarajan, V., and R. Ravi, *Approximation Algorithms for Distance Constrained Vehicle Routing Problems*. Tepper School of Business, Carnegie Mellon University, Pittsburgh 2008.
- [11] Park, J., and , B.-I.Kim, *The school bus routing problem: A review*. European Journal of Operational Research 202, 311–319, 2010.
- [12] Toth, P., and D. Vigo, *The Vehicle Routing Problem*. SIAM, 2001.
- [13] C. A. Tovey, *A simplified np-complete satisfiability problem*. Discrete Applied Mathematics, 8(1): 85–89, 1984.
- [14] Vansteenwegen, P., Souffriau, W., and D. Van Oudheusden, *The orienteering problem: A survey*. European Journal of Operational Research 209, 1–10, 2011.
- [15] V. V. Vazirani, *Approximation Algorithms*. Springer-Verlag, 2001.

Exact solution for branch vertices constrained spanning problems

Merabet Massinissa¹ Sylvain Durand² Miklos Molnar³

*LIRMM
Montpellier 2 university
Montpellier, France*

Abstract

Given a connected graph G , a vertex v of G is said to be a branch vertex if its degree is strictly greater than 2. The Minimum Branch Vertices Spanning Tree problem (MBVST) consists in finding a spanning tree of G with the minimum number of branch vertices. This problem has been well studied in the literature and has several applications specially for routing in optical networks. However, this kind of applications do not explicitly impose a sub-graph as solution. A more flexible structure called hierarchy is proposed. Hierarchy, which can be seen as a generalization of trees, is defined as a homomorphism of a tree in a graph. Since minimizing the number of branch vertices in a hierarchy does not make sense, we propose to search the minimum cost spanning hierarchy such that the number of branch vertices is less than or equal to an integer R . We introduce the Branch Vertices Constrained Minimum Spanning Hierarchy (BVCMSH) problem which is NP-hard. The Integer Linear Program (ILP) formulation of this new problem is given. To evaluate the difference of cost between trees and hierarchies, we confront the BVCMSH problem to the Branch Vertices Constrained Minimum Spanning Tree (BVCMST) problem by comparing its exact solutions. It appears from this comparison that when $R \leq 2$, the hierarchies improve the average cost from more than 8% when $|V_G| = 20$ and from more than 12% when $|V_G| = 30$.

Keywords: Spanning tree, homomorphism, branch vertices constraint.

¹ Email: merabet@lirmm.fr

² Email: sylvain.durand@lirmm.fr

³ Email: molnar@lirmm.fr

1 Introduction

Multicast routing applied in optical networks provide several research problems on spanning tree. In optical networks, the ability of splitting the light signal is limited. Given a connected graph G , a vertex v of G is said to be a branch vertex if its degree is strictly greater than 2. The Minimum Branch Vertices Spanning Tree problem (MBVST) consists in finding a spanning tree of G with the minimum number of branch vertices [CGI09]. All researches on branch vertices constrained spanning problems are based on spanning trees, Conversely the routing do not explicitly impose a sub-graph as solution. A more flexible structure is proposed in [Mol08]. In contrast with trees, this structure (called hierarchy) is not a sub-graph but a homomorphism of a tree in a graph. Surprisingly, in connected graphs, there always exists a spanning hierarchy without branch vertices.

In this paper we introduce the Branch Vertices Constrained Minimum Spanning Hierarchy (BVCMSH) problem which consists in searching the minimum cost spanning hierarchy such that the number of branch vertices is less than or equal to an integer R . To evaluate the difference of cost between trees and hierarchies, we confront the BVCMSH problem to the Branch Vertices Constrained Minimum Spanning Tree (BVCMST) problem. As we will show, the interest of the hierarchy concept to solve these problems is obvious. Often the branch vertices constraints exclude spanning tree to be solution whereas the coverage is possible with a hierarchy. Moreover, the spanning hierarchy always achieves lower cost than the spanning tree's cost since a spanning tree is a special hierarchy. As proved in [GHSV02], it is NP-hard to find a branch vertices constrained spanning tree. We prove in section 2 that BVCMSH is also NP-hard for any value R . In order to compare the cost of solutions, we compute exact solution for both BVCMST and BVCMSH based on ILP formulation. The improvement provided by the use of hierarchy is significant (more than 12% when the graph has more than 30 vertices among which less than 3 are allowed to be branch vertices).

The rest of the paper is organized as follows. First, the branch BVCMSH problem is formulated in Section 2. In Section 3, ILP formulation is developed to compute the optimal hierarchy. Simulations are done in Section 4 to compare optimal hierarchies and optimal trees. Finally the paper is concluded in Section 5.

2 Problem formulation

A hierarchy is not always a sub-graph. It is a graph related structure obtained by a homomorphism of a tree in a graph. Remember that in graphs, a homo-

morphism can be defined as follows. Let $Q = (W, F)$ and $G = (V, E)$ be two (undirected) graphs. An application $h : W \rightarrow V$ mapping a vertex in V to each vertex in W is a homomorphism if the mapping preserves the adjacency: $(u, v) \in F$ implies $(h(u), h(v)) \in E$ [HZ94]. If Q is a connected graph without cycle (a tree) then the triple (Q, h, G) defines a *hierarchy in G* .

Figure 1 shows an example of hierarchy. Each vertex of the tree Q is associated with a unique vertex of the graph G . But, some vertices of G are mapped to several vertices in Q . A vertex in Q can be labelled by the vertex in G with which it is associated. To distinguish the occurrences related to a same vertex v in G , we will use the labels v^1, v^2, \dots, v^k if needed. If the application h is injective, then the hierarchy corresponds to a tree in G . Let u be a vertex in Q such that the mapping associates $v \in V_G$ ($h(u) = v$) to this vertex in the hierarchy. The degree $d_H(v^i)$ of the vertex occurrence v^i in H is : $d_H(v^i) = d_Q(v^i)$ where $d_Q(v^i)$ gives the degree of v^i in Q .

Since the spanning structure for routing must be connected but is not necessary a sub-graph, the branch vertices constrained minimum spanning hierarchy problem is defined as follows. Let $G = (V, E, c)$ be the weighted graph of the network with positive edge costs $c(e), \forall e \in E$.

Definition 2.1 Given a weighted graph $G = (V, E, c)$ and an integer R , the BVCMSH problem consists in finding a minimum cost spanning hierarchy $H = (Q, h, G)$ of G such that the number of branch vertices in H is less than or equal to R .

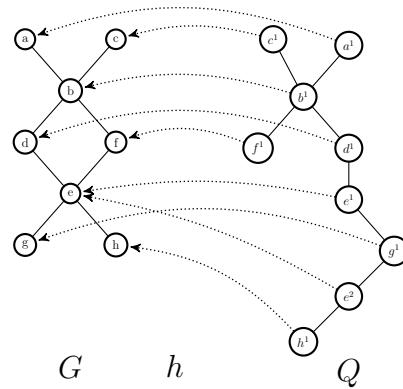


Figure 1. Mapping of vertices for a hierarchy

Figure 1 illustrates how a hierarchy can satisfy the branch vertices constraint. In all connected spanning trees of G both b and e are branch vertices

(degree strictly greater than 2). However the number of branch vertices in H is equal to 1 because of duplication of the vertex e . So there is no feasible spanning tree of G for branch vertices constrained spanning problem with $R = 1$ but there is at least one feasible hierarchy in G . There is even a hierarchy without branch vertex since any traversal of G is a hierarchy.

For both tree and hierarchy structures, finding an optimal branch vertices constraint spanning structure is NP-hard for any value of R . When $R = 0$ and all edge costs are equals, solving the problem allows to find a Hamiltonian path if it exists. When $R \geq 1$, let $G = (V, E)$ be a graph and $v \in V$ be a vertex of G (randomly chosen). Make $2 * R$ copies of G , all edges having cost 1 in these copies. Build a chain of R vertices connected by edges of cost 1. Each of these vertices is connected to 2 copies of G by adding an edge of cost K (with $K > R * |E|$) to v . Let G' be the graph obtained. In any optimal solution of BVCMSH with a bound of R , all the vertices of the chain are branch vertices (otherwise, the edges of cost K would be used more than once). Thus, there is a solution of cost $(R - 1) + 2 * (|V| - 1)$ if and only if there is an Hamiltonian path starting from v in G .

3 ILP formulation of the BVCMSH problem

In this section, an Integer Linear Program is solved to find the branch vertices constrained optimal hierarchy. In our model the connectivity is preserved by flow formulation. Thus, the initial graph has to be transformed in a symmetric oriented graph. Furthermore, a flow can transit more than once in each direction of an arc. Then each arc has to be duplicated $|V_G| - 1$ times (Figure 2) since $|V_G| - 1$ is the maximal number of times that an edge can be used in a hierarchy. This model allows to respect the branch vertices constraint. We introduce for each vertex $m \in V_G$ a binary variable Y_m equal to 1 if m is a branch vertex and 0 otherwise. Obviously, solving our linear program does not give a solution for the BVCMSH problem since the relation must be done between edges duplication and vertices duplication. A polynomial step is added to construct the optimal hierarchy.

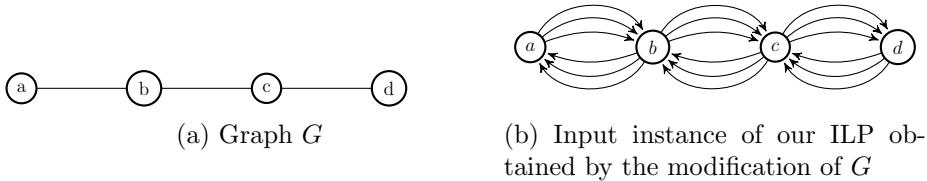


Figure 2. Our transformation of graph to feasible input of the ILP

3.1 ILP Formulation

In the following the linear program is presented.

Network parameters:

- $In(m)$: The set of vertices which have an outgoing arc leading to node m .
- $Out(m)$: The set of vertices which can be reached from m .
- $C(m, n)$: The cost of the arc linking node m to node n . All duplications of this link have the same cost.
- β : Global upper bound on the number of duplications of each vertex (equals to $|V_G| - 1$).

ILP variables:

- $L_i(m, n)$: Binary variable. Equals to 1 if the occurrence i of the arc (m, n) is in the output graph, 0 otherwise.
- $F_i(m, n)$: Commodity flow variable. Denotes the quantity of flow transiting on the occurrence i of the arc (m, n) .
- $Y(m)$: Binary variable. Equals to 1 if the vertex m is a branch vertex, 0 otherwise.

The objective of our problem is to minimize the total cost of edges belonging the hierarchy structure:

$$\text{Minimize} : \sum_{m \in V_G} \sum_{n \in Out(m)} \sum_{i=1}^{\beta} L_i(m, n) * C(m, n) \quad (1)$$

This objective function is subject to a set of constraints.

Degree constraints:

$$\sum_{n \in Out(s)} \sum_{i=1}^{\beta} L_i(s, n) - \sum_{n \in In(s)} \sum_{i=1}^{\beta} L_i(n, s) - 2 \leq (|V_G| - 1) * Y(s) \quad (2)$$

$$\sum_{n \in Out(m)} \sum_{i=1}^{\beta} L_i(m, n) - \sum_{n \in In(m)} \sum_{i=1}^{\beta} L_i(n, m) \leq (|V_G| - 1) * Y(m) \quad \forall m, n \in V \setminus \{s\} \quad (3)$$

Constraints (2) and (3) ensure that, if $Y(m)$ is equal to 0 then for each vertex except the source, the number of successors is at most equal to the number of predecessors. Otherwise, the number of successors is as large as necessary (bounded by $|V_G| - 1$).

$$\sum_{m \in V_G} Y(m) \leq R \quad (4)$$

Constraint (4) ensures the respect of the branch vertices bound R .

$$L_i(m, n) \leq 1 - Y(m), \quad \forall m \in V, \quad \forall n \in Out(m), \quad \forall i > 1 \quad (5)$$

Constraint (5) ensures that if a vertex m is a branch vertex then this vertex is duplicated only once.

Connectivity constraints:

In order to guarantee the connectivity of the output graph, we have introduced some flow constraints adapted to the specificities of degree constrained hierarchies.

$$\begin{aligned} \sum_{i=1}^{\beta} \sum_{n \in Out(s)} F_i(s, n) &= \sum_{i=1}^{\beta} \sum_{n \in In(s)} F_i(n, s) + |V| - 1 & (6) \\ \sum_{i=1}^{\beta} \sum_{n \in In(m)} F_i(n, m) &= \sum_{i=1}^{\beta} \sum_{n \in Out(m)} F_i(m, n) + 1 \quad \forall m \in V - \{s\} & (7) \end{aligned}$$

Constraints (6) and (7) ensures that each vertex except the source "consumes" one and only one unit of flow. This constraint also guarantees that each vertex is reachable from the source s . the flow emitted by the source is equal to $|V| - 1$.

$$F_i(n, m) \geq L_i(m, n) \quad \forall e = (m, n) \in E \quad (8)$$

$$F_i(m, n) \leq (|V| - 1) * L_i(m, n) \quad \forall e = (m, n) \in E \quad (9)$$

Constraints (8) and (9) allow each arc to carry non-zero flow if and only if it is used in the output graph. The value of this flow should not exceed the total flow emitted by the source.

4 Experimentation

In order to demonstrate the advantage of the proposed hierarchy structure, simulation is conducted to compare it with the spanning tree structure. ILP formulations are implemented in C with GLPK package [Mak09] by using random graphs generated with NetGen [KNS74].

We consider two values for number of vertices of random graph: $|V| \in \{20, 30\}$. The density value $d = |E|/|V|$ is fixed to 2. Graphs with this density are considered as sparse graphs. Sparse graphs are more accurately modelling real applications in networks. Thirty feasible instances for the BVCMST with random edge costs are generated for each value of $|V|$. Then ILP are

solved to search the optimal branch vertices constrained spanning tree and the optimal branch vertices constrained spanning spanning hierarchy with $R \in \{1, 2, 3, 4, 5\}$. Since it is the first exact method for solving the BVCMSH problem, we do not focus on the time complexity of our method. Our goal is mainly to evaluate the improvement obtained by using a hierarchy instead of a tree. Nevertheless, the average running time to solve the BVCMSH problem is reasonable (about 14 minutes for $|V_G| = 20$ using a 2.20GHz Intel(R) Core(TM) i3 with 4GB of RAM running the Linux operating system).

$ V_G = 20$					
R	1	2	3	4	5
Average cost of BVCMST	7711.31	7003.42	6544.86	6280.93	6084.01
Average cost of BVCMSH	6787.38	6423.11	6215.03	6120.51	6084.01
Improvement	11.98%	8.28%	5.03%	2.55%	1.26%
$ V_G = 30$					
R	1	2	3	4	5
Average cost of BVCMST	13075.17	11041.47	10045.07	9500.05	9214.98
Average cost of BVCMSH	10537.10	9645.13	9005.65	8900.74	8850.68
Improvement	19.41%	12.64%	10.35%	6.31%	3.95%

Table 1
Hierarchies average cost versus trees average cost

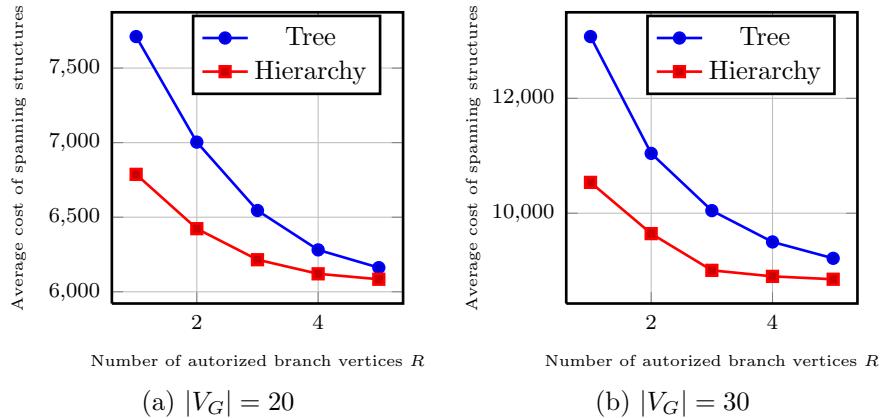


Figure 3. Optimal spanning tree versus optimal spanning hierarchy

The numerical results of the cost are presented in Table 1. Since an optimal tree for BVCMST is a feasible solution for BVCMSH, the optimal hierarchy cost is always lower than the tree cost.

The cost improvement with hierarchies is clearer in Figures 3a and 3b which represent the average costs of trees and hierarchies for $|V_G| = 20$ and $|V_G| = 30$ respectively. As shown in these figures, the average hierarchy cost is lower than tree cost in any situations. When $R \leq 2$, it can be observed that the hierarchies improve the average cost from more than 8% when $|V_G| = 20$ and from more than 12% when $|V_G| = 30$. As expected, the improvement

grows with the size of the graph and decreases when the allowed number of branch vertices grows.

5 Conclusion

Our experiments show that when the spanning structure should not necessarily be a sub-graph, hierarchies can be a good alternative to trees. Besides the fact that for any instance and any positive R there always exists a feasible solution for the MVCMSH, the cost of the optimal solution (hierarchy) of the BVCMSH problem is lower than the cost of the optimal solution (tree) of the BVCMST problem. This observation is obvious since trees are a special cases of hierarchies, but these experiments give a more accurate assessment of the cost improvement. We can particularly note that when $R \leq 2$, the hierarchies improve the average cost from more than 8% when $|V_G| = 20$ and from more than 12% when $|V_G| = 30$. For future work, it could be interesting to study more this structure in order to explore the approximation of the MBCMSH problem.

References

- [CGI09] R Cerulli, M Gentili, and A Iossa. Bounded-degree spanning tree problems: models and new algorithms. *Comput. Optim. Appl.*, 42:353–370, April 2009.
- [GHSV02] L Gargano, P Hell, L Stacho, and U Vaccaro. Spanning Trees with Bounded Number of Branch Vertices. In *Proceedings of the 29th International Colloquium on Automata, Languages and Programming, ICALP '02*, pages 355–365, London, UK, UK, 2002. Springer-Verlag.
- [GJ79] M R Garey and D S Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co, New York, NY, USA, 1979.
- [HZ94] P Hell and X Zhu. Homomorphisms to oriented paths. *Discrete Mathematics*, 132(13):107 – 114, 1994.
- [KNS74] D Klingman, A Napier, and J Stutz. A program for generating large scale capacitated assignment, transportation, and minimum cost flow network problems. *Management Science*, 20(5):814–821, 1974.
- [Mak09] O Makhorin. *GNU Linear Programming Kit (GLPK) v 4.38*, gnu project edition, May 2009.
- [Mol08] M Molnár. Hierarchies for Constrained Partial Spanning Problems in Graphs. Technical report PI-1900, 2008.

On the Biobjective Adjacent Only Quadratic Spanning Tree Problem

Sílvia M. D. M. Maia^{1,2} Elizabeth F. G. Goldbarg^{1,3}
Marco C. Goldbarg^{1,4}

*Department of Informatics and Applied Mathematics
Federal University of Rio Grande do Norte
Natal, Brazil*

Abstract

The adjacent only quadratic minimum spanning tree problem is an NP-hard version of the minimum spanning tree where the costs of interaction effects between every pair of adjacent edges are included in the objective function. This paper addresses the biobjective version of this problem. A Pareto local search algorithm is proposed. The algorithm is applied to a set of 108 benchmark instances. The results are compared to the optimal Pareto front generated by a branch and bound algorithm, which is a multiobjective adaptation of a well known algorithm for the mono-objective case.

Keywords: Multiobjective Optimization, Quadratic Minimum Spanning Tree, Pareto Local Search, Branch and Bound.

¹ This research is partially supported by Capes/CNPq

² Email: silvinha.treze@yahoo.com.br

³ Email: beth@dimap.ufrn.br

⁴ Email: gold@dimap.ufrn.br

1 Introduction

The minimum spanning tree problem (MST) seeks for the minimum cost subgraph spanning all vertices of a finite and connected graph. It is a well known combinatorial optimization problem that is solvable in polynomial time. A review of efficient algorithms for this problem is presented in [2]. The quadratic minimum spanning tree problem (QMST), introduced by [9], is a version of the MST in which a quadratic cost structure is considered. This quadratic structure models interaction effects between pairs of edges. Linear and quadratic costs are added up to compose the total cost of the generated spanning tree. When those interactions are restricted to adjacent edges, the problem is named adjacent only QMST (AQMST). The AQMST and QMST problems are NP-hard as demonstrated by [1]. Several transportation and distribution network problems can be modeled by these problems. For example, in transportation of oil and its derivatives from a pipe to another, the cost may depend on the nature of the interface between the two pipes [1]. Transportation or road networks with penalties may also require a quadratic model. A similar situation can emerge in other contexts, like telecommunication and energy distribution networks [5]. In all those cases, the AQMST emerges as the most suitable model to real problems. Exact and heuristic algorithms developed for these problems can be found in [1], [3],[4], [5], [7], [8] and [10]. It is worth noting that just a few works concern QMST and AQMST and most of them were presented recently. The growing interest of researchers indicates that these are relatively new and unexplored NP-hard combinatorial optimization problems with high applicability.

Although the linear and the quadratic costs are added up, in real world applications, it may be interesting to separate them in order to provide a better basis for decision makers. In this sense, multiobjective optimization seems to provide a more realistic model for the QMST and AQMST problems. A review of the state-of-the-art, so far, was not able to find a paper in which the inherent biobjective nature of these problems is considered. This paper presents the biobjective version of the adjacent only quadratic spanning tree problem (bi-AQST) in which linear and quadratic costs are considered as separate and independent objectives. In section 2 the problem and an exact algorithm for it are presented. The exact algorithm is a multiobjective version of the branch and bound introduced by [1] for the AQMST. A Pareto local search algorithm is presented in section 3. It is applied to the bi-AQST on a set of instances proposed by [3] for the mono-objective case. The results are reported in section 4. They are compared to the optimal Pareto front generated by the branch

and bound algorithm. Finally, conclusions are outlined in section 5.

2 The bi-AQST

Let $G = (V, E)$ denote an undirected, connected and weighted graph where $V = \{v_1, \dots, v_n\}$ is a finite set of vertices and $E = \{e_1, \dots, e_m\}$ is a finite set of edges. A weight w_i , $0 < i \leq m$, is assigned to each edge e_i . Let c_{ij} be a cost associated with the pair of edges (e_i, e_j) , known as intercost. Let T represent the set of all spanning trees of G and A_i be the set of all edges adjacent to e_i , $0 < i \leq m$, in G . Thus, the AQMST [1] may be stated as :

$$(1) \quad \text{Min} \left\{ z(x) = \sum_{i=1}^m w_i x_i + \sum_{i=1}^m \sum_{j=1, j \in A_i}^m c_{ij} x_i x_j \mid x \in T \right\}$$

where x_i equals 1 if edge e_i belongs to tree x and 0 otherwise. As in literature, given two edges e_1 and e_2 , both c_{12} and c_{21} are counted in this work.

The bi-AQST takes into account that the linear and quadratic components of expression 1 may have different and conflicting nature. Therefore, they are considered as separate and independent criteria, according to the formulation given in expressions 2 and 3:

$$(2) \quad \text{Min} \{z(x) = (f_1(x), f_2(x)) \mid x \in T\}$$

where:

$$(3) \quad f_1(x) = \sum_{i=1}^m w_i x_i \text{ and } f_2(x) = \sum_{i=1}^m \sum_{j=1, j \in A_i}^m c_{ij} x_i x_j$$

The multiobjective approach usually implies there is not a single solution for the problem. So, the word *minimize* has to be understood in another context, that is the Pareto optimization. Pareto dominance relations are explained as follows [11]. Let k be the number of objectives and $x, y \in X$, the set of all feasible solutions. Then x dominates y , written $x \prec y$, iff $\forall i, i = 1, \dots, k, f_i(x) \leq f_i(y)$ and $\exists i$, such that $f_i(x) < f_i(y)$. If neither $x \prec y$ nor $y \prec x$, x and y are non-dominated with respect to each other. The set of Pareto optimal solutions $X^* \subseteq X$, also called efficient solutions, is composed by the non-dominated solutions regarding the entire set of feasible solutions X . Thus, a solution $x^* \in X^*$ iff $\nexists x \in X$ such that $x \prec x^*$. Therefore, the bi-AQST seeks for the efficient set of solutions obtained with the formulation given in expressions 2 and 3.

An iterative branch and bound algorithm for the bi-AQST problem, based on the one presented by [1] for the single objective problem, is presented in the following. The upper bound is the set of non-dominated solutions so far

discovered. The lower bound concerning the linear objective is the cost of the minimum spanning tree of G when the weights in the edges correspond to their linear costs and the edges already inserted in the tree by the branch and bound are made mandatory. The lower bound for the quadratic objective is also the cost of a minimum spanning tree where the edges already in the tree are mandatory. However, in this case, the cost associated to each edge e_i of G out of the tree is g_i , defined in equations 4 and 5, where U is the set of edges already in the tree and F is the set of available edges.

$$(4) \quad g_i = \min\{c_{ij} | e_j \in F \cap A_i\}, \text{ if } U \cap A_i = \emptyset$$

$$(5) \quad g_i = \sum_{j \in U} c_{ij} + c_{ji}, \text{ otherwise}$$

At each step of the branch and bound algorithm, the available edge with the smallest lower bound with respect to the quadratic objective is selected to enter the tree. If this inclusion does not induce a cycle, the lower bound is recalculated. If the new lower bound corresponds to a dominated value regarding the current set of non-dominated solutions, the tree is pruned. Otherwise, the branching process continues. An edge is moved to the set of unavailable edges if one of the three following conditions occurs: the edge is included in the solution tree, its inclusion induces a cycle or its inclusion causes the search tree to be pruned. When no edges are available for insertion in the tree, the backtracking is performed and the appropriate edges in the unavailable set are moved back to the set of available edges. The search ends when there is no available edge to insert in the tree and the tree is empty, i.e., the backtracking condition holds for the first level of the branch and bound search tree.

3 Pareto Local Search Algorithm for bi-AQST

Pareto Local Search (PLS) was presented by [6] and is a general framework for designing local search algorithms in multiobjective scenarios. The first step of the PLS algorithm presented in this paper is to generate an initial solution π using a random procedure composed essentially of three main steps. At first, a random edge is selected to get into the initial solution. Then, iteratively, the list of candidate edges to get into the tree is updated and a random edge e_i is selected from that list to enter the initial solution. Edges are removed from the list of candidate edges if they induce a cycle with the edges in the tree. Edges are added to the list if they are adjacent to e_i and do not induce a cycle with the edges in the tree. This process continues until the tree is complete.

After π is built, it is marked as non-visited and added to the current list of non-dominated solutions NDL . Then, an iterative process is initiated. At each step, a non-visited solution sol from NDL is chosen to be visited. All the neighbors of sol are generated according to a specific neighborhood structure detailed further. If a neighbor α of sol is non-dominated with respect to all solutions in NDL , α is marked as non-visited and added to NDL . Solutions dominated by α in NDL are removed from this set. After the exam of its neighborhood, sol is marked as visited. This iterative procedure continues until all solutions in NDL become visited. In the end of the algorithm, NDL is the approximation set generated by the PLS algorithm.

The neighborhood structure adopted during the PLS procedure to explore sol , named N_t^1 , is widely applied for the QMST problem. In order to generate a neighbor of a given tree t with respect to the neighborhood N_t^k , one should proceed as follows. First, randomly select k edges to leave t . Then, choose, among the edges out of the tree, a set of k edges that keeps t as a tree. If the best improvement approach is adopted, this set should contain the k edges that allow the best improvement in the objective function. Finally, add the selected k edges to the tree [4]. Thus, a tree t' is a neighbor of t with respect to N_t^k , if t' differs from t by exactly k edges.

As described above, the PLS algorithm examines the whole neighborhood of sol (the solution being visited). Therefore, when generating the neighborhood N_{sol}^1 of sol , instead of randomly choose edges to leave the tree, we perform a systematic search, replacing each edge in sol by each edge out of sol that does not induce a cycle.

4 Computational Experiments

This section reports the results obtained by the branch and bound and the PLS algorithms. The instances adopted in the computational experiments are available at <http://homes.di.unimi.it/~cordone/research/qmst.html>. They were proposed by [3] and are here named *IS*. The set consists of 108 graphs divided into 9 classes, according to the number of vertices, that is a value from the set $NV = \{10, 15, 20, 25, 30, 35, 40, 45, 50\}$. Each class contains 12 graphs, generated from a combination of three characteristics, namely: density of the graph (33%, 67% or 100%); range of choice for linear cost ([0, 10] or [0, 100]) and range of choice for intercost ([0, 10] or [0, 100]). The algorithms were implemented in C++, using g++ compiler and Ubuntu 12.04 32 bits. All experiments were performed in a machine with an i5 650 processor operating at 3.2 GHz and 4GB of RAM memory at 1333 MHz.

We were able to run the branch and bound algorithm for all instances with 10 and 15 vertices and for the first 4 instances with 20 vertices. For other instances, the algorithm was not able to end in less than 24 hours. In average, the branch and bound algorithm spent, respectively, 0.235 seconds, 41.519 minutes and 5.898 hours to run instances with 10, 15 and 20 vertices.

Table 1 shows the results of the PLS algorithm for the subset of instances solved by the branch and bound algorithm. The PLS algorithm was executed 30 times for each instance. The column named “Time” presents the average time in seconds. The column “Opt” presents the average percentage of Pareto optimal solutions generated by the PLS algorithm. Thus, this information is calculated as 100 times the number of optimal solutions generated by the PLS divided by the number of solutions in the Pareto front generated by the branch and bound. In average, the PLS algorithm was able to find about 66.70% of the Pareto front.

Table 1
Pareto Local Search Results - part 1

Instance	Time	Opt	Instance	Time	Opt
10.1	0.000	100.00	15.3	0.006	62.38
10.2	0.000	100.00	15.4	0.009	71.75
10.3	0.000	100.00	15.5	0.017	50.23
10.4	0.000	100.00	15.6	0.019	50.35
10.5	0.004	100.00	15.7	0.020	76.00
10.6	0.003	71.82	15.8	0.021	50.95
10.7	0.004	100.00	15.9	0.022	39.33
10.8	0.004	74.76	15.10	0.022	34.64
10.9	0.007	44.35	15.11	0.027	65.65
10.10	0.007	55.07	15.12	0.033	62.01
10.11	0.008	56.47	20.1	0.014	48.60
10.12	0.009	63.44	20.2	0.017	40.00
15.1	0.005	65.41	20.3	0.022	56.16
15.2	0.006	65.38	20.4	0.038	62.90

Table 2 exhibits the results of the PLS algorithm for the remaining instances. In this case, we present the average computational time, in seconds, and the average number of solutions in the generated approximation set. Considering the whole set of instances, PLS consumes, in average, 5.14s of computational time and generates, in average, 70.19 solutions per approximation set.

During the computational experiments, variations of the local search algorithm for the bi-AQST were investigated, where both solutions quality and execution time were considered. A multi-start and an Iterated Local Search version of the PLS were developed. Another attempt to improve PLS’s results considered the use of both N_t^1 and N_t^2 in a VNS algorithm. Nevertheless, the

Table 2
Pareto Local Search Results - part 2

Inst	Time	numSol	Inst	Time	numSol	Inst	Time	numSol
20.1	0.014	24.00	30.5	0.452	30.70	40.9	3.224	30.27
20.2	0.017	31.57	30.6	0.668	42.07	40.10	5.188	50.70
20.3	0.022	48.20	30.7	1.027	84.83	40.11	10.500	111.83
20.4	0.038	71.03	30.8	2.099	131.20	40.12	25.957	183.07
20.5	0.053	24.60	30.9	0.798	26.60	45.1	2.332	65.43
20.6	0.068	29.57	30.10	1.232	42.17	45.2	3.341	77.57
20.7	0.112	55.03	30.11	1.523	61.20	45.3	5.632	142.67
20.8	0.201	86.70	30.12	3.375	107.53	45.4	11.580	204.37
20.9	0.117	24.27	35.1	0.503	46.13	45.5	5.090	51.47
20.10	0.153	29.07	35.2	0.667	55.33	45.6	8.482	70.33
20.11	0.212	51.50	35.3	1.103	100.40	45.7	16.099	138.50
20.12	0.331	68.67	35.4	2.200	156.87	45.8	31.612	210.13
25.1	0.055	23.97	35.5	1.067	32.23	45.9	6.870	36.87
25.2	0.068	31.07	35.6	1.672	46.00	45.10	12.009	58.73
25.3	0.094	58.33	35.7	2.676	100.27	45.11	20.890	113.63
25.4	0.121	70.30	35.8	6.114	160.23	45.12	48.255	190.50
25.5	0.221	24.20	35.9	1.810	28.83	50.1	4.348	70.80
25.6	0.249	31.80	35.10	2.685	40.40	50.2	6.073	90.07
25.7	0.424	67.23	35.11	4.959	91.80	50.3	10.449	160.67
25.8	0.608	89.00	35.12	10.278	143.83	50.4	24.373	261.07
25.9	0.339	23.63	40.1	0.957	53.90	50.5	8.025	44.70
25.10	0.460	34.43	40.2	1.500	66.73	50.6	13.767	73.07
25.11	0.798	61.60	40.3	2.160	116.67	50.7	22.773	165.43
25.12	1.336	95.577	40.4	4.277	183.13	50.8	36.894	259.83
30.1	0.225	45.07	40.5	2.503	42.07	50.9	7.046	44.20
30.2	0.293	51.07	40.6	3.733	58.60	50.10	12.370	69.53
30.3	0.375	75.87	40.7	5.979	104.37	50.11	26.260	163.90
30.4	0.633	107.7	40.8	14.346	187.57	50.12	75.377	254.77

high increase in computational effort was not significantly compensated by improvements in the quality of the approximation sets produced by the local search variations. The Mann-Whitney test regarding the quality indicators hypervolume and additive binary epsilon [11] with significance level 0.05 revealed evidences that the PLS is the best approach among the investigated variations.

5 Conclusions

This paper introduced a new problem, named Biobjective Adjacent Only Quadratic Spanning Tree, which is a more realistic model for real network problems. A branch and bound and a Pareto Local Search algorithm were proposed for the problem and tested on a set of 108 benchmark instances. The exact algorithm was able to solve instances up to twenty nodes. The approximation sets produced by the Pareto Local Search algorithm were com-

pared to the optimal Pareto fronts obtained with the exact algorithm. For the subset of instances solved by the branch and bound algorithm, the PLS was able to find approximately 66.70% of the Pareto front within a reasonable time.

References

- [1] Assad, A. and W. Xu, *The quadratic minimum spanning tree problem*, Naval Research Logistics **39** (1992), 399–417.
- [2] Bazlamaç, C. F. and K. S. Hindi, *Minimum-weight spanning tree algorithms: a survey and empirical study*, Computers and Operations Research **28** (2001), 767–785.
- [3] Cordone, R. and G. Passeri, *Solving the quadratic minimum spanning tree problem*, Applied Mathematics and Computation **218** (23) (2012), 11597–11612.
- [4] Öncan, T. and A.P. Punnen, *The quadratic minimum spanning tree problem: a lower bounding procedure and an efficient search algorithm*, Computers and Operations Research **37** (2010), 1762–1773.
- [5] Palubeckis, G., D. Rubliauskas and A. Targamadzé, *Metaheuristic Approaches for the Quadratic Minimum Spanning Tree Problem*, Information Technology and Control **39** (4) (2010), 257–268.
- [6] Paquete, L. and T. Stützle, “Stochastic Local Search Algorithms for Multiobjective Combinatorial Optimization: a review”, Technical report, 2006.
- [7] Soak, S. M., D. W. Corne and B. H. Ahn, *The edge-window-decoder representation for tree based problems*, IEEE Transactions on Evolutionary Computation **10** (2006) 124–144.
- [8] Sundar, S. and A. Singh, *A swarm intelligence approach to the quadratic minimum spanning tree problem*, Information Sciences **180** (2010), 3182–3191.
- [9] Xu, W., “Quadratic minimum spanning tree problems and related topics”, Ph.D. thesis, University of Maryland, 1984.
- [10] Zhou, G. and M. Gen, *An effective genetic algorithm approach to the quadratic minimum spanning tree problem*, Computers and Operations Research **25** (1998) 229–237.
- [11] Zitzler, E., L. Thiele, M. Laumanns, C. M. Fonseca, V. G. Fonseca, *Performance assessment of multiobjective optimizers: an analysis and review*, IEEE Transactions on Evolutionary Computation **7** (2) (2003), 117–132.

On single-path network routing subject to max-min fair flow allocation

Edoardo Amaldi

*Dipartimento di Elettronica, Informazione e Bioingegneria
Politecnico di Milano, Milano, Italy
amaldi@elet.polimi.it*

Stefano Coniglio

*Lehrstuhl II für Matematik
RWTH Aachen University, Aachen, Germany
coniglio@math2.rwth-aachen.de*

Luca G. Gianoli, Can Umut Ileri

*Dipartimento di Elettronica, Informazione e Bioingegneria
Politecnico di Milano, Milano, Italy
gianoli@elet.polimi.it, ilteri@mail.polimi.it*

Abstract

Fair allocation of flows in multicommodity networks has been attracting a growing attention. In Max-Min Fair (MMF) flow allocation, not only the flow of the commodity with the smallest allocation is maximized but also, in turn, the second smallest, the third smallest, and so on. Since the MMF paradigm allows to approximate the TCP flow allocation when the routing paths are given and the flows are elastic, we address the network routing problem where, given a graph with arc capacities and a set of origin-destination pairs with unknown demands, we must route each commodity over a single path so as to maximize the throughput, subject

to the constraint that the flows are allocated according to the MMF principle. After discussing two properties of the problem, we describe a column generation based heuristic and report some computational results.

Keywords: Max-min fairness, network routing, integer programming, column generation

1 Introduction

Telecommunication multicommodity flow problems involving the concept of max-min fairness have been attracting a growing attention (see, e.g., [1] and the references therein).

Consider the task of sharing the network capacity (bandwidth) among n commodities (users). Let $\phi \in \mathbb{R}^n$ denote a flow vector where the i -th component ϕ_i corresponds to the flow allocated to the i -th commodity. Let σ be a sorting operator permuting the components of ϕ in nondecreasing order, i.e., such that $\sigma(\phi)_i \leq \sigma(\phi)_j$ whenever $i < j$.

Definition: A flow vector $\phi \in \mathbb{R}^n$ is *Max-Min Fair* (MMF) if, for any other flow vector $\phi' \in \mathbb{R}^n$, $\sigma(\phi)$ lexicographically dominates $\sigma(\phi')$, i.e., either $\sigma(\phi) = \sigma(\phi')$ or there exists an integer k , with $1 \leq k \leq n$, such that $\sigma(\phi)_k > \sigma(\phi')_k$ and $\sigma(\phi)_l = \sigma(\phi')_l$ for all $l < k$.

In other words, a flow vector is MMF if there is no way to increase the flow of any commodity without decreasing the flow of a commodity with a smaller or equal flow.

Let us briefly summarize previous work on network routing problems with MMF flow allocation. When the routing paths are given, a simple polynomial-time algorithm, the so-called *Water Filling* algorithm yields an MMF flow allocation [2]. When the routing paths are not known a priori, algorithms to find an MMF flow allocation have been proposed for unsplittable (single path) or splittable routing, see, e.g., [3] and the survey [1]. For the special case with a single source, see [4] for a polynomial-time algorithm which applies to splittable (fractional) flows, and [5] for an \mathcal{NP} -hardness result and approximation algorithms for unsplittable routing. The reader is referred to [1] for a survey dealing also with some generalizations.

To the best of our knowledge, in the literature the MMF paradigm has only been considered as the objective function. This is in contrast with the fact that network operators aim at maximizing a network utility function (e.g., throughput) and in IP networks, once the routing paths have been chosen by

the IP layer, the transport protocol (e.g., TCP) achieves an average bandwidth allocation which is approximated by MMF [6].

In this work, along the line of [7], we address the following problem:

Max-Throughput Single-Path Network Routing subject to MMF flow allocation (MT-SPNR-MMF): Given a directed graph $G = (V, A)$ with capacities c_{ij} for each $(i, j) \in A$ and a set of commodities K with origin-destination pairs (s, t) , route each commodity over a single path so as to maximize the throughput, subject to the constraint that the amount of flow allocated to the commodities be MMF w.r.t. all the flow allocations that are feasible for the paths that have been chosen.

As observed in [7], MT-SPNR-MMF is a bilevel problem where, at the upper level, the leader (network operator) selects a routing path for each commodity and, at the lower level, a follower (TCP protocol) allocates the flows (bandwidth) to the chosen paths according to the MMF paradigm.

2 Properties of MT-SPNR-MMF

Let us start with two simple properties of the problem.

Proposition 2.1 *MT-SPNR-MMF is \mathcal{NP} -hard even when $c_{ij} = 1, \forall (i, j) \in A$.*

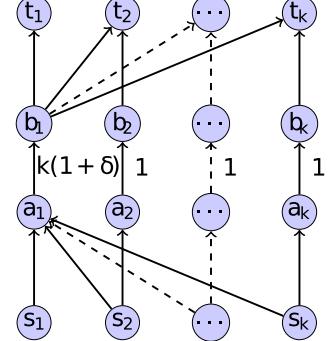
Proof. By straightforward reduction from the \mathcal{NP} -complete problem of deciding whether a given digraph G with k origin-destination pairs admits k edge-disjoint paths. It suffices to observe that G admits k edge-disjoint paths if and only if the MT-SPNR-MMF instance with the same graph G , the same k origin-destination pairs, and all $c_{ij} = 1$ admits an MMF flow allocation with a total throughput of value k . If it is the case, since each commodity achieves the maximum flow of value 1, k edge-disjoint paths are used. \square

Consider the “classical” Single-Level MMF Single-Path Network Routing problem (SL-MMF-SPNR) where we look for a solution whose flow vector is lexicographically undominated w.r.t. all the possible choices of flows and routing paths.

Proposition 2.2 *The gap in terms of throughput or smallest flow allocation between an optimal solution to MT-SPNR-MMF and an optimal solution to SL-MMF-SPNR can be arbitrarily large.*

Proof. Consider the example in the figure below, with k commodities, capacity 1 for the arcs (a_i, b_i) , $i = 2, \dots, k$, capacity $k(1 + \delta)$ for the arc (a_1, b_1) ,

with $\delta > 0$, and an arbitrarily large capacity for the other arcs. It is easy to verify that $\sigma(\phi) = (1, \dots, 1, k(1 + \delta))$, with a total throughput of $k(1 + \delta) + k - 1$, is an optimal solution to MT-SPNR-MMF, while $\sigma(\phi) = (1 + \delta, \dots, 1 + \delta)$, with a total throughput of $k(1 + \delta)$, is an optimal solution to SL-MMF-SPNR. For a fixed δ and an arbitrarily large k , these solutions differ by an additive factor of $k - 1$ in terms of throughput. For a fixed k and an arbitrarily large δ , these solutions differ by an additive factor of δ in terms of smallest flow allocation.



3 Column generation

In this section, we describe a column generation method for MT-SPNR-MMF based on a mixed integer programming (MILP) path formulation. For an arc formulation, see [7].

Let P^{st} be the set of feasible paths for each commodity $(s, t) \in K$ and let the constant σ_{ij}^{pst} be 1 if the path $p \in P^{st}$ contains the arc $(i, j) \in A$ and 0 otherwise. Let the variable $\lambda_{st}^p \in \{0, 1\}$ be 1 if the path of index p is chosen for the commodity (s, t) and 0 otherwise. For each commodity $(s, t) \in K$, let ϕ^{st} be the amount of flow assigned to it and let f_{ij}^{st} be its amount on arc $(i, j) \in A$. Let u_{ij} be an upper bound on any flow over the arc $(i, j) \in A$. The variables y_{ij}^{st} , which are instrumental in the MMF constraints, are described after the formulation. Then, MT-SNPR-MMF can be formulated as follows:

$$\max \sum_{(s,t) \in K} \phi^{st} \quad (1)$$

$$\text{s.t. } \sum_{p \in P_{st}} \lambda_{st}^p = 1 \quad \forall (s, t) \in K \quad (2)$$

$$\sum_{(i,j) \in \delta^+(i)} f_{ij}^{st} - \sum_{(j,i) \in \delta^-(i)} f_{ji}^{st} = \begin{cases} \phi^{st} & \text{if } i = s \\ -\phi^{st} & \text{if } i = t \\ 0 & \text{else} \end{cases} \quad \forall (s,t) \in K, i \in V \quad (3)$$

$$\sum_{(s,t) \in K} f_{ij}^{st} \leq c_{ij} \quad \forall (i,j) \in A \quad (4)$$

$$f_{ij}^{st} - c_{ij} \sum_{p \in P^{st}} (\sigma_{ij}^{pst} \lambda_{st}^p) \leq 0 \quad \forall (i, j) \in A, (s, t) \in K \quad (5)$$

$$\phi^{st} \geq \frac{\min_{(i,j) \in A} \{c_{ij}\}}{|K|} \quad \forall (s,t) \in K \quad (6)$$

$$\sum_{(i,j) \in A} y_{ij}^{st} \geq 1 \quad \forall (s,t) \in K \quad (7)$$

$$\sum_{(o,d) \in K} f_{ij}^{od} \geq c_{ij} y_{ij}^{st} \quad \forall (i,j) \in A, (s,t) \in K \quad (8)$$

$$u_{ij} \geq f_{ij}^{st} \quad \forall (i,j) \in A, (s,t) \in K \quad (9)$$

$$u_{ij} - f_{ij}^{st} \leq c_{ij}(1 - y_{ij}^{st}) \quad \forall (i,j) \in A, (s,t) \in K \quad (10)$$

$$\phi^{st} \geq 0, f_{ij}^{st} \geq 0, y_{ij}^{st} \in \{0,1\}, \lambda_{st}^p \in \{0,1\}, u_{ij} \geq 0 \quad \forall (i,j) \in A, (s,t) \in K. \quad (11)$$

Constraints (3)-(4) are standard flow conservation and capacity constraints. Constraints (5) guarantee $f_{ij}^{st} = 0$ if no path $p \in P^{st}$ with $\sigma_{ij}^{pst} = 1$ is selected. Constraints (6) impose a valid lower bound on the flow values, achieved when all the flows are routed over the link with minimum capacity.

Constraints (7)-(10), which are expressed in a slightly different w.r.t. [8], are the MMF constraints imposing that the flow vector be MMF for the selected paths. Let us briefly explain why they are sufficient to guarantee an MMF flow allocation. For a given set of paths, an MMF bandwidth allocation can be uniquely determined via the Water Filling (WF) algorithm. Starting from $\phi^{st} = 0$ for all $(s,t) \in K$, WF simultaneously increases all the flows until one or more arcs are saturated (we refer to them as to *bottleneck* arcs), removes them and the saturating commodities, updates the capacities to their residual values, and iterates until no commodity or arcs are left. For each arc $(i,j) \in A$ and commodity $(s,t) \in K$, we thus introduce the binary variable y_{ij}^{st} which is equal to 1 if (i,j) is a bottleneck arc for (s,t) , and to 0 otherwise. Due to the correctness of WF, ϕ is MMF for the given set of paths if and only if it satisfies (7)-(10). Constraints (7) ensure that we have at least a bottleneck arc for each $(s,t) \in K$, Constraints (8) guarantee that the bottleneck arcs are saturated, and Constraints (9)-(10) impose that the flow through a bottleneck arc (i,j) for a pair (s,t) be at least as large as the largest flow through (i,j) for all the other commodities.

The restricted master problem is obtained from (1)-(11) by restricting P^{st} , for each $(s,t) \in K$, to the paths which have been generated so far. Its LP relaxation is obtained by dropping the integrality constraints on y_{ij}^{st} and λ_{st}^p .

Let $w^{st} \in \mathbb{R}$ and $\pi_{ij}^{st} \geq 0$ be the dual variables associated to, respectively, Constraints (2) and (5). For any commodity (s,t) and a corresponding path of index $p \in P^{st}$, the dual constraint corresponding to λ_{ij}^p is $w^{st} - \sum_{(i,j) \in A} c_{ij} \sigma_{ij}^{pst} \pi_{ij}^{st} \geq 0$. It is violated (or, equivalently, the corresponding column has a nonnegative reduced cost) if $\sum_{(i,j) \in A} c_{ij} \sigma_{ij}^{pst} \pi_{ij}^{st} \geq w^{st}$. For each commodity (s,t) , the pricing subproblem thus amounts to finding a longest path on the original graph with weights $c_{ij} \pi_{ij}^{st}$ for each arc $(i,j) \in A$. This

can be done by solving a MILP which is obtained by adding to the standard LP formulation for the shortest path problem with nonnegative costs a modified version of the variables and constraints used to prevent subtours in the extended formulation for the TSP by Wong [9]. Let z_i , for $i \in V \setminus \{s, t\}$, be 1 if the path contains node i as an intermediate node and 0 otherwise. Let then $V_s := V \setminus \{s\}$ and let $q_{ij}^k \geq 0$ be the variable of an auxiliary flow from node s to a node $k \in V_s$. Then, the pricing subproblem can then be formulated as:

$$\max \sum_{(i,j) \in A} c_{ij} \sigma_{ij}^{pst} \pi_{ij}^{st} \quad (12)$$

$$\text{s.t. } \sum_{(i,j) \in \delta^+(i)} \sigma_{ij}^{pst} - \sum_{(j,i) \in \delta^-(i)} \sigma_{ji}^{pst} = \begin{cases} 1 & \text{if } i = s \\ -1 & \text{if } i = t \\ 0 & \text{otherwise} \end{cases} \quad \forall i \in V \quad (13)$$

$$\sum_{(i,j) \in \delta^+(i)} q_{ij}^k - \sum_{(j,i) \in \delta^-(i)} q_{ji}^k = \begin{cases} z_k & \text{if } i = s \\ -z_k & \text{if } i = k \\ 0 & \text{otherwise} \end{cases} \quad \forall k \in V_s, i \in V \quad (14)$$

$$q_{ij}^k \leq \sigma_{ij}^{pst} \quad \forall (i,j) \in A, k \in V_s \quad (15)$$

$$z_i \leq \sum_{(j,i) \in \delta^-(i)} \sigma_{ji}^{pst} \quad \forall i \in V_s \quad (16)$$

$$\sigma_{ij}^{pst} \in \{0, 1\} \quad \forall (i,j) \in A \quad (17)$$

$$q_{ij}^k \geq 0 \quad \forall (i,j) \in A, k \in V_s \quad (18)$$

$$z_i \in \{0, 1\} \quad \forall i \in V. \quad (19)$$

To accelerate our column generation, we initialize the column pool with 3 random paths per commodity and, to favor diversity between the paths, we proceed as follows. For each commodity, as long as edge-disjoint paths are found, we generate the new path by solving a shortest path problem w.r.t. uniformly random weights. Whenever a duplicate path is generated, we increase the weight of its arcs by a factor of 2, and then repeat the generation.

To speed up the convergence to an optimal solution of the relaxed master problem, we consider alternative techniques to the “text-book” generation of columns with a maximally reduced cost. Among various options, the best results are obtained with a pseudo-random technique where, in the pricing subproblem, we optimize a linear function with uniformly drawn random coefficients, subject to constraint that the reduced cost be at least 10^{-6} .

Since, in general, the solution found with column generation is not integer, after the algorithm is halted, we simply impose the integrality on y_{ij}^{st} and λ_{st}^p and reoptimize the restricted master problem to obtain a lower bound. In the following, the resulting method is referred to as *Column Generation based MILP heuristic*.

4 Some computational results

We consider four network topologies taken from the SND library: `polska` ($|V| = 12, |A| = 36$), `france` ($|V| = 25, |A| = 90$), `nobel-us` ($|V| = 14, |A| = 42$), `atlanta` ($|V| = 15, |A| = 44$). From each topology, we generate 20 instances adopting 4 different generation techniques, with an increasing number of commodities and using either randomly generated capacities (10, 5, 2, and 1 Gbps with a probability of, respectively, 0.4, 0.3, 0.2, and 0.1) or uniform ones. For a detailed description, see the full version of the paper.

The computational experiments are carried out with CPLEX 12.3 using the AMPL modeling language on a machine equipped with 2 Intel Xeon E5645 CPUs with 2.40 GHz and 16 GB of RAM. When solving the MILP restricted master problem, we adopt a time limit of 3600 seconds. Whenever solving a MILP, we set the parameter `mipemphasis=4`.

Due to space restrictions, in the following table we only report the results for a subset of the instances.

Network	Inst. Code	$ K $	Complete formulation			Column generation					
			LB	Int. Gap	Time	UB	LB	Gap	Time CG	Time CG+MILP	# Gen. Cols.
atlanta	1-1	12	48.5	0.0	1.4	50.8	48.5	4.8	2.0	0.2	69
atlanta	1-2	20	62.5	0.0	16.2	62.5	59.5	5.0	2.3	27.4	68
atlanta	1-3	30	96.1	2.1	3593.9	98.6	96.5	2.1	6.9	452.1	71
atlanta	1-4	42	-	-	3594.1	76.0	-	-	3.3	3594.0	112
atlanta	1-5	56	-	-	3594.1	87.3	82.1	6.4	3.8	3594.0	96
france	1-2	10	52.5	0.0	58.3	55.0	52.5	4.8	2.2	0.4	100
france	1-3	15	56.5	0.0	482.6	60.1	56.5	6.3	6.2	1.2	199
france	1-4	21	76.5	0.9	3600.0	78.1	76.5	2.0	12.2	94.3	317
france	1-5	28	-	-	3600.0	114.8	113.5	1.1	6.7	33.4	192
france	1-6	36	-	-	3600.0	116.3	116.0	0.2	28.5	184.9	403
nobel-us	1-3	15	63.5	0.0	62.1	63.8	63.5	0.5	2.0	1.5	67
nobel-us	1-4	21	75.0	0.0	2627.0	76.0	75.0	1.3	4.9	47.6	126
nobel-us	1-5	28	104.5	2.4	3600.0	108.0	105.3	2.6	6.5	407.8	220
nobel-us	1-6	36	-	-	3600.0	91.0	90.3	0.8	4.2	2280.1	132
nobel-us	1-7	42	-	-	3600.0	103.9	103.5	0.5	4.5	2414.8	203
polska	1-3	21	90.0	0.0	20.0	90.8	86.7	4.8	0.4	5.3	22
polska	1-4	28	71.4	0.8	3594.1	71.9	68.7	4.6	1.3	5.3	27
polska	1-5	36	81.8	0.8	3593.9	82.4	80.5	2.4	4.2	680.8	69
polska	1-6	42	-	-	3600.0	118.3	111.4	6.2	2.8	2676.4	69
polska	1-7	56	-	-	3600.0	151.6	143.6	5.6	2.3	3594.0	100

The Column Generation based MILP heuristic clearly outperforms the complete arc formulation proposed in [7]. Indeed, in the time limit of 3600 seconds, with the complete formulation no feasible solution is found for 28 instances out of 80 (more than 25%), as opposed to only 4 when using the former one. If we compute the ratio, for each instance, between the value of the solution found with the two methods and then consider their geometric mean, with the column generation based method we have solutions that are worse

than those for the complete formulation by less than 0.01%. Most interestingly, if we compute the geometric mean of the ratios for the computing times, we observe that the column generation based method requires less than 15% the time that is needed when solving the complete formulation.

As to the column generation method without adopting the MILP heuristic, if we compute the gap between its upper bounds and the lower bounds corresponding to the best feasible solution found with either methods, we obtain a value of only 2.75%. Note that, for all the instances but two, the total time invested in the column generation procedure is much smaller than 10 seconds and, overall, it never exceeds 30 seconds. The quality of such bounds suggests that a Branch-and-Price algorithm, which is under development, may be an effective technique to solve the problem to optimality.

References

- [1] D. Nace and M. Pióro. Max-min fairness and its applications to routing and load-balancing in communication networks: a tutorial. *Communications Surveys & Tutorials, IEEE*, 10(4):5–17, 2008.
- [2] D. Bertsekas and R. Gallager. *Data Networks, 1992*. Prentice-Hall, 1992.
- [3] W. Ogryczak, M. Pióro, and A. Tomaszewski. Telecommunications network design and max-min optimization problems. *Journal of Telecommunications and Information Technology*, 3:1–14, 2005.
- [4] N. Megiddo. Optimal flows in networks with multiple sources and sinks. *Mathematical Programming*, 7(1):97–107, 1974.
- [5] J. Kleinberg, Y. Rabani, and É. Tardos. Fairness in routing and load balancing. In *Foundations of Computer Science, 1999. 40th Annual Symposium on*, pages 568–578. IEEE, 1999.
- [6] L. Massoulié and J. Roberts. Bandwidth sharing: objectives and algorithms. *Networking, IEEE/ACM Transactions on*, 10(3):320–328, 2002.
- [7] E. Amaldi, A. Capone, S. Coniglio, and L. Gianoli. Network optimization problems subject to max-min fair flow allocation. *Under revision*, 2012.
- [8] A. Tomaszewski. A polynomial algorithm for solving a general max-min fairness problem. *European transactions on telecommunications*, 16(3):233–240, 2005.
- [9] R.T. Wong. Integer programming formulations of the travelling salesman problem. In *Proc. IEEE Conf. on Circuits and Computers*, pages 149–152, 1980.

Two Node-Disjoint 3-Hop-Constrained Survivable Network Design and Polyhedra

Ibrahima Diarrassouba¹

*Laboratoire LMAH
Le Havre University
Le Havre, France*

Hakan Kutucu²

*Department of Computer Engineering
Karabuk University
Karabuk, Turkey*

A. Ridha Mahjoub³

*Laboratoire LAMSADE
Université Paris Dauphine
Paris, France*

Abstract

Given a weighted undirected graph G with a set of pairs of terminals $\{s_i, t_i\}, i = 1, \dots, d$, and an integer $L \geq 2$, the two node-disjoint hop-constrained survivable network design problem (TNHNDP) is to find a minimum weight subgraph of G such that between every s_i and t_i there exist at least two node-disjoint paths of length at most L . This problem has applications to the design of survivable telecommunications networks with QoS-constraints. We discuss this problem from a polyhedral point of view. We present several classes of valid inequalities along with necessary and/or sufficient conditions for these inequalities to be facet defining. We also discuss separation routines for these classes of inequalities. Using this, we propose a Branch-and-Cut algorithm for the problem when $L = 3$, and present some computational results.

Keywords: Survivable network, node-disjoint paths, hop constraint, polyhedron, facet, branch-and-cut

1 Introduction

Given a weighted undirected graph $G = (N, E)$, an integer $L \geq 2$, and a set of demands $D \subseteq N \times N$, the two node-disjoint hop-constrained survivable network design problem (TNHNDP) consists in finding a minimum weight subgraph of G containing at least two node-disjoint paths of at most L hops between each pair of nodes $\{s, t\}$ in D .

The edge version of the problem (TEHNDP) has been already investigated by several authors when $L = 2, 3$. In particular, [5] give a complete and minimal linear description of the corresponding polytope when $L = 2, 3$ and $|D| = 1$. [3] and [1] have studied the problem when $|D| \geq 2$ and when two and k edge-disjoint paths, respectively, are required. They devise Branch-and-Cut algorithms for the problem when $L = 2, 3$ and give some computational results. Also, Huygens and Mahjoub [4] have studied the two versions of the problem (TNHNDP and TEHNDP) when $L = 4$. They give an integer programming formulation for the problem in the two cases.

Given an edge subset $F \subseteq E$, the $0-1$ vector $x^F \in \mathbb{R}^E$, such that $x^F(e) = 1$ if $e \in F$ and $x^F(e) = 0$ otherwise, is called the *incidence vector* of F . The convex hull of the incidence vectors of the solutions to the TNHNDP on G , denoted by $P(G, L)$, will be called the TNHNDP polytope. If $W \subset N$ is a node subset of G , then the set of edges that have only one node in W is called a cut and denoted by $\delta(W)$. We will write $\delta(v)$ for $\delta(\{v\})$. A cut $\delta(W)$ such that $s \in W$ and $t \in N \setminus W$ will be called an *st-cut*. Given a node $z \in N$, the graph $G - z$ is the subgraph obtained from G by deleting the node z and all its incident edges (but not their other end nodes). Let V_0, V_1, \dots, V_{L+1} be a partition of N such that $s \in V_0$, $t \in V_{L+1}$, and $V_i \neq \emptyset$ for all $i = 1, \dots, L$. Let T be the set of edges $e = uv$, where $u \in V_i$, $v \in V_j$, and $|i - j| > 1$. The set T is called an *L-path-cut*. The nodes in N that do not belong to any demand of D will be called *Steiner* nodes. $\delta(V_0, V_1, \dots, V_{L+1})$ is the set of edges between any two subsets of the partition of N .

The following linear system along with the integrality constraints formulates the TNHNDP as an integer program when $L = 2, 3, 4$ (see [4]).

¹ Email: diarrasi@univ-lehavre.fr

² Email: hakankutucu@karabuk.edu.tr

³ Email: mahjoub@lamsade.dauphine.fr

$$x(\delta(W)) \geq 2 \quad \text{for all } st\text{-cuts } \delta(W), \quad (1)$$

$$x(\delta_{G-z}(W)) \geq 1 \quad \text{for all } st\text{-cuts } \delta_{G-z}(W), \text{ for all } z \in N \setminus \{s, t\}, \quad (2)$$

$$x(T) \geq 2 \quad \text{for all } L\text{-path-cuts } T, \quad (3)$$

$$x(T_{G-z}) \geq 1 \quad \text{for all } L\text{-path-cuts } T_{G-z}, \text{ for all } z \in N \setminus \{s, t\}, \quad (4)$$

$$x(e) \leq 1 \quad \text{for all } e \in E, \quad (5)$$

$$x(e) \geq 0 \quad \text{for all } e \in E. \quad (6)$$

Inequalities (1),(2),(3) and (4) are called *st-cut inequalities*, *st-node-cut inequalities*, *L-path-cut inequalities* and *L-path node-cut inequalities*, respectively. Inequalities (5) and (6) are called *trivial inequalities*.

2 Further Valid Inequalities

In this section, we describe further classes of inequalities that are valid for $P(G, L)$. As it will turn out, these inequalities define facets in some cases and reinforce the linear relaxation of the integer programming formulation presented in the previous section.

In the following two theorems, we present two classes of inequalities that are valid for the TEHNDP polytope. As we will mention below, they are also valid for $P(G, L)$ when $L = 3$.

Theorem 2.1 [3]

Let $L = 3$ and $T = \{t_1, \dots, t_p\}$ be a subset of p destination nodes w.r.t. to a node s . Let $\pi = (V_0, V_1, \dots, V_p)$ be a partition of N such that $s \in V_0$, and $t_i \in V_i$, $i = 1, \dots, p$. Then, the inequality

$$x(\delta(V_0, \dots, V_p)) \geq \lceil 4p/3 \rceil \quad (7)$$

is valid for the two-edge connected hop-constrained network design problem.

Theorem 2.2 [1,5]

Let $L = 3$ and $\Pi = \{V_0^1, V_0^2, V_1, V_2, V_3, V_4\}$ be a partition of V such that $\pi = (V_0^1, V_0^2 \cup V_1, V_2, V_3, V_4)$ induces a 3-st-path-cut, and V_1 induces a valid st-cut in G . If $F \subseteq [V_0^2 \cup V_1 \cup V_4, V_2]$ is chosen such that $|F|$ is odd, then the inequality

$$\begin{aligned} x([V_0^1, V_1 \cup V_2 \cup V_3 \cup V_4]) + x([V_0^2, V_1 \cup V_3 \cup V_4]) + x([V_1, V_3 \cup V_4]) \\ + x([V_0^2 \cup V_1 \cup V_4, V_2]) \geq \lceil 3 - |F|/2 \rceil \end{aligned} \quad (8)$$

is valid for the two-edge connected hop-constrained network design problem.

Inequalities of type (7) are called *Rooted Partition inequalities* and inequalities of (8) are called *Double Cut inequalities*. Since every solution of the TNHNDP is also solution of the TEHNDP, Inequalities (7) and (8) are also valid for TNHNDP. As we will see in the next section, rooted partition inequalities define facets in some cases.

Now, in the following theorem, we introduce a new class of inequalities which allows to describe the optimal solutions of the TNHNDP in the case where the demands are rooted and when the edge weights satisfy the triangle inequalities.

Theorem 2.3 *Suppose that $L \geq 2$ and the weight of the edges of G satisfy the triangle inequalities. Let $u \in V$ be a Steiner node and $F \subseteq E$ an optimal solution of the rooted TNHNDP w.r.t. to these edges weights. If s is the root of the demands and $F \cap [s, u] = \emptyset$, then $\delta(u) \cap F = \emptyset$.*

Proof. See [2]. □

Corollary 2.4 *Consider the rooted TNHNDP and let s be the root of the demand set. When $L = 3$ and when the weight of the edges satisfy the triangle inequalities, then the incidence vector of any optimal solution of the rooted TNHNDP satisfies*

$$x(su) \geq \sum_{uv \in \delta(u) \setminus \{su\}} x(uv), \text{ for every Steiner node } u \in N. \quad (9)$$

Inequalities (9) are called *optimality constraints*. They are in polynomial number and, as we will see in Section 5, they are effective in solving the rooted TNHNDP.

3 Facets of the TNHNDP Polytope

In this section, we give necessary and sufficient conditions for inequalities (1)-(6). We also give sufficient conditions for inequalities (7) to define facets of $P(G, L)$. These conditions will be used later to derive efficient separation procedures.

First, we discuss the dimension of $P(G, L)$ when $L = 3$. An edge e is said to be *essential* if it belongs to an st -cut or a 3- st -path-cut of cardinality 2, or to an st -node-cut or a 3- st -path-node-cut of cardinality 1. We denote by E^* the set of essential edges of G . We have the following theorem.

Theorem 3.1 $\dim(P(G, 3)) = |E| - |E^*|$.

If $G = (N, E)$ is complete and $|N| \geq 4$, then $E^* = \emptyset$. In the remainder of this paper, G is a complete and simple graph with $|N| \geq 4$. Thus, $P(G, 3)$ is full dimensional. If G is not complete, one can make it complete and obtain an equivalent problem by adding the missing edges with sufficiently high weights.

Theorem 3.2 (i) Inequality $x(e) \leq 1$ defines a facet of $P(G, 3)$ for all $e \in E$.

(ii) Inequality $x(e) \geq 0$ defines a facet of $P(G, 3)$ if and only if either

- (a) $|N| \geq 5$ or
- (b) $|N| = 4$ and $e = uv$ with $\{u, v\} \in N \setminus \{s, t\}$.

Theorem 3.3 Every st -cut inequality and every st -node-cut inequality other than those induced by $\{s\}$ or $\{t\}$ defines a facet of $P(G, 3)$.

Theorem 3.4 Every L -path-cut and L -path-node-cut inequality defines a facet of $P(G, 3)$ if and only if $|V_0| = 1$ and $|V_{L+1}| = 1$.

Theorem 3.5 The rooted partition inequality (7) induced by a partition $\pi = (V_0, V_1, \dots, V_p)$ defines a facet of $P(G, 3)$ if $|V_i| = 1$ for all $i = 1, \dots, p$ and p is not a multiple of 3.

4 Separation Procedures

In order to develop our Branch-and-Cut algorithm, we devise separation algorithms for Inequalities (1)-(4) and Inequalities (7) and (8).

As showed in [1], Inequalities (1)-(4) can be separated in polynomial time by computing a maximum flow in a special directed graph. For double cut inequalities, we use the separation heuristic developed by [1].

In the following, we describe a separation heuristic for the rooted partition inequalities when $L = 3$ and $|V_i| = 1$, for all $i \in \{1, \dots, p\}$, and p is not multiple of 3. By Theorem 3.5, they define facets in this case. Our separation algorithm in this case is based on the reduction of the separation problem to that of finding a maximum flow with lower bounds in a special graph.

First, we can easily see that Inequality (7) is equivalent to

$$x(\delta(V_0)) + \sum_{u \in N \setminus V_0} x(\delta(u)) \geq \frac{8}{3}p + \alpha \quad (10)$$

with

$$\alpha = \begin{cases} \frac{4}{3} & \text{if } p = 3q + 1, \text{ for some } q \in \mathbb{N} \\ \frac{2}{3} & \text{if } p = 3q + 2, \text{ for some } q \in \mathbb{N}. \end{cases}$$

Since $p = |V \setminus V_0|$ and $x(\delta(V_0)) = x(\delta(N \setminus V_0))$, Inequality (10) is equivalent to

$$x(\delta(N \setminus V_0)) \geq \sum_{u \in N \setminus V_0} y_u + \alpha \quad (11)$$

where $y_u = \frac{8}{3} - x(\delta(u))$, for all $u \in N \setminus V_0$.

Therefore, if \bar{x} is a solution of \mathbb{R}^E , then there exists a violated rooted partition inequality induced by a partition $\pi = (V_0, V_1, \dots, V_p)$ with $|V_i| = 1$ for all $i = 1, \dots, p$, if and only if the inequality (11) induced by π is violated by \bar{x} .

The separation problem of the rooted partition inequalities in this case then reduces to finding a set of terminals W for which the corresponding inequality (11) is violated by \bar{x} . This can be done by computing a feasible flow with lower bounds in a special directed graph obtained from G and \bar{x} and can be implemented in polynomial time. For more details, the reader can refer to [2].

5 Branch-and-Cut and Computational Results

Based on the results described in previous sections, we have developed a Branch-and-Cut algorithm to solve the TNHNDP when $L = 3$. The algorithm has been implemented in C++, using ABACUS 3.2 to manage the Branch-and-Cut tree and CPLEX 12.2 as linear solver. It was tested on a Xeon Quad-Core E5507 machine at 2.27 GHz with 8GB RAM, running under Linux. The maximum CPU time has been fixed to 5 hours.

The test problems are composed of complete graphs from TSPLIB (with euclidean edge weights). The demands are randomly generated. Each set of demand is either rooted at the same node s or arbitrary having multiple sources and destinations.

We use in our Branch-and-Cut algorithm all the different inequalities we have presented above, except in the non-rooted case where we do not use optimality constraints (since they are not valid in this case). The separations of the different constraints are performed in the following order: (i) st -cut inequalities, (ii) 3- st -path-cut inequalities, (iii) st -node-cut inequalities, (iv)

3-*st*-path-node-cut inequalities, (v) rooted-partition inequalities, (vi) double cut inequalities.

Table 1 below presents the results obtained for instances with graphs having up to 76 nodes. For each instance, we give the type of the demand set ("a" for arbitrary, "r" for rooted), the number of nodes $|N|$ and the number of demands $|D|$, the number of generated constraints (NC and NNC for *st*-cut and *st*-node-cut inequalities, LPC and LPNC for 3-*st*-path-cut and 3-*st*-path-node-cut inequalities, RP and DC for rooted partition and double cut inequalities), the relative error, in percentage, between the best upper bound and the best lower bound obtained at the root node of the Branch-and-Cut tree (resp., the best lower bound obtained over all the Branch-and-Cut tree) Gap1 (resp. Gt1). We also give Gap2 and Gt2 which are the gaps (as defined before) achieved when the optimality constraints, the rooted partition and double cut inequalities are not used in the Branch-and-Cut algorithm. We finally give the number of nodes in the Branch-and-Cut tree and the total CPU time in hours:min.sec. Remark that a value of 0 for Gt1 and Gt2 indicates that the upper bound obtained is optimal.

Table 1
Results for real instances when $L = 3$

$ N $	$ D $	NC	NNC	LPC	LPNC	RP	DC	Gap1	Gap2	Gt1	Gt2	Tree1	Tree2	CPU1	CPU2
a 14	10	45	13	5978	517	0	23	14.7	14.7	0	0	563	575	0:01:19	0:01:29
a 14	7	4	0	191	0	0	0	2.14	2.14	0	0	9	9	0:00:01	0:00:01
r 14	10	37	15	257	15	4	0	3.58	9.5	0	0	47	171	0:00:01	0:00:05
r 14	7	17	7	56	6	1	0	3.44	10.38	0	0	21	53	0:00:01	0:00:01
a 17	45	475	72	43186	138487	0	43	24.5	30.65	5.46	14.97	1639	795	5:00:00	5:00:00
a 17	8	32	6	17330	694	0	80	14.18	14.18	0	0	1801	2259	0:14:01	0:20:03
r 17	16	74	33	9522	1659	52	0	8.22	16.21	0	4.63	1915	949	0:07:36	5:00:00
a 30	10	78	2	3108	35	0	0	6.19	6.19	0	0	97	97	0:00:52	0:00:57
a 30	15	19	2	52744	13	0	0	43.97	43.97	42.91	42.91	101	101	5:00:00	5:00:00
r 30	10	318	751	509	72	8	0	6.38	9	0	2.98	443	481	0:01:17	5:00:00
r 30	15	778	1435	14098	1726	92	0	11.53	29.28	0	19.24	7411	3007	1:30:53	5:00:00
a 48	10	111	18	76852	18	0	0	42.14	42.14	40.01	40.01	13	13	5:00:00	5:00:00
a 48	15	68	0	28895	0	0	0	58.9	58.9	58.79	58.79	7	7	5:00:00	5:00:00
r 48	10	2425	1411	641	139	18	0	8.1	29.59	0	27.81	387	15	1:05:13	5:00:00
r 48	15	631	897	4001	601	55	0	9.62	45.07	0	40.96	1649	45	0:43:42	5:00:00
a 52	10	202	4	41755	11	0	0	19.15	19.15	15.87	15.87	35	35	5:00:00	5:00:00
a 52	20	57	2	73270	0	0	0	56.35	56.35	56.11	56.11	35	35	5:00:00	5:00:00
r 52	10	1188	2583	562	81	8	0	6.38	25.68	0	23.34	451	35	1:36:51	5:00:00
r 58	20	183	215	27780	3125	91	0	22.27	∞	16.5	∞	2339	1	5:00:00	5:00:00
r 58	30	69	31	21317	284	38	0	41.42	71.87	38.9	71.87	823	3	5:00:00	5:00:00
r 58	40	244	9	23881	127	23	0	71.88	74.87	70.76	74.75	215	7	5:00:00	5:00:00
r 76	20	26	46	14097	2288	37	0	27.72	45.03	23.68	43.33	1635	17	5:00:00	5:00:00
r 76	40	14	1	12171	116	10	0	53.88	∞	53.01	∞	305	1	5:00:00	5:00:00

Table 1 shows that for the test problems used in these experiments, 12 instances over 23 have been solved to optimality. For the instances solved to optimality, the CPU1 time varies from 1 sec to 1h36min. We notice that a small number of double cut inequalities are generated and for almost all the instances, the rooted partition inequalities are generated. Also, we notice that for the instances which are not solved to optimality, the gaps (Gap1 and Gap2)

are relatively high while the number of nodes in the Branch-and-Cut tree is relatively small. Also, we notice that a large number of basic inequalities are generated. This let us suppose that the algorithm spends a lot of time in the separation of the different inequalities and does not have enough time to explore more solutions in the Branch-and-Cut tree.

We have also checked the efficiency of the different constraints presented in this paper, especially for rooted partition inequalities and optimality constraints. For this, we have tried to solve the problem without rooted partition and double cut inequalities and optimality constraints. In this case, we remark that the efficiency of the algorithm is significantly decreased. For some instances, we do not have the optimal solution after 5 hours when these constraints are removed while the algorithm is able to obtain the optimal solution, in short time, when they are used (see for example (r17,16) and (r30,10)). Moreover, for some instances (see (r58,20) and (r76,40)), the algorithm spends all the time at the root node when the optimality constraints are removed. This shows the efficiency of the additional inequalities, especially the rooted partition inequalities and the optimality constraints, in solving the problem.

Acknowledgments

The research of the second author has been supported by a TUBITAK-BIDEB fellowship.

References

- [1] Diarrassouba I., Survivable Network Design Problems with High Connectivity Requirements, PhD Thesis, Université Blaise Pascal, France, 2009.
- [2] Diarrassouba I., Kutucu H. and Mahjoub A.R., Two Node-Disjoint Hop-Constrained Survivable Network Design and Polyhedra, Technical Report, Cahier de Recherche LAMSADE 332, France, 2013.
- [3] Huygens D., Labb   M., Mahjoub A.R., Pesneau P., The two-edge connected hop-constrained network design problem: Valid inequalities and branch-and-cut. Networks 49(1) (2007), 116-133.
- [4] Huygens D. and Mahjoub A.R., Integer programming formulations for the two 4-hop-constrained paths problem, Networks 49 (2007), 135-144
- [5] Huygens D., Mahjoub A.R. and Pesneau P., Two edge-disjoint hop-constrained paths and polyhedra, SIAM J Disc Math 18 (2004), 287-312.

The track formulation for the Train Dispatching problem

Leonardo Lamorgese, Carlo Mannino¹

*Applied Mathematics
SINTEF ICT
Oslo, Norway*

Abstract

With few exceptions, train movements are still controlled by human operators, the dispatchers. They establish routes and precedence between trains in real-time in order to cope with normal operations but also to recover from deviations from the timetable, and minimize overall delays. Implicitly they tackle and solve repeatedly a hard optimization problem, the Train Dispatching Problem. We recently developed a decomposition approach which allowed us to solve real-life instances to optimality or near optimality in times acceptable for dispatchers. We present here some new ideas which appear to significantly reduce computational times while solving to optimality even large instances.

Keywords: Rescheduling, dispatching, disjunctive formulation

1 Introduction

Railways all over Europe are becoming ever more congested and punctuality and reliability are rapidly deteriorating. Still, in Europe the volume of people

¹ Email:[leonardo.lamorgese\(carlo.mannino@sintef.no](mailto:leonardo.lamorgese(carlo.mannino@sintef.no)

and freight transported on railway is increasing (see [3]), increasing congestion evermore. A fast and economically viable way to increase capacity is by improving the efficiency of daily operations so as to be able to control a larger number of running trains. Today, nearly all train traffic control is carried out "by hands", namely by human operators called *dispatchers*. When trains fail to follow the official timetable, dispatchers establish new routes and new schedules *in real-time* trying to restore the original timetable, or to approximate it as much as possible. We call the resulting optimization problem the *Train Dispatching Problem* (TD).

In a by now classical approach to this type of problems, the movements on the rail of each train are represented as a path (*route*). The coordinated movements of all trains are then further represented in a single graph which contains all routes. To avoid conflicts on the use of the resource, decisions on "which train goes first" are represented in this model by *disjunctive arcs* (see [6]). The TD problem is finally represented as a (typically very large) disjunctive optimization problem. The latter may be very hard to solve in practice and typically one resorts to heuristic approaches.

In [4],[5] we introduce a decomposition approach which allows to solve the TD problem to optimality (or nearly) for a number of real-life instances from the Italian railway system. We identify two sub-problems, the Line Dispatching problem (LD) and the Station Dispatching problem (SD), which naturally arise when controlling a railway line. Both are modeled as Mixed Integer Linear Programs (MILPs) and the overall TD problem is solved in a master-slave fashion. In particular, the Line Dispatching problem acts as the master problem which is solved iteratively; at each iteration a solution to the Station Dispatching problem is generated (for every station of the line) and, if necessary, new constraints (in the variables of the master problem) are added to the Line Dispatching problem which is then resolved again. This approach allows to drastically reduce the number of variables with respect to the "general" disjunctive formulation and model station layouts with increasing complexity.

In [4], [5] the major decision variables of the master problem are associated with every pair of trains and every point on the line where trains can meet or pass each other. This choice mimics the actual behaviour of dispatchers which take such decisions in order to avoid conflicts. In this paper we introduce a new modeling approach which allowed us to further reduce the computational effort and solve previously unsolved instances. In particular, the new variables are somehow complementary to the original ones, in the sense that they model where a pair of trains *will not* meet on the line. We assume that the railway resources of a line can be ordered so that any train runs through either

(monotonically) increasing or (monotonically) decreasing resources. It follows that, for any given pair of trains, if they do *not* meet in a specific region p , then one of the following conditions must be satisfied i) the trains meet before p ; ii) the trains meet after p . This simple idea actually allowed for several algorithmic enhancements. In particular, we devised a delayed row generation mechanism which is a natural extension of this definition and corresponds to generating recursive partitions of the line. Preliminary computational results appear very promising.

2 The Train Dispatching problem on single-track lines

Railway networks can contain very complex stations and several parallel tracks between stations, plus other infrastructures such as sidings and cross-overs. However in this paper we focus on single-tracks lines, where stations are connected by one track and trains in both directions will alternate on it. To understand the relevance of such lines, consider (e.g.) that in Italy they represent almost 60% of the entire network, and in Norway almost 95%! We also use a simple model for the stations, which applies in most cases and to all stations considered in our real-life test-bed. In any case, the concepts here discussed for single-track lines and simple stations can be readily extended to multiple tracks and more complex stations.

2.1 Single-Track lines

A single-track line is an alternating sequence of stations and tracks. Let $S = \{1, \dots, q\}$ be the set of stations and $B = \{1, \dots, q - 1\}$ be the set of tracks, with track k connecting station k and station $k + 1$. At most one train at the time can occupy a track, while each station $s \in S$ can accommodate up to c_s trains, where c_s is the number of platforms in the station (or *station capacity*).

Let $R = S \cup B$ be the set of *railway resources* (in our case stations and tracks) and let T be the set of trains. The route of train $i \in T$ is a sequence of stations and tracks, namely a directed path $G^i = \{v_1^i, (v_1^i, v_2^i), \dots, (v_{l(i)-1}^i, v_{l(i)}^i), v_{l(i)}^i\}$, with node set V^i and arc set A^i . Node $v \in V^i$ represents the occupation of a resource in R , either a station or a track. Arc $(u, v) \in A^i$ models that train i visits u right before v and we let $L_{u,v}^i \geq 0$ be the minimum time to go from u to v . So, if u is a station (node), then v is the next track on the route and $L_{u,v}^i$ is the minimum time train i spends in the station before departing. If u is a track (node), then L_v^i is the track's running time for train i .

2.2 The graph of routes

The train routes are exploited in the construction of a new "logical" graph, which represents the movements of all trains. We call this graph the *graph of routes* $G^T = (V, A)$, which is the union of each route graph G^i plus an additional *root* node. Namely, we let $V = \{r\} \cup \{v \in V^i : i \in T\}$ and $A = \{(r, v_1^i), i \in T\} \cup \{(u, v) \in A^i : i \in T\}$. The new node r is a source, connected to the first node of each train route G^i , and is used to model the start of the time horizon. Now, we introduce the *schedule* $t \in R_+^V$. If $v \in V$ corresponds to node v_k^i in the route of train i , then t_v is the earliest time at which train i can reach the k -th resource on its path. Also, we let $t_r = 0$. Indeed, if v corresponds to a station on route G^i , then t_v represents the earliest *arrival time* for train i in such station. Similarly, if v is a track on route G^i , then t_v represents the *departing time* from the station that precedes the track on G^i . With every possible schedule t we associate a cost function $c(t)$, which we chose (after discussions with railway engineers) to be a convex and piecewise linear function of arrival delays, so that larger delays have non-decreasing marginal cost. We are now able to state more formally the Train Dispatching problem for single-track lines:

Problem 2.1 *Given a set of stations S , a set of tracks B , and a set of trains T find a schedule t for the trains in T so that $c(t)$ is minimized, no two trains occupy simultaneously the same track and no more than c_s trains meet in station $s \in S$.*

3 Modelling the Train Dispatching problem

We assume hereby that the set of single-track railway resources $R = S \cup B$ can be ordered so that any train runs through either (monotonically) increasing or (monotonically) decreasing resources². Given trains $i, j \in T$, we will refer to i meeting j *before* (after) a given railway resource $r \in R$ if i and j meet in any railway resource $r' \in R$ such that $r' \prec r$ ($r' \succ r$). Accordingly, we have an increasing direction (when trains run from smaller resources to larger ones) and a decreasing direction.

We model now that trains cannot meet on a track b , that is they meet either before b or after b . Let train i run in increasing direction, and train j run in decreasing direction. Assume they meet before track b . Then train

² This is always the case in single-track lines. However, for more complex lines, we may content ourselves with a milder assumption as, indeed, it suffices that, for any given pair of trains, the property holds for the common sub-network

j arrives in station b before train i leaves it, and the precedence constraint $t_v - t_u \geq 0$ holds, where t_v and t_u are, respectively, the exit and arrival times of i and j in station b . Similarly, the constraint $t_q - t_p \geq 0$ holds if i and j meet after b , with t_q and t_p being, respectively, the exit and arrival times of j and i in station $b + 1$. So, every feasible schedule must satisfy the disjunction: $t_v - t_u \geq 0 \vee t_q - t_p \geq 0$. Similar disjunctions hold for pair of trains running in other combinations of directions. Disjunctive precedence constraints can be represented on the graph of routes G^T by a pair of directed arcs $\{\alpha(i, j, s), \beta(i, j, s)\}$, where $\alpha(i, j, b) = (p, q)$ and $\beta(i, j, b) = (u, v)$, with $p, q, u, v \in V$: such pair is called *disjunctive arc* (See ([1])).

For every pair of distinct trains $\{i, j\}$ and every track $b \in B$ we introduce a binary variable w_{ij}^b which is 1 if i and j meet *before* b and 0 if i and j meet *after* b . It is well known that by introducing a suitable large coefficient M the above disjunction can be expressed by a (conjunctive) pair of linear constraints $t_v - t_u \geq M(w_b^{ij} - 1)$ and $t_q - t_p \geq -Mw_b^{ij}$.

Now, recall that no more than c_s trains can meet simultaneously in a station $s \in S$. To express this constraint we introduce for every pair of distinct trains i, j and every station $s \in S$ a binary variable y_s^{ij} which is 1 if i and j meet in s and 0 otherwise. In [5] we show that the station capacity constraint is satisfied if and only if, for every subset of trains $K \subseteq T$, with $|K| = c_s + 1$, the following constraint holds:

$$(1) \quad \sum_{\{i,j\} \subseteq K} y_s^{ij} \leq \frac{1}{2}(c_s + 1)c_s - 1$$

Clearly, trains i and j meet in $s \in S$ if and only if they meet after track $s - 1$ and before track s . This can be expressed by the linear constraint $y_s^{ij} \geq w_{ij}^s - w_{ij}^{s-1}$. So, the TD problem for single-track lines writes as shown in (2).

Note that since $c(t)$ is piece-wise linear and convex, it can be easily linearized. The formulation can be strengthened in various ways. First, observe that if train i and j meet before b , then they meet before $b + 1, b + 2, \dots$. Then, for $\{i, j\} \subseteq T$ and $k \in B$, we have the following valid constraints (not obtainable as conic combinations of other constraints): $w_{ij}^k - w_{ij}^{k-1} \geq 0$. Also, since $w_{ij}^k - w_{ij}^{k-1} \geq 0$, then it is not difficult to see that for every feasible integral w, y we have $y_k^{ij} = w_{ij}^k - w_{ij}^{k-1}$. Then the variable y_k^{ij} can be dropped from the formulation. Again, observe that this equality holds necessarily only for integral values, that is the equation is not directly implied by the other linear inequalities.

The above MILP is the basis of our master-slave algorithm for solving the

$$\begin{aligned}
\min \quad & c(t) \\
s.t. \quad & \\
(i) \quad & t_v - t_u \geq L_{uv} \quad (u, v) \in A \\
(ii) \quad & t_v - t_u \geq M(w_b^{ij} - 1) \quad \{i, j\} \subseteq T, b \in B, (u, v) = \beta(i, j, b) \\
(2) \quad (iii) \quad & t_q - t_p \geq -Mw_b^{ij} \quad \{i, j\} \subseteq T, b \in B, (p, q) = \alpha(i, j, b) \\
(iv) \quad & y_s^{ij} \geq w_{ij}^s - w_{ij}^{s-1} \quad \{i, j\} \in K, s \in S \setminus \{1\} \\
(v) \quad & \sum_{\{i, j\} \subseteq K} y_s^{ij} \leq \frac{1}{2}(c_s + 1)c_s - 1 \quad s \in S, K \subseteq T, |K| = c_s + 1 \\
(vi) \quad & t \in \mathbb{R}^V, w, y \text{ binary}
\end{aligned}$$

TD problem, and is solved by delayed row generation. Namely, we start by solving a problem with a subset of the overall constraints and we generate more constraints only if necessary. We follow a path somehow intermediate between the classical Benders's decomposition algorithm (see, e.g., [7]) and the combinatorial variant suggested by Codato and Fischetti[2]. We limit to separating constraints from (ii) to (v) only in branching nodes corresponding to integer solutions. In this way, the separation of (v) becomes easy (see [5]) as it reduces to computing a maximum clique on an interval graph.

4 Preliminary Computations

Our tests were run on an Intel(R) Core(tm) i7-2640M CPU 870 2.80GHz machine and we used the commercial solver ILOG CPLEX. Although a very preliminary implementation, results show that this approach allows to successfully tackle the Train Dispatching problem for single-track instances and find solutions within the stringent time window required by a real-time application. In our computations we solved real-life instances from the Trento-Bassano (T-BG), Terontola-Foligno (T-F), Foligno-Orte (F-O) and Falconara-Foligno (F-F) lines in Italy. In Table 1 we confront the performance of the new formulation against the precedent "station meeting" formulation (see [5]) for nine instances of T-BG. *TC* stands for *Track Conflict*, while *SM* stands for *Station Meeting*. Results show that TC appears competitive with the former. Indeed, computation time is reduced significantly, as computing optimal solutions for TC requires separating and adding, in the general case, a far smaller set of

constraints.

ID	#Trains	Initial Rows	Rows Generated		Computation Time(s)	
			SM	TC	SM	TC
1	29	1333	695	13	12,4	0,78
2	28	1302	168	6	1,84	0,33
3	29	1351	68	12	1,64	0,37
4	29	912	354	13	3,90	0,21
5	28	1302	615	8	38,11	0,45
6	18	798	190	6	1,46	0,15
7	19	796	2307	52	14,62	0,24
8	19	796	1039	24	1,67	0,19
9	28	1298	4815	230	>60s	13,63

Table 1
Preliminary computational results. The average time horizon is 9 hours.

In Table 2, we also compare our new approach with the dispatching algorithm currently in operation on the lines in Table 2 in terms of the average distribution of delayed trains³. Trains were clustered in macro-ranges according to the difference between expected and actual arrival time at destination.

As emerges from Table 2, in all cases the percentage of trains on time increased tangibly with respect to the current practice. This was not only due to slightly delayed trains arriving on time, but also to a clear decrease in the number of trains running severely late. Also, TC's performance is only very slightly affected by larger instances, as it is still able to produce, in the vast majority of cases, high quality solutions within the time limit. In conclusion, these preliminary results appear very promising and confirm evermore the potential impact of effective exact methods on Train Dispatching.

³ Actually, two of these are mixed single/double-track lines but the extending the model was straightforward

Line	#	Horizon (hrs)	Trains	On time		Late between 3 and 6 mins		Later than 6 mins	
				H	TC	H	TC	H	TC
F-O	318	12	43	85%	94%	9%	2%	6%	4%
T-BG	365	9	31	64%	86%	15%	4%	21%	10%
T-F	634	11	32	65%	90%	25%	7%	10%	3%
F-F	834	12	44	75%	86%	10%	7%	15%	7%

Table 2

Punctuality distribution for four lines in Italy. Average figures. *H* stands for *Heuristic*. Column "#” refers to the number of instances solved

References

- [1] Balas, E., Machine sequencing via disjunctive graphs, *Operations Research* 17 (1969) pp. 941–957.
- [2] Codato G., Fischetti M., *Combinatorial Benders' Cuts for Mixed-Integer Linear Programming*, *Operations Research*, 54 (4), pp. 756-766, 2006.
- [3] Corman F., *Rail-time railway traffic management: dispatching in complex, large and busy railway networks*, Ph.D. Thesis, TRAIL Thesis Series T2010/14, the Netherlands TRAIL Research School.
- [4] C. Mannino, Real-time traffic control in railway systems, *Proceedings of Atmos'11*, A. Caprara and S. Kontogiannis (Eds.), OASIcs Vol. 20, 2011
- [5] L. Lamorgese, C. Mannino, An exact decomposition approach for the real-time train dispatching problem, Technical Report N. A23274, SINTEF ICT, Norway, 2012, submitted.
- [6] Mascis A., D. Pacciarelli, *Job shop scheduling with blocking and no-wait constraints*, *European Journal of Operational Research*, 143 (3), pp. 498–517, 2002.
- [7] Nemhauser G.L. and Wolsey L.A., *Integer and Combinatorial Optimization*, Wiley-Interscience, 1999.

Martin Skutella		Flows Over Time: Modeling Network Routing Problems including a Temporal Dimension	Monday	09:00-09:50	1
Paolo Serafini		Computing the probabilities of small component sizes in large networks	1	Monday	10:00-11:30
Pablo Adasme	Rafael Andrade, Marc Letournel, Abdel Lisser	A polynomial formulation for the stochastic maximum weight forest problem	1	Monday	10:00-11:30
Bernd Zey	Ivana Ljubić, Petra Mutzel	Stochastic Survivable Network Design Problems	1	Monday	10:00-11:30
Michael Poss		Robust combinatorial optimization with uncertain cost	1	Monday	10:00-11:30
Alessia Violin	Bernard Fortz, Martine Labbé	Dantzig-Wolfe Reformulation for the Network Pricing Problem with Connected Toll Arcs	3	Monday	10:00-11:30
Ariel Waserhole	Vincent Jost, Nadia Brauner	Pricing techniques for self regulation in Vehicle Sharing Systems	3	Monday	10:00-11:30
E.Malaguti	I. Méndez-Díaz, J.J. Miranda-Bront, P.Zabala	(k, c)- coloring via Column Generation	3	Monday	10:00-11:30
Andrei V. Orlov		Pricing in telecommunication networks and bilevel optimization	3	Monday	10:00-11:30
J. Fabian Meier	Uwe Clausen	Strategic planning in LTL logistics - increasing the capacity utilization of trucks	18	Monday	10:00-11:30
Jessica Raffaelli	Alberto Coppi, Paolo Detti	A planning and routing model for patient transportation in health care	18	Monday	10:00-11:30
Luigi De Giovanni	Claudio E. Palazzi	Optimal Client-Server Configuration of Mobile Ad-Hoc Networks	18	Monday	10:00-11:30
Maria Grazia Scutellà	Paola Cappanera	Home Care optimization: impact of pattern generation policies on scheduling and routing decisions	18	Monday	10:00-11:30
Claudia Stangl	Ralf Gollmer, Rüdiger Schultz	Checking Feasibility in Stationary Models of Gas Transportation Networks - Methodological Foundation	7	Monday	12:00-13:30
Ralf Gollmer	Rüdiger Schultz, Claudia Stangl	Checking Feasibility in Stationary Models of Gas Transportation Networks - Case Study	7	Monday	12:00-13:30
David Raz	Shira Levy	A Study of the Connectivity Over Time Behavior of On-Demand Ad Hoc Path Selection Schemes	7	Monday	12:00-13:30
Martin Tieves	Arie Koster, Truong Khhoa Phan	Extended Cutset Inequalities for the Network Power Consumption Problem	7	Monday	12:00-13:30
Okan Dükancı	Bahar Yetiş Kara	Routing and Scheduling Decisions in the Hierarchical Hub Location Problem	8	Monday	12:00-13:30
A. İrfan Mahmutoğulları	Bahar Y. Kara	Hub Location Problem under Competition: (r p) hub-medianoid and (r p) hub-centroid problems	8	Monday	12:00-13:30
Julia Sender	Uwe Clausen	Heuristics for solving a capacitated multiple allocation hub location problem with application in German wagonload traffic	8	Monday	12:00-13:30
Hande Yaman	Inmaculada Rodríguez Martín, Juan José Salazar González	A Branch-and-cut Algorithm for the Hub-Cycle Location Problem	8	Monday	12:00-13:30
Sara Mattia		The cut condition for robust network design	21	Monday	12:00-13:30
Cédric Hervet	Aline Faye, Matthieu Chardy, Marie-Christine Costa, Stanislas Francfort	Solving the Two-Stage Robust FTTH network design Problem under Demand Uncertainty	21	Monday	12:00-13:30
Jiangqian Cheng	Abdel Lisser, Marc Létoumel	Distributionally robust stochastic shortest path problem	21	Monday	12:00-13:30
Walid Ben-Ameur	Mateusz Ziółkiewicz	Multipolar routing: where dynamic and static routing meet	21	Monday	12:00-13:30
Fabien Cornillier		Steiner tree network scheduling with opportunity cost of time	9	Monday	15:30-17:00
R. Taktak	S. Bone, A.R. Mahjoub	A branch-and-cut algorithm for the Multiple Steiner TSP with Order constraints	9	Monday	15:30-17:00
Jannik Matuschke	Andreas Bley, Benjamin Müller	Approximating the capacitated facility location problem with length bounded trees	9	Monday	15:30-17:00
Alexandre Salles da Cunha	Bernard Gendron, Abilio Lucena, Luidi Simonetti	The Degree Preserving Spanning Tree Problem: Valid Inequalities and Branch-and-cut method	9	Monday	15:30-17:00
Markus Leitner	Luis Gouveia, Ivana Ljubić	The Two-Level Diameter Constrained Spanning Tree Problem	10	Monday	15:30-17:00
Angelo Sifaleras	George Gerani	Dynamic trees in exterior-point Simplex-type algorithms for network flow problems	10	Monday	15:30-17:00
Dilson Lucas Pereira	Michel Gendreau, Alexandre Salles da Cunha	Stronger Lower Bounds for the Quadratic Minimum Spanning Tree Problem with Adjacency Costs	10	Monday	15:30-17:00
Elena Fernández	Carlos Luna-Mota, Achim Hildenbrandt, Gerhard Reinelt, Stefan Wiesberg	A Flow Formulation for the Optimum Communication Spanning Tree	10	Monday	15:30-17:00
Christina Büsing	Fabio D'Andreagiovanni	Multi-Band Robustness I: Model and Theory	6	Monday	15:30-17:00
Fabio D'Andreagiovanni	Christina Büsing, Arie M.C.A. Koster, Manuel Kutschka	Multi-Band Robustness II: Constructing the Uncertainty Set	6	Monday	15:30-17:00
Manuel Kutschka	Fabio D'Andreagiovanni, Arie M.C.A. Koster, Christina Büsing	Multi-Band Robustness III: Application to Network Design Problems	6	Monday	15:30-17:00
Grit Claßen	Arie M.C.A. Koster, Manuel Kutschka, Anke Schmeink	Traffic Node Assignment in Wireless Networks: A Multi-Band Robust Knapsack Approach	6	Monday	15:30-17:00
S. Consoli	J. A. Moreno Pérez, N. Mladenović	Intelligent variable neighbourhood search for the minimum labelling spanning tree problem	11	Monday	17:30-19:00
Cristina Requejo	Agostinho Agra, Luidi Simonetti	Enhanced dynamic programming algorithms for the constrained subtree of a tree problem	11	Monday	17:30-19:00
Rafael Castro de Andrade	Adriano Tavares de Freitas	Disjunctive combinatorial branch in a subgradient tree algorithm for the DCMST problem with VNS-Lagrangian bounds	11	Monday	17:30-19:00
S. Raghavan	Mustafa Sahin	Local Search for the Reload Cost Spanning Tree Problem	11	Monday	17:30-19:00
Maren Martens	Markus Chimani, Maria Kandyba	2-InterConnected Facility Location: Specification, Complexity, and Exact Solutions	14	Monday	17:30-19:00
Axel Werner	Markus Leitner, Ivana Ljubić, Markus Sinnl	On the Two-Architecture Connected Facility Location Problem	14	Monday	17:30-19:00
Ashwin Arulselvan	Olaif Maurer, Martin Skutella	An incremental algorithm for the uncapacitated facility location problem	14	Monday	17:30-19:00
Andreas Bley	M. Rezapour, S. M. Hashemi	IP Modeling of the survivable hop constrained connected facility location problem	14	Monday	17:30-19:00
Lehüedé Fabien	Pétron Olivier	A three step decomposition approach for a transportation network design problem with non-linear costs	24	Monday	17:30-19:00
Wataru Kishimoto		An Extension of a Balanced Flow in a Network	24	Monday	17:30-19:00
Amaro de Sousa	Carlos Lopes, Dorabella Santos	Loading balancing optimization of telecommunication networks with two differentiated services	24	Monday	17:30-19:00
Eduardo Amaldi	Stefano Coniglio, Luca G. Gianoli, Can Umut Ileri	On single-path network routing subject to max-min fair flow allocation	24	Monday	17:30-19:00
Martin Labbé		Network design problems in phylogenetics	Tuesday	09:00-09:50	1
Robert Voll	Uwe Clausen	Branch-and-Price for a European variant of the Railroad Blocking Problem	5	Tuesday	10:00-11:30
Zhiyuan Lin	Raymond S. K. Wan	An integer fixed-charge multicommodity flow (FCMF) model for train unit scheduling	5	Tuesday	10:00-11:30
Martin Philip Kidd	Fabio Furini	A fast heuristic approach for train timetabling in a railway node	5	Tuesday	10:00-11:30
Leonardo Lamorgese	Carlo Mennino	The track formulation for the Train Dispatching problem	5	Tuesday	10:00-11:30
Harmida Seba	Rashed Khennoufa	Edge Coloring by Total Labelings of 4-regular Circular Graphs	22	Tuesday	10:00-11:30
Mahdi Doostmohammadi	Agostinho Agra, Cid Carvalho de Souza	Intersecting a simple mixed integer set with a vertex packing set	22	Tuesday	10:00-11:30
Amal Benhamiche	A. Ridha Mahjoub, Nancy Perrot, Eduardo Uchoa	Capacitated Network Design using Bin-Packing	22	Tuesday	10:00-11:30
Faiz Hamid	Walid Ben-Ameur, Mohamad Assaad	An interval Assignment Problem for Resource Optimization in LTE Networks	22	Tuesday	10:00-11:30
Sibel Alumur	Stefan Nickel, Francisco Saldaña-da-Gama, Vedat Verter	Multi-period Reverse Logistics Network Design	25	Tuesday	10:00-11:30
Bradley Paynter	Douglas R. Shier	A Network Approach to a Geometric Packing Problem	25	Tuesday	10:00-11:30
Vinicio Leal do Forte	Abilio Lucena, Nelson Maculan	Formulations for the Minimum 2-Connected Dominating Set Problem	25	Tuesday	10:00-11:30
Abilio Lucena	Alexandre Salles da Cunha, Luidi Simonetti	Formulating and Solving the Minimum Dominating Cycle Problem	25	Tuesday	10:00-11:30
Yauhen Maisiuk	Irina Gribkovskaya	Fleet sizing for offshore supply vessels	2	Tuesday	12:00-13:30
Luis Cadalso	Ángel Marín	Integrated Airline Robust Scheduling and Fleet Assignment under Demand Uncertainty	2	Tuesday	12:00-13:30
Hanane Allouaya	Sylvie Borne, Lucas Létocart and Roberto Wolffert Castro	A heuristic approach for solving a home health care problem	2	Tuesday	12:00-13:30
Valentina Simini	Juan José Salazar González	A heuristic approach for an integrated fleet-assignment, aircraft-routing and crew-pairing problem	2	Tuesday	12:00-13:30
Güneş Erdogan	Gilbert Laporte, Roberto Wolffert Calvo	The One Commodity Pickup and Delivery Traveling Salesman Problem with Demand Intervals	12	Tuesday	12:00-13:30
Odivanez Pedro	Rodney Saldanha, Ricardo Camargo	A Tabu Search Approach for the Prize Collecting Traveling Salesman Problem	12	Tuesday	12:00-13:30
Nacima Labadie	H. Murat Afsar	Team Orienteering Problem with Decreasing Profits	12	Tuesday	12:00-13:30
Hipólito Hernández-Pérez	Juan José Salazar González	Heuristics procedures to solve the multi-commodity Pickup-and-Delivery Traveling Salesman Problem	12	Tuesday	12:00-13:30
İdil Arslık	F. Sibel Salman	Modeling Earthquake Vulnerability of Highway Networks	26	Tuesday	12:00-13:30
Richard Li-Yang Chen	Cynthia A. Phillips	k-Edge Failure Resilient Network Design	26	Tuesday	12:00-13:30
Marie-José Huguet	Dominik Kirchler, Pierre Parent, Roberto Wolffert Calvo	Efficient algorithms for the 2-Way Multi Modal Shortest Path Problem	26	Tuesday	12:00-13:30
Bauguion Pierre-Olivier	Ben Ameur Walid, Gourdin Eric	A new model for multicommodity flow problems, and a strongly polynomial algorithm for single-source Maximum Concurrent Flow	26	Tuesday	12:00-13:30
Jean-François Cordeau		Benders Decomposition for Hub Location	Wednesday	09:00-09:50	1
Pedro Patrício	Luis Gouveia, Amaro de Sousa	Lexicographical Minimization of Routing Hops in Telecommunication Networks	4	Wednesday	10:00-11:30
Yogesh K. Agarwal	Yash P. Aneja	Design of Fixed Charge Multi-commodity Networks using k-Partition Facets	4	Wednesday	10:00-11:30
Lena Hupp	Frauke Liers	A polyhedral study of the Hamiltonian p-median problem	4	Wednesday	10:00-11:30
Ibrahimia Diarrassouba	Hakan Kutucu, A. Ridha Mahjoub	Two Node-Disjoint 3-Hop-Constrained Survivable Network Design and Polyhedra	4	Wednesday	10:00-11:30
Artur Tomaszewski		The final answer to the complexity of a basic problem in resilient network design	16	Wednesday	10:00-11:30
Yuan Li	Michal Pióro	Integer programming models for maximizing parallel transmissions in wireless networks	16	Wednesday	10:00-11:30
F. Castaño	A. Rossi, M. Sevaux, N. Velasco	On the use of multiple sinks to extend the lifetime in connected wireless sensor networks	16	Wednesday	10:00-11:30
Giuliana Carello	Bernardette Addis, Sara Mattia	Energy-aware operations management in telecommunication network with shared protection mechanism	16	Wednesday	10:00-11:30
Stefano Nasini	Jordi Castro	Generating conditional uniform random networks by optimization procedures	28	Wednesday	10:00-11:30
Meltem Peker	Bahar Yetiş Kara	P-Hub Maximal Covering Problem and Extensions for Gradual Decay Functions	28	Wednesday	10:00-11:30
Halenur Sahin	Oya Ekin Karahan, Bahar Yetiş Kara	On Debris Removal During the Response Phase	28	Wednesday	10:00-11:30
Andreas Bärmann	Frauke Liers	Solving Network Design Problems via Iterative Graph Aggregation	28	Wednesday	10:00-11:30
F.V. Louveaux	J.J. Salazar-Gonzalez	Solving the Single Vehicle Routing Problem with Variable Capacity	15	Wednesday	12:00-13:30
Iris Martínez-Salazar	Francisco Angel-Bello, Ada Alvarez	Minimizing waiting times in a route design problem with multiple use of a single vehicle	15	Wednesday	12:00-13:30
Adrian Bock	Laura Sanitá	On the approximability of two capacitated vehicle routing problems	15	Wednesday	12:00-13:30
Javier Faulin	Angel A. Juan, Oscar Dominguez	A Multi-Start Approach for Optimizing Routing Networks with Vehicle Loading Constraints	15	Wednesday	12:00-13:30
Başak Yazar	Bahar Yetiş Kara, Oya Ekin Karahan	Fiber Optical Network Design Using Passive Splitters	17	Wednesday	12:00-13:30
Daniel Karch	Andreas Bley, Fabio D'Andreagiovanni	WDM Fiber Replacement Scheduling	17	Wednesday	12:00-13:30
Thiago F. Noronha	Daniel Morales dos Reis, Sérgio R. de Souza	Iterated Local Search for Fiber Installation in Optical Network Optimization	17	Wednesday	12:00-13:30
S.P. van Logerenberg	M.J. Grobler, S.E. Terblanche	Solving the Passive Optical Network with Fiber Duct Sharing Planning Problem Using Discrete Techniques	17	Wednesday	12:00-13:30
Mario Ruthmair	Luis Gouveia	Multi-dimensional layered graphs models for routing problems	27	Wednesday	12:00-13:30
Guerrero, W.J.	Velasco, N. Prodhon, C. Amaya, C.A.	On the Generalized Elementary Shortest Path Problem: A heuristic approach	27	Wednesday	12:00-13:30
Bernard Gendron	Rosario Garoppo, Gianfranco Nencioni, Maria Grazia Scutellà, Luca Tavanti	Benders Decomposition for a Location-Design Problem in Green Wireless Local Area Networks	27	Wednesday	12:00-13:30
Luis Gouveia	Juan José Salazar González	Polynomial-time separation of Enhanced Reverse Multistar Inequalities	27	Wednesday	12:00-13:30
Manuel Iori	Stefano Novellani, Barbara Panucci, Christian Bogenberger, Mauro Dell'Amico, Gerhard H. Ehwartz	Optimization Models for the Construction of a Large Highway System	13	Wednesday	15:30-17:00
Marcus Sinnl	Markus Leitner, Ivana Ljubić	Solving the bi-objective price-collecting Steiner tree problem with the epsilon-constraint method	13	Wednesday	15:30-17:00
Silvia M. D. M. Maia	Elizabeth F. G. Goldbarg, Marco C. Goldbarg	On the Biobjective Adjacent Only Quadratic Spanning Tree Problem	13	Wednesday	15:30-17:00
Jorge Riera-Ledesma	María Batista-Gálván, Manuel Iori	The Double Vehicle Routing Problem with Multiple Stacks: Exact Approaches	13	Wednesday	15:30-17:00
Pablo Romero	Franco Robledo, Pablo Rodríguez-Bocca, Claudia Rastognol	Mathematical Analysis of caching policies and cooperation in YouTube-like services	19	Wednesday	15:30-17:00
Guillaume Sagnol	Ralf Borndörfer, Julia Buwaya, Elmar Swarat	Optimizing Toll Enforcement in Geometric Transportation Networks: a Game-Theoretic Approach	19	Wednesday	15:30-17:00
Frédéric Lassabe	Qing Xu, Hakim Mabed, Alexandre Caminade	Fitness Landscape Analysis for Scalable Multicast RRM Problem in Cellular Network	19	Wednesday	15:30-17:00
Federico Perea	Juan A. Mesa, Gilbert Laporte	Introducing new stations on a high-speed railway corridor competing with another transportation mode	19	Wednesday	15:30-17:00
Dilek Güneş	S. Raghavan, Rui Zhang	The Least Cost Influence Problem	20	Wednesday	17:30-19:00
İssam Tahiri	Frédéric Giroire, Stéphane Pérennes	On the Hardness of Equal Shortest Path Routing	20	Wednesday	17:30-19:00
Merabet Massinissa	Sylvain Durand, Miklos Molnar	Exact solution for branch vertices constrained spanning problems	20	Wednesday	17:30-19:00
I. Diarrassouba	V. Gabrel, L. Gouveia, A. R. Mahjoub, P. Pesneau	Integer Programming Formulations for the k-Edge-Connected 3-Hop-Constrained Network Design Problem	20	Wednesday	17:30-19:00
Olaf Maurer	Andreas Bley, Ivana Ljubić	Lagrangian Decompositions of the Two-Level FTTH Network Design Problem	23	Wednesday	17:30-19:00
Alessandra Cornoaro	Gian Paolo Clemente	Effective MILP formulation for optimal 1-for-N diversity coding	23	Wednesday	17:30-19:00
Monia Giandomenico	Adam N. Letchford, Fabrizio Rossi, Stefano Smriglio	A New Lower Bound for the Kirchhoff Index using a numerical procedure based on Majorization Techniques	23	Wednesday	17:30-19:00
		Approximating the Lovász theta Function with the Subgradient Method	23	Wednesday	17:30-19:00