Alma Mater Studiorum - Università di Bologna
DEI - Dipartimento di Ingegneria dellEnergia Elettrica e dellInformazione
"Guglielmo Marconi"

Dottorato di Ricerca in Automatica e Ricerca Operativa
Ciclo XVI

Settore concorsuale di afferenza: 01/A6 - RICERCA OPERATIVA
Settore scientifico disciplinare: MAT/09 - RICERCA OPERATIVA

Decomposition Methods and Network Design Problems

Tiziano Parriani

Coordinatore
Prof. Daniele Vigo

Relatore
Prof. Andrea Lodi

Esame Finale 2014

Contents

Keywords

- *Combinatorial Optimization*
- *Dantzig-Wolfe Decomposition*
- *Frank-Wolfe Algorithm*
- *Generalized Bundle Methods*
- *Heuristic*
- *Local Search*
- *Multicommodity Network Flows*
- *Progressive Hedging*
- *Robust Network Design*
- *Traffic Assignment*
- *Two-stage Programming*
- *Unit Commitment*

A Nazzareno

Ringraziamenti

Sono molte le persone che sento di dover ringraziare.

Purtroppo questi ringraziamenti non potranno essere letti dalla persona che più di tutti li merita, Alberto Caprara. È grazie a lui se conosco il piacere della ricerca e la soddisfazione della scoperta. È grazie ad Alberto se sono qui ora a scrivere questa tesi.

Ringrazio di cuore il mio advisor, Andrea Lodi, per avermi guidato con sapienza verso questo importante traguardo.

Ringrazio Antonio Frangioni per avermi saputo consigliare, motivare e migliorare costantemente. La sua disponibilità in questi anni è stata pari soltanto alla sua competenza e precisione.

Ringrazio Valentina Cacchiani. Fra i tanti motivi per cui Valentina merita questi ringraziamenti mi piace ricordare l'appoggio che ha saputo darmi nei primi mesi di dottorato.

Ringrazio tutte le persone da cui ho potuto apprendere i molti modi, tutti belli, di fare ricerca: Emiliano Traversi; Joerg Schweizer; Daniel Schmidt, Frauke Liers e Mike Jünger; Guojing Cong.

Ringrazio i miei compagni di viaggio e di "corridoio", Paolo Tubertini ed Eduardo Álvarez-Miranda. Senza di loro il dottorato non sarebbe stato la bellissima esperienza che è stata.

Ringrazio Chiara, che ha saputo consigliarmi, motivarmi e migliorarmi già molto tempo prima che io iniziassi il dottorato.

E ringrazio i miei genitori, per avermi sempre e comunque sostenuto in tutte le mie scelte, per avermi dato tutto l'appoggio di cui ho avuto di bisogno e spesso molto di più. In fondo, è grazie a loro se sono qui a scrivere queste pagine.

That's one giant leap for **a** *man,*
one small step for mankind.

Chapter 1

Introduction

1.1 Preface

This thesis comprises the research activity I have done during my Ph.D., under the supervision of Professor Alberto Caprara, Professor Andrea Lodi, and - even though not formally - Antonio Frangioni.

My first *attempts* of doing research started two-years before the Ph.D., when Alberto and Emiliano Traversi accepted to supervise my Master's theses. Column generation and decomposition methods, applied to flows on networks, have been the very first topics of my research. Once I started the Ph.D., further studies related to decomposition approaches have been motivated by the necessity of efficiently solving real world train timetabling problems, formulated as multicommodity flows. Decomposition approaches applied to multicommodity networks flows gained a primary role in my research activity.

I extended my studies to system traffic assignment problems in which a set of commodities must be routed through a network at minimum cost. Solutions of this problem are useful for network orientation and design, especially in relation with public transportation systems as the Personal Rapid Transit. Network design problems accompanied my studies on decomposition approaches for great part of my Ph.D., since my participation in the *Programma Vigoni*, by the German-Italian University Centre, gave me the opportunity to study robust network design problems.

I had the occasion to see a "real-world" application of my theoretical studies during my research period at IBM, the third summer of my Ph.D.. At IBM, two-stage stochastic unit commitment problems are solved to support the state of California plan of producing 33% of its electric energy from renewable resources, I am glad to have had the opportunity of partecipating to this commitment.

1.2 Content of the Thesis

The content of this thesis is divided in four chapters. An abstract is below presented for each chapter. Chapters correspond to research contributions (article, technical report, or conference presentation) whose status at the moment of writing is mentioned below. The notation varies among the chapters as each chapter is thought to be a self-contained manuscript whose comprehension does not strictly rely on any of the other chapters.

1.2.1 Chapter 2

The chapater refers to the *Technical Report OR-14-10*, "*Partial aggregation for Generalized Bundle Methods, An Application to the Multicommodity Min-Cost Flow Problem*", co-authored with Alberto Caprara and Antonio Frangioni[1]. An overview of the contribution of this chapter has been presented at the *12th INFORMS Telecommunications Conference* with the title "*A study of trade-offs in decomposition approaches to multicommodity network flows*". Other aspect of the same topic, and earlier studies, has also been presented in chronological order at the *42nd Annual Conference of the Italian Operational Research Society*, with the title "*A Comparison of the Natural Existing Approaches for Solving Multicommodity-Flow Problems*"; during the poster session at the *Mixed Integer Programming workshop*, at the *21st International Symposium on Mathematical Programming*, and at the *17th Combinatorial Optimization Workshop*, with the title "*An Analysis of Natural Approaches for Solving Multicommodity-Flow Problems*".

The chapter overviews the concepts of decomposition in linear programming. Decomposition based approaches are recalled from primal and dual point of view, emphasizing the equivalence between the Dantzig-Wolfe decomposition, the Kelley's cutting plane approach, and the Benders' decomposition. Especially in presence of unstable dual solutions, slow convergence issues may arise that may result in poor overall performances for this class of approaches. Several stabilization tools has been proposed in bibliography to reduce the oscillatory behaviour of the reduced master problem dual solutions. The notion of generalized bundle method is recalled as an unified theory that encloses the different stabilization strategies.

Convergence can be speeded-up consistently by also using disaggregated formulations for the reduced master problem. We investigate the possibility of building partially disaggregated reduced master problems. This extends the idea of aggregated-versus-disaggregated formulation to a gradual choice of alternative level of aggregation.

We applied the partial aggregation to the linear multicommodity minimum cost flow problem. The possibility of having only partially aggregated bundles opens a wide

[1] Department of Computer Science - University of Pisa

range of alternatives with different trade-offs between the number of iterations and the required computation for solving it. We explore this trade-off on several sets of instances and compare the computational results with the ones obtained by directly solving the node-arc formulation. Results suggest that an appropriate selection of the master problem problem formulation may make decomposition approaches more competitive.

1.2.2 Chapter 3

This chapter refers to the work "*Solving the optimum system traffic assignment problem for Personal Rapid Transit networks using the Frank-Wolfe algorithm*", co-authored with Joerg Schweizer[2], Emiliano Traversi[3], and Federico Rupi[2] (Technical Report DEI, OR-13-17). Some rudimentary studies related with this project can also be found in my Master's theses, "*Modelli e Metodi per l'Orientamento di Grafi*" (in Italian), supervised by Alberto Caprara, Emiliano Traversi and Joerg Schweizer.

The chapter describes a static traffic assignment method, including a minimum link cost model, for a class of emerging public transport systems called Personal Rapid Transit (PRT). PRT is a fully automated public transportation system where small-size vehicles run on exclusive guideways. Vehicles are available on-demand at stations and passengers can book individual trips, which are all non stop and without transfers.

As vehicles have only room for few passengers, higher capacities must be achieved by short, but safe headways. Due to its automated, on-demand service, a centralized PRT control system is not only able to route individual vehicles to their destination on the fastest route, but also to reroute empty vehicles to stations with waiting passengers.

The proposed PRT traffic assignment is based on continuous optimization and takes into account the following considerations: (1) the weighted sum of the travel times of all vehicles has to be minimized; (2) the information of the position of all vehicles as well as the origin and destination of all passengers are known by the central control; (3) the minimum safe distance vehicle control policy determines the flow-dependent link costs and ultimately limits link capacity; (4) the flow of empty vehicles must counterbalance the flow of vehicles occupied by passengers.

An iterative solution process to the route assignment problem is proposed, based on the well-known Frank Wolfe algorithm. The algorithm starts from a known initial feasible solution. A linear multicommodity min-cost flow problem is solved to optimality by using the decomposition techniques presented in the first chapter. The obtained solution is proved to be feasible for the traffic assignment problem and provided as input to the algorithm. The convergence of this algorithm is then demonstrated theoretically and

[2]Department of Electrical, Electronic and Information Engineering - University of Bologna
[3]Computer Science Laboratory of Paris - North University (LIPN)

numerically using two large scale network examples with uniform demand. Finally, the speed of convergence is studied in terms of iterations and computing time.

1.2.3 Chapter 4

This chapter is related with the article "*Single-commodity Robust Network Design Problem: Complexity, Instances and Heuristic Solutions*", co-authored with Eduardo Álvarez-Miranda[2], Valentina Cacchiani[2], Andrea Lodi[2], and Daniel Schmidt[4]. The paper is currently under the second round of reviews in *European Journal of Operational Research* and extends a previous work entitled "Models and Algorithms for Robust Network Design with Several Traffic Scenarios", by the same authors and Tim Dorneth, Michael Jünger[4] and Frauke Liers[5], published in *Combinatorial Optimization, LNCS series, Volume 7422, 2012, pp 261-272.*

In this work, we address a single-commodity Robust Network Design problem in which an undirected graph with edge costs is given together with a discrete set of balance matrices, representing different supply/demand scenarios. The goal is to determine the minimum cost installation of capacities on the edges such that the flow exchange is feasible for every scenario.

Previously conducted computational investigations on the problem motivated the study of the complexity of some special cases and we present complexity results on them, including hypercubes. In turn, these results lead to the definition of new instances (random graphs with {-1,0,1} balances) that are computationally hard for the natural flow formulation. These instances are then solved by means of a new heuristic algorithm for RND, which consists of three phases. In the first phase the graph representing the network is reduced by heuristically deleting a subset of the arcs, and a feasible solution is built. The second phase consists of a neighborhood search on the reduced graph based on a Mixed-Integer (Linear) Programming (MIP) flow model. Finally, the third phase applies a *proximity search* approach to further improve the solution, taking into account the original graph. The heuristic is tested on the new instances, and the comparison with the solutions obtained by CPLEX on a natural flow formulation shows the effectiveness of the proposed method.

Note This chapter appears in the Ph.D. thesis of **Eduardo Álvarez-Miranda** (Ph.D. Program in *Automatic Control and Operational Research*, Cycle XVI, Dipartimento di Ingegneria dell'Energia Elettrica e dell'Informazione, Università di Bologna, 2014).

[2]Department of Electrical, Electronic and Information Engineering - University of Bologna
[4]Computer Science Department - University of Koln
[5]Department of Mathematics - University of Erlangen-Nürnberg

1.2.4 Chapter 5

This chapter is related with the paper "*An Efficient Approach for Solving Large Stochastic Unit Commitment Problems Arising in a California ISO Planning Model*" co-authored with Guojing Cong[6], Carol Meyers[7], and Deepak Rajan[7]. At the moment of writing, the paper has been accepted and will be published in the *Proceedings of the 2014 IEEE PES General Meeting*. This work is the result of my research period at the *Thomas J. Watson Research Center - IBM*. A complementary result entitled *A framework for solving mixed-integer programming problems with decomposition and generic approaches*, with Guojing Cong and me as inventors, has been submitted to the *United States Patent and Trademark Office* (*Research Disclosure YOR8-2013-1124*).

The chapter describes our experience in obtaining significant computational improvements in the solution of large scale stochastic unit commitment problems. The model we use is a stochastic version of a planning model used by the California Independent System Operator, covering the entire WECC western regional grid. We solve daily hour-timestep stochastic unit commitment problem using a new progressive hedging approach that features linear subproblems and heuristic approaches for finding feasible solutions.

With respect to a brute-force approach based on the use of state-of-the-art commercial software, For stochastic problems with 5 scenarios, the algorithm produces near-optimal solutions with a 6 times improvement in serial solution time, and over 20 times when run in parallel; for previously unsolvable stochastic problems, we obtain near-optimal solutions within a couple of hours. We observed that, although being demonstrated for stochastic unit commitment problems, the algorithm is suitable for being applied to generic stochastic optimization problems.

[6] Advanced Computing Technology Center - IBM
[7] Lawrence Livermore National Laboratory

Chapter 2

Partial Aggregation for Generalized Bundle Methods, An Application to the Multicommodity Min-Cost Flow Problem[1]

2.1 Introduction

As a natural extension of the single-commodity case, multicommodity network flow problems has been of interest since the early 1960's (see, e.g., [17, 18, 35]). The wide area of applications for this class of problems goes from telecommunication to scheduling problems, from train timetabling to transportation problems (see e.g. [37, 19, 1]). Besides the number of applications, interest around multicommodity flows problems is also due to the typical block-angular structure of the constraint matrix that motivated the very first studies of decomposition approaches. As the dimension of the problem rises, and even considering the abilities of nowadays state-of-the-art of commercial software, the direct resolution of the corresponding linear program may results in very poor performance in terms of computing time. In order to reduce the computation required to solve real-world problems, the idea of exploiting the structure of the constraint matrix of a linear program found a first application to multicommodity flow problems.

[1] **Technical Report** "*Partial aggregation for Generalized Bundle Methods, An Application to the Multicommodity Min-Cost Flow Problem*", A. Caprara, A. Frangioni, T. Parriani *Tech. Rep. DEI, OR-14-10.*

In Paragraph 2.2 an overview of decomposition methods in linear programming is given. Decomposition-based approaches are recalled from primal and dual point of view, emphasizing the equivalence between the Dantzig-Wolfe decomposition, the Kelley's cutting plane approach, and the Benders' decomposition. Despite being applied with success in many contests, slow convergence issues may arise in this class of approaches resulting in poor overall performances. Several stabilization tools has been proposed in bibliography to reduce the oscillatory behaviour of the reduced master problem dual solutions. The notion of generalized bundle method is recalled, as an unified theory that encloses the different stabilization strategies.

Convergence can be speeded-up consistently also using a disaggregated formulation of the reduced master problem. In Paragraph 2.3 the main contribution of this chapter is presented, as we investigate the possibility of building partially aggregated bundles starting from disaggregated pricer solutions. An application to the linear multicommodity minimum cost flow problem is presented, and the possibility of choosing among alternative partially aggregated formulations for the reduced master problem is studied.

In Paragraph 2.4 computational results are presented. Tests has been performend on a pool of instances with different topologies and with different definition of commodities. Results of several alternative stabilized and partially aggregated formulations are compared with the direct resolution of the natural linear formulation for the problem.

Finally, on the basis of the observed resuts, in Paragraph 2.5 we draw the conclusions and mentions interesting prospective for future work.

2.2 Decomposition Approaches in Linear Programming

2.2.1 A Primal Point of View

In [17] Lester R. Ford, Jr. and Delbert R. Fulkerson introduced a formulation for the maximal multi-commodity network flow problem that makes use of a "very large" number of variables. They also suggested the use of an iterative process, today known as as *"column generation"*, for solving the proposed formulation. George B. Dantzig and Philip S. Wolfe generalized the idea to linear programming [14, 15] and today this very well known approach is commonly referred to as *Dantzig-Wolfe decomposition* (DW) approach. We here recall its basic principles.

Any linear program

$$(LP) \quad \max \quad c^T u \tag{2.1}$$

$$s.t.$$

$$Du \leq d \tag{2.2}$$

$$Au \leq b \tag{2.3}$$

$$u \in \mathbb{R}^n_+ \quad , \tag{2.4}$$

can be written also as:

$$\max \quad c^T u \tag{2.5}$$

$$s.t.$$

$$Au \leq b \tag{2.6}$$

$$u \in \mathcal{U} \tag{2.7}$$

where $\mathcal{U}$ is the feasibility region defined by the so called *"easy"* constraints (2.2). On the opposite, constraints (5.3) are commonly referred to as *"complicating"* constraints. This is due to the fact that the approach is effective when LP become a much easier optimization problem once (5.3) are relaxed. A common situation (that also applies to the multicommodity flow problem, as discussed later) is the one in which, for (5.3) relaxed, LP fully decomposes into a number of independent and easier linear problems.

Assuming for the sake of simplicity that $\mathcal{U}$ is non-empty and bounded, it is possible to write an equivalent formulation for LP in terms of the the set of extreme points of the polyhedron $\mathcal{U} := \{u \in \mathbb{R}^n_+ | Du \leq d\}$, which we will denote as P_u. In fact, any vector $u \in \mathcal{U}$ can be written as a convex combination of the extreme points of $\mathcal{U}$:

$$u = \sum_{j \in P_u} \theta_j u^j \quad , \quad \sum_{j \in P_u} \theta_j = 1 \quad , \quad 0 \leq \theta_j \leq 1 \quad j \in P_u \quad . \tag{2.8}$$

This immediately leads to the *Dantzig-Wolfe reformulation* of LP

$$(\text{MP}) \quad \max \quad c^T \left(\sum_{j \in P_u} \theta_i u^j \right) \tag{2.9}$$

$$s.t.$$

$$\sum_{j \in P_u} \theta_j = 1 \tag{2.10}$$

$$A \left(\sum_{j \in P_u} \theta_j u^j \right) \leq b \tag{2.11}$$

$$\theta_j \geq 0 \qquad \forall j \in P_u \quad . \tag{2.12}$$

Extending the theory to more general assumptions is relatively straightforward: if $\mathcal{U}$ is

empty then LP is infeasible, while if $\mathcal{U}$ is unbounded it is sufficient to define a vector $u \in \mathcal{U}$ as a convex combination of not only the extreme points but also the extreme rays of $\mathcal{U}$ [14].

The number of variables in MP easily becomes intractable for non-trivial problems and "the proposed method would be of little interest or value if it were not possible effectively to reduce this number" (cf. [15]). The idea at this point is to solve a formulation of the problem in which only a subset of all the possible variables are present: the "*reduced master problem*" (RMP). Once the RMP has been solved, dual information are used to decide wheter the solution is optimal also for MP or if one or more new variables should be "generated" and the RMP should be solved again in an iterative process. Such procedure takes the name of *"column generation"*.

2.2.1.1 Column Generation

The column generation is a general framework that can be applied to any linear program, but it is effective in practice when the number of variables is, generally speaking, very large. MP formulations, originated by the application of the DW approach to several different classes of problems, have been by far the the most successful application for the column generation (**rif.**).
Given a subset of extreme points $\overline{P}_u \in P_u$, we define the RMP:

$$\text{(RMP)} \quad \max c^T \left(\sum_{j \in \overline{P}_u} \theta_j u^j \right) \tag{2.13}$$

$s.t.$

$$(y) \qquad \sum_{j \in \overline{P}_u} \theta_j = 1 \tag{2.14}$$

$$(x) \qquad A \left(\sum_{j \in \overline{P}_u} \theta_j u^j \right) \leq b \tag{2.15}$$

$$\theta_j \geq 0 \qquad \forall j \in \overline{P}_u \quad . \tag{2.16}$$

From the point of view of the simplex algorithm [13], in RMP we are forcing all the variables θ_i for which $u^j \in \overline{P}_u{}' \doteq P_u \setminus \overline{P}_u$ to be out of the optimal basis. The reduced cost of a generic θ_i is given by $r_j \doteq (c^T - xA)u^j - y$, where y is the dual variable associated with constraint (2.14) and x is the vector of the dual variables associated with constraints (2.15). Letting (y_i, x_i) be an optimal dual solution for the RMP in a generic iteration i, if

$$r_i^j = (c^T - x_i A_j)u^j - y_i \leq 0 \; \forall j : u^j \in \overline{P}'_u \tag{2.17}$$

then the corresponding primal solution $u_i = \sum_{j \in P_u} \theta_i u^j$ is optimal for the original problem. Otherwise, any column j for which $r_i^j > 0$ is a good candidate for entering in

the basis and therefore can be generated (i.e., added to the RMP). The RMP is then solved again in an iterative process that terminates when condition (2.17) is satisfied.

At least theoretically, it is possible to store all the extreme points in $\mathcal{U}$ and evaluate at each iteration their reduced costs. Anyway, such procedure is not of practical interests in most of the cases as the size of $|P_u|$ is, generally speaking, intractable.
A *"pricer"* is then defined as a linear program that, given as input a (partial) dual solution for RMP, x_i, returns a vertex u^j with the most negative reduced cost. Recalling (2.8) the (aggregated) subproblem can be formulated as:

$$f(x) = \max \quad (c - xA)u \tag{2.18}$$

$$s.t.$$

$$Du \leq d \tag{2.19}$$

$$u \in \mathbb{R}^n_+ \tag{2.20}$$

If $f(x_i) \leq y_i$ then condition (2.17) is satisfied and there is no column that can enter the basis with a positive reduced cost, so θ_i is optimal for MP. Otherwise, the vertex u_i^j optimal for $f(x_i)$ is than the most promising candidate and is generated in RMP.

In theory, column generation converge to the optimal solution in a finite number of iterations. In [15] is shown how this property can be derived directly form the convergence theory of the simplex algorithm. In practice, the required number of iteration is generally speaking much lower than the total number of vertexes of $\mathcal{U}$. While the computation required to solve the reduced master problem depends on the number of vertexes inserted, the column generation procedure is effective when the pricer is easy in the sense that is computationally convenient to solve many time the pricer instead of solving directly the original problem.

2.2.2 A dual point of view

If from a "primal point of view" the DW can be seen in extreme synthesis as a huge-scale reformulation of the problem solved by column generation, from a "Lagrangian dual point of view" the DW is known to be equivalent to Kelley's cutting plane method. In fact, since $A(\sum_{j \in P_u} \theta_j u^j) \leq b \Leftrightarrow \sum_{j \in P_u} (Au^j - b)\theta_j \leq 0$, the dual of MP can be written as:

$$\min \quad y \tag{2.21}$$

$$s.t.$$

$$y \geq c^T u^j + x^T(b - Au^j) \qquad j \in P_u \tag{2.22}$$

$$x \geq 0 \tag{2.23}$$

which is equivalent to:

$$(\Pi) \quad \min_{x\geq 0} \{ f(x) = \max_{j\in P_u} \{ c^T u^j + x^T(b - Au^j) \} \} \tag{2.24}$$

Note at this point that Π is the Lagrangian dual of the original problem LP with x as the vector of Lagrangian multipliers associated to the constraints (2.2). Kelley's cutting plane method is now applied to solve Π by iteratively solving an approximation of the original function that considers only a limited subset $\mathcal{B}$ of the elements in P_u:

$$(\Pi_{\mathcal{B}}) \quad \min_{x\geq 0} \max_{j\in\mathcal{B}} c^T u^j + x^T(b - Au^j) \quad . \tag{2.25}$$

The subset $\mathcal{B}$, as for the DW, is iteratively built using an "oracle" that takes as input a "*tentative point*" x_i, optimal for $\Pi_{\mathcal{B}}$ in a generic iteration i, and that returns the solution $(f(x_i), u_i)$ optimal for

$$f(x) = \quad \max_{u\in\mathcal{U}} (c^T - x^T A)u \ \ (+ x^T b) \tag{2.26}$$

which clearly corresponds to DW pricer. Note that the component $x^T b$ does not depend on u and generally is not considered in the optimization process.
During the iterative process, a finite set of tentative points and the corresponding optimal solutions $(f(x_i), u_i)$ are therefore collected to form $\Pi_{\mathcal{B}}$. In other words, instead of the "true" function $f(x)$ we minimize its "*cutting plane model*" $f_{\mathcal{B}}(x)$, i.e., it polyhedral lower approximation built up with the information in $\mathcal{B}$. Introducing $z_i \doteq b - Au_i \in \partial f(x_i)$, one can write $\mathcal{B} = \{ (x_i, f(x_i), z_i) \}$ and, by simple substitution of u_i with the corresponding z_i, the (aggregated) *cutting-plane model* can now be written more conveniently as:

$$f_{\mathcal{B}}(x) \ = \ \max \ \{ \ f(x_i) + z_i(x - x_i) \ : \ (x_i, f(x_i), z_i) \in \mathcal{B} \ \} \ \leq \ f(x) \tag{2.27}$$

while a tentative point is the optimal solution of

$$\min_x \{ \ f_{\mathcal{B}}(x) \ \} \ = \ \min_{v,x} \{ \ v \ : \ v \geq f(x_i) + z_i(x - x_i) \quad (x_i, f(x_i), z_i) \in \mathcal{B} \ \} \ .$$

Recognizing the equivalence between the DW and Kelley's cutting plane method is useful for the understanding of an unified convergence theory that covers both the classes of approaches under the same concepts. For similar reasons, despite it may seem at this point an useless change of notation, it helps writing (2.27) instead of (2.26) for identifying the cutting plane model.
The equivalence it is well known to be extendible also to Benders' decomposition, when

this is applied to the dual of LP:

$$\min \gamma^T d + x^T b \tag{2.28}$$
$$D^T \gamma + A^T x = c \tag{2.29}$$
$$\gamma, x \leq 0 \quad . \tag{2.30}$$

Following the nomenclature proper of Benders' decomposition, variables γ take here the name of *"easy"* variables while variables x are the *"complicating"* ones. We now recall that Benders' decomposition can be applied even if a more complex function $g(x)$ is present is place of $x^T b$. The feasibility region X itself can be much more complex than $\mathbb{R}$, therefore there are several situation that, in general combinatorial optimization, motivates this dichotomy of variables, as mentioned in [29]. Anyway, limiting the discussion to continuous and linear optimization the correspondence with DW *easy* and *complicating* constraints is clear. In fact, the Benders' reduced master problem for matches exactly with $\Pi_{\mathcal{B}}$ while the Benders' subproblem:

$$\min \quad \gamma^T d \tag{2.31}$$
$$s.t.$$
$$D^T \gamma = c - xA \tag{2.32}$$
$$\gamma \leq 0 \tag{2.33}$$

is again dually equivalent to $f(x)$.

2.2.3 The stopping criteria

2.2.4 Block-angular structured problem

As already mentioned, there might be several reasons that motivate the identification of a subset of constraints as *complicating*. One of the most common situations in continuous optimization, the same that originally motivated the development of decomposition-based approaches, is the one in which the *complicating* constraints couple a finite subset of otherwise independent and easy problems (see Fig. 2.1). In this case *complicating* constraints usually take the name of *"joint"* or *"coupling"* while the coefficient matrix can be arranged to a so called *"block-angular"* structure [50].

In many application, an obvious arrangement of the coefficients in *"blocks"* derives naturally from the definition of the problem itself. Nevertheless, as will be discussed more in detail in the following paragraphs, it may be computationally convenient in some cases an aggregation, or a disaggregation, of the naturally defined *blocks*. Even when *blocks* definition is not obvious, it is still possible to derive the structure through the direct application of algebraic analysis on the coefficient matrix [27, 5, 8, 28].

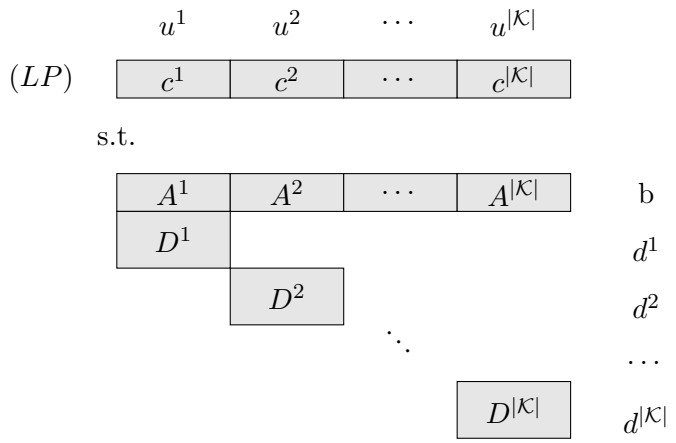

FIGURE 2.1: *Block-angular* structured LP.

The fully decomposable problem in which *coupling* constraints are relaxed is by definition the DW pricer. Letting $k \in \mathcal{K}$ be the set of *"blocks"* in LP, $f(x)$ is therefore decomposed in k fully independent components $f^k(x)$, with:

$$f^k(x) = \max \left\{ (c^k - xA^k)u^k \ : \ u^k \in \mathcal{U}^k \right\} \quad \forall\, k \in \mathcal{K} \quad . \tag{2.34}$$

2.2.5 Stabilization and Generalized Bundle Methods

Although decomposition algorithms have been used with success in many applications, slow convergence is known to be a key issue that often threatens the overall performances of the approach, as it translates into a large number of iterations and to the necessity of adding many column (cuts) to reach optimal solutions. A first motivation behind this behavior might be detected in a "bad" definition of the initial set of variables/cuts. It comes natural to think in fact, especially from a cutting plane point of view, that a bad initial cut-set will produce far-from-optimum solutions and, consequently, poor quality tentative points provided to the pricer. In practice, however, it is hard to construct a "good enough" initial bundle: indeed, especially in presence of primal degeneracy, even if the pricer were invoked on the very optimal solution there would in fact be no guarantee that it would return the whole set of cuts that are necessary to prove its optimality. It is more likely in fact that after a new cut is added the solution of $\Pi_{\mathcal{B}}$ moves away from the previous one, and that the previous solution will be definitely loss during the evaluation of the next tentative point. Moreover, the next solution may be far from the optimal one, and the quality of cuts generated by the pricer on that point will be lower. This effect can be countered by employing *stabilization* approaches in which a good solution for Π, among those obtained so far, is chosen as *"stability center"*, and the distance from this center is bounded (as in [44]) or penalized ([42]); the stability center is changed if a "better" dual point is found. Many different types of stabilization techniques have been analysed in the literature (see, e.g., [2, 20, 30, 33, 38, 40, 46]. Among those, we here recall the well known bundle

method and its generalization, namely, generalized bundle methods which combines several of the stabilization approaches under an unified convergence analysis.

Bundle methods go back to the 70's and are known to be among the most efficient implementable algorithms for convex non-differentiable optimization. We present them in our context, i.e., where the function to be minimized is a Lagrangian function of a (convex) problem.

Naming $\overline{x}$ the stability center, bundle methods penalize the displacement from $\overline{x}$ with a so called *"stabilizing"* term proportional to $||x - \overline{x}||^2$. In summary therefore, the bundle method asks to find a new tentative point $x_{test} \geq 0$ by minimizing, for some prescribed parameter $t > 0$ the function:

$$f_{\mathcal{B}}(x) + \frac{1}{2t}||x - \overline{x}||^2$$

When the stabilizing term is considered in the objective function we have a *stabilized reduced master problem*. *Generalized* bundle methods, introduced in [20], considers the use of a more generic closed convex function $D_t(d)$, The stabilized (dual) reduced master problem becomes:

$$(\Pi_{\mathcal{B},\overline{x},t}) = \phi_t(\overline{x}) = \inf_d \{f_{\mathcal{B}}(\overline{x} + d) + D_t(d)\} \quad . \tag{2.35}$$

The stabilizing term D_t may be simply kept fixed during the iterative process (assuming that it has suitable properties [20]), in practice on-line tuning of t to reflect the actual "quality" of the approximation $f_{\mathcal{B}}$ of f is known to be very important. General rules can be given to ensure that the corresponding algorithm is convergent.

The apparent difficulty behind the generalization of the stabilizing term to any closed convex function rely on the necessity of obtaining its dual formulation to compose the pricer. The key idea consists here in using the concepts of *convex conjugate* and of *Fenchel duality* (see, e.g., [34]) instead of the "canonical" one. Recalling the definition of convex conjugate:

$$f^*(z) = \sup(z^T x - f(x)), z \in \mathbb{R}^n \tag{2.36}$$

Given that $x_i = f^*(z_i)$, we have that z_i is a subderivative of f in x_i and, if f is differentiable in x_i, that $\partial f(x_i) = z_i$.

Naming $v(\Pi)$ the optimal value for $\Pi_{\overline{x},t}$, we thus have that $f^*(0) = v(\Pi)$ and that the (apparently weird) dual problem

$$(\Delta) \quad \inf_z \{f^*(z) : z = 0\} \tag{2.37}$$

is equivalent to $\Pi_{\overline{x},t}$, i.e., $v(\Pi) = -v(\Delta)$. One can thus construct the Fenchel's dual of any convex program by computing the conjugate of the objective function and

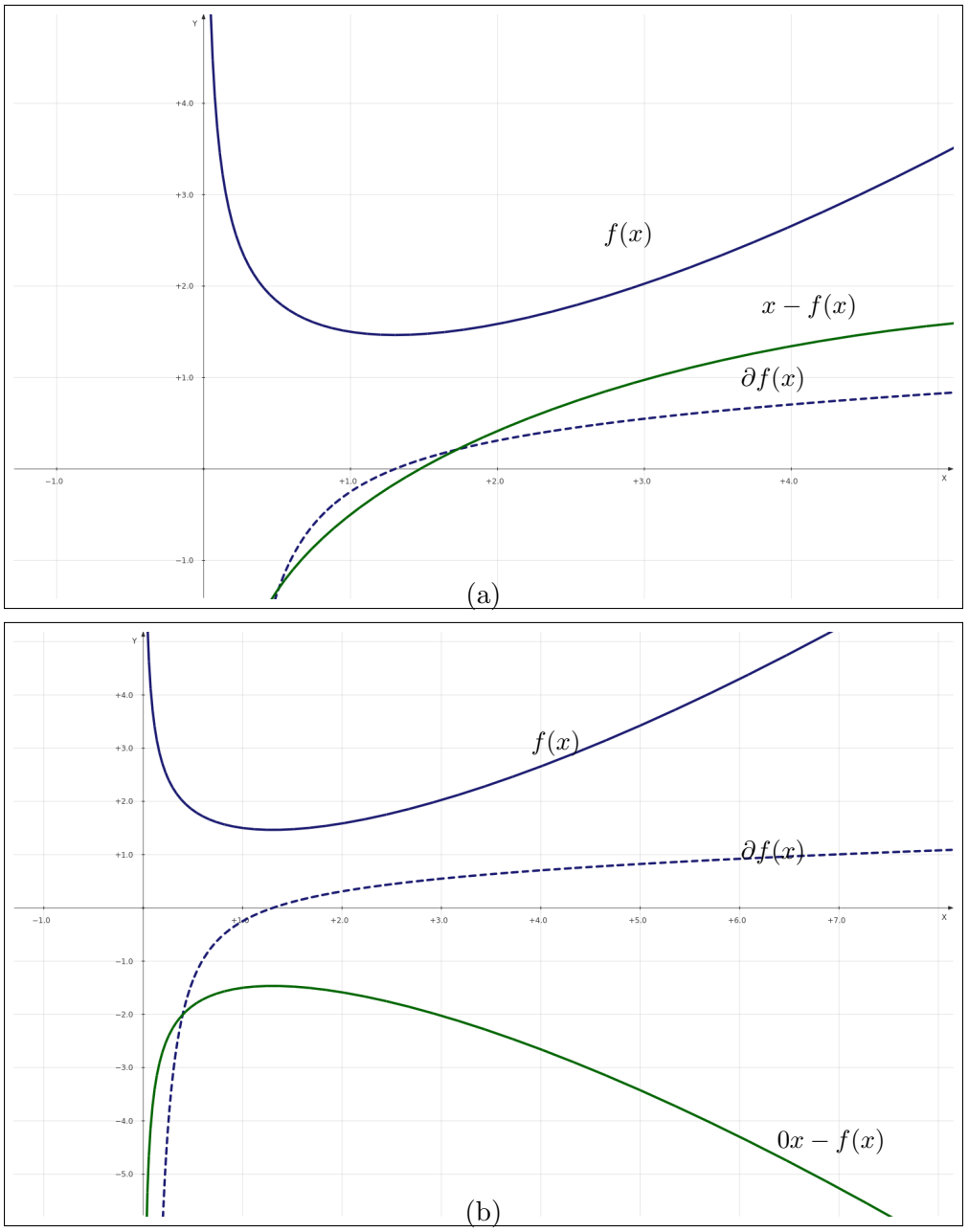

FIGURE 2.2: Examples of convex conugate for a differentiable function: The conjugate for $z = 1$ and the conjugate for $z = 0$ are shown in Fig. 2.2.a and2.2.b, respectively. The maximum for $f^*(z)$ is obtained in the point for which $\partial f(x) = z$, when $z = 0$ such point corresponds to the minimum for f.

evaluating it in 0; doing this for (2.35) reveals the stabilized reduced master problem:

$$(\Delta_{\mathcal{B},\overline{x},t}) \quad \inf_z\{f^*_{\mathcal{B}}(z) - z\overline{x} + D^*_t(-z)\} \quad . \tag{2.38}$$

With $f_{\mathcal{B}} = f$, (2.38) is the dual of Π and for $D_t = 0$ it matches exactly with the RMP. It can be in fact seen as a (generalized) augmented Lagrangian of (2.37) where the constraints $z = 0$ are replaced with the linear term $-\overline{x}z$ (with Lagrangian multipliers $\overline{x}$) and the nonlinear term $D^*_t(-z)$ in the objective function. The Lagrangian relaxation of (2.37) with respect to the constraints $z = 0$, using x as Lagrangian multipliers,

$$(\Delta_x) \quad f(x) = -\inf_z\{f^*(z) - zx\} \tag{2.39}$$

is, finally, a pricer for $\Delta_{\mathcal{B},\overline{x},t}$: in fact,

$$-\inf\{f^*(z) - zx\} = \sup zx - f^*(z) = (f^*)^*(x) = f(x) \quad .$$

Any convex closed convex function can now be "translated" in his dual to compose the primal formulation of the reduced master problem.

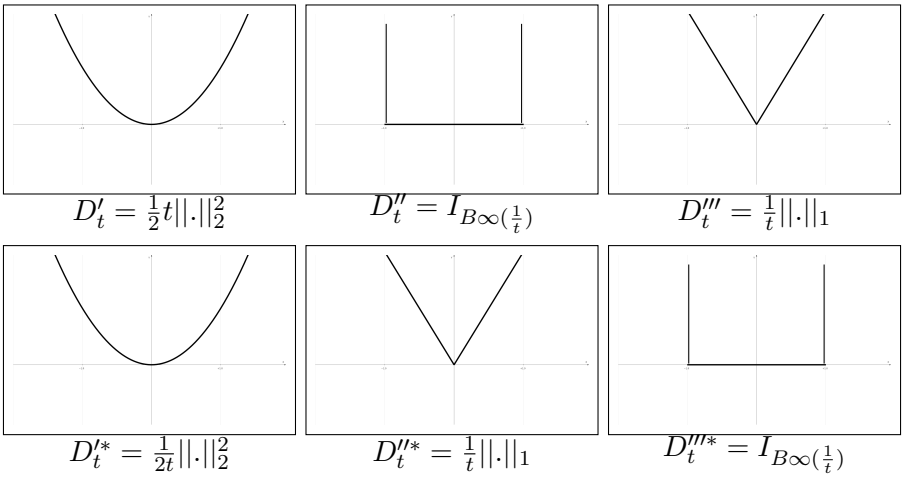

FIGURE 2.3: Well known differentiable and non-differentiable stabilization terms and their convex conjugates.

2.3 Partially Aggregated Bundles, An Application to the Multicommodity Flow Problem

2.3.1 Partially Aggregated Bundles

For what said so far, a single cut returned by the pricer is added to the model of $f(x)$ after each iteration of the generalized bundle method. However, the very first application of the DW approach came with an "arch-chain" formulation for multicommodity

flow problems that assumes the generation of a variable for each commodity at each iteration ([17, 48]). In this (that is the typical) case, being the pricer decomposed among a set of $\mathcal{K}$ subproblems (see fig. 2.1) it comes natural to compose the bundle directly using the "*disaggregated*" functions $f^k(x)$ (see (2.34)). In other words, a separate oracle is available for each component $k \in \mathcal{K}$ providing $f^k(x_i)$ and $z_j^k \in \partial f^k(x_i)$, so that $f(x_i) = \sum_{k\in\mathcal{K}} f^k(x_i)$ $(+x_i b)$ and $z_i = \sum_{k\in\mathcal{K}} z_i^k$ $(+b)$. Thus, one can keep $|\mathcal{K}|$ separate *disaggregated bundles* $\mathcal{B}^k$ and use them to construct $|\mathcal{K}|$ *independent models* $f_{\mathcal{B}}^k$, one for each component. For the same set of tentative points x_i probed, the *disaggregated cutting plane model*

$$f_{\mathcal{B}}(x) = \sum_{k\in\mathcal{K}} \Big(\; f_{\mathcal{B}}^k(x) = \max \{ \; f^k(x_i) + z_i^k(x - x_i) \; : \; (x_i, f^k(x_i), z_i^k) \in \mathcal{B}^k \; \} \; \Big) \quad (2.40)$$

gives a much better approximation of the original function than the aggregated one. This of course comes at the cost that the *disaggregated primal and dual reduced master problem*

$$(\Delta_{\mathcal{B},\overline{x},t}) \quad \inf_z \{\textstyle\sum_{k\in\mathcal{K}} f_{\mathcal{B}}^{k*}(z) - z\overline{x} + D_t^*(-z)\} \quad (2.41)$$

and

$$(\Pi_{\mathcal{B},\overline{x},t}) = \inf_d \{\sum_{k\in\mathcal{K}} f_{\mathcal{B}}^k(\overline{x} + d) + D_t(d)\} \quad , \quad (2.42)$$

have a significantly larger number of variables and/or constraints than their aggregated counterparts, and therefore can be much more costly to solve. However, the use of the disaggregated model improves the convergence speed so much as to amply compensate the increased cost due to the larger size of the corresponding master problems [4, 6, 16, 36, 47, 24]. Given the availability of several possible strategies to stabilize the reduced master problem, the focus of this work is thus on the the choice of the model $f_{\mathcal{B}}$.

Let us consider any possible partition $\mathcal{C} = \{\zeta_\varsigma : \varsigma = 1, \ldots, |\mathcal{C}|\}$ of the (index set of) blocks $\mathcal{K}$; this automatically defines "*partially aggregated*" pricer solutions $u_i^\varsigma = [u_i^k]_{k\in\zeta_\varsigma}$ for each $\varsigma \in \mathcal{C}$, and the corresponding partition of the vector $c = [c^\varsigma]_{\varsigma\in\mathcal{C}}$ and of the matrix $A = [A^\varsigma]_{\varsigma\in\mathcal{C}}$. Consequently, one has

$$z_i^\varsigma \doteq -Au_i^\varsigma \; , f^c(x) \doteq ((c^\varsigma)T - x^T A^\varsigma)u_i^\varsigma \quad \forall \varsigma \in \mathcal{C} \quad (2.43)$$

resulting in:

$$f(x) = \sum_{\varsigma\in\mathcal{C}} f^\varsigma(x) \; (+xb) \qquad \text{and} \qquad z_i = \sum_{\varsigma\in\mathcal{C}} z_i^\varsigma \; (+b) \; ; \quad (2.44)$$

note that f is thus decomposed in $|\mathcal{C}| + 1$ components, separately counting the linear term xb with its constant (sub)gradient b. We can now partially aggregate the solutions of the subproblems and collect them in a set of *partially aggregated bundles* $\mathcal{B}^\varsigma$.

Proposition 2.1. *The "partially aggregated model":*

$$f_{\mathcal{B}}(x) = xb + \sum_{\varsigma \in \mathcal{C}} \left(\; f^{\varsigma}_{\mathcal{B}}(x) = \max \left\{ \; f^{\varsigma}(x_i) + z_i^{\varsigma}(x - x_i) \; : \; (x_i, f^{\varsigma}(x_i), z_i^{\varsigma}) \in \mathcal{B}^{\varsigma} \; \right\} \; \right). \tag{2.45}$$

is a valid model for $f(x)$.

Proof. Any partially aggregated pricer solution u_i^{ς} corresponds to a vertex $u^j \in P_u^{\varsigma}$ of the feasibility region $\mathcal{U}^{\varsigma}$ defined by all the easy constraints of the blocks $k \in \zeta_{\varsigma}$. Following the steps of paragraph 2.2.2 backwards we can write the "*partially aggregated reduced master problem*" as:

$$(\text{RMP}') \quad \max \quad c^T \left(\sum_{\varsigma \in \mathcal{C}} \sum_{j \in \overline{P}_u^{\varsigma}} \theta_j u^j \right) \tag{2.46}$$

$$s.t.$$

$$\sum_{j \in \overline{P}_u^{\varsigma}} \theta_j = 1 \quad \forall \varsigma \in \mathcal{C} \tag{2.47}$$

$$A \left(\sum_{\varsigma \in \mathcal{C}} \sum_{j \in \overline{P}_u^{\varsigma}} \theta_j u^j \right) \leq b \tag{2.48}$$

$$\theta_j \geq 0 \qquad \forall \varsigma \in \mathcal{C} \; \forall \; j \in \overline{P}_u^{\varsigma} \tag{2.49}$$

which is equal to the disaggregated reduced master problem with additional constraints that forces all the θ_j associated to the same partially aggregated variable to have the same value. Therefore, the feasibility region defined by RMP$'$ is strictly contained in the one defined by the disaggregated reduced master problem and, from a dual point of view, it holds $f_{\mathcal{B}}(x) \leq f(x)$ for $f_{\mathcal{B}}(x)$ defined as in 2.45 . □

The result can be easily seen by a different angle; the aggregated master problem basically is nothing else than the disaggregated one where one forces all the multipliers corresponding to the same iteration to be equal. While this clearly restricts the feasible region, it has the benefit to reduce the number of free variables and therefore the actual size of the master problem. However, nothing prevents us from only partly playing the same game: only multipliers belonging to the same iteration *and component* are restricted to be equal, but they can differ from the ones of the same iteration and a different component. The choice of the model is therefore not simply between "aggregated" and "disaggregated"; a larger spectrum of possibilities exist that one may want to explore in search of the best possible trade-off between the convergence speed and the cost of the reduced master problem solution. As our results will show, exploiting this trade-off can allow to reduce computation times significantly in our application.

2.3.2 The Multicommodity Min-Cost Flow Problem

The Multicomnmodity Min-cost Flow (MMCF) problem may be formulated as follows. Given a directed graph $G = (N, A)$, where N is the set of nodes and A is the set of arcs, and given a set of commodities K, we have for each commodity $k \in K$ a balance vector $b^k = [b_i^k]_{i \in N}$ indicating the net amount of flow required by every node: nodes with positive balance are *"sources"*, nodes with negative balance are *"destinations"*, and nodes with zero balance are *"transshipment"*. All the balances must be satisfied. For each arc $(i, j) \in A$ and for each commodity is defined an "*individual capacity*" u_{ij}^k that restricts the amount flow of k routed on (i, j). For each arc is also defined a cost c_{ij}^k and a "*mutual capacity*", u_{ij}, that restricts the total amount of flow routed on the arc. The problem is to minimize the overall routing cost satisfying the capacity constraints. A compact formulation for the problem can be easily derived from the single-commodity min cost flow problem, and writes

$$\begin{aligned}
\min \quad & \textstyle\sum_{k \in K} \sum_{(i,j) \in A} c_{ij}^k w_{ij}^k && && (2.50)\\
s.t. \quad & && &\\
& \textstyle\sum_{(j,i) \in A} w_{ji}^k - \sum_{(i,j) \in A} w_{ij}^k = b_i^k && i \in N\ ,\ k \in K && (2.51)\\
& \textstyle\sum_{k \in K} w_{ij}^k \le u_{ij} && (i,j) \in A && (2.52)\\
& 0 \le w_{ij}^k \le u_{ij}^k && (i,j) \in A\ ,\ k \in K && (2.53)
\end{aligned}$$

where the *mutual capacity constraints* (2.52) couples a set of $|K|$ otherwise fully independent problems given by the flow conservation and the individual capacity constraints. Recognizing in (2.52) the complicating constraint for this class of problem, decomposition based techniques have always been seen as a natural approach for MMCF problems (see e.g. [7, 12, 21, 22, 26, 36, 41]). Naming $x = [\ x_{ij}\ ]_{(i,j) \in A} \ge 0$ the vector of Lagrangian multipliers, the pricer (see 2.26):

$$\begin{aligned}
f(x) = \min \quad & \textstyle\sum_{k \in K} \sum_{(i,j) \in A} (c_{ij}^k + x_{ij}) w_{ij}^k && && (2.54)\\
s.t. \quad & && &\\
& \textstyle\sum_{(j,i) \in A} w_{ji}^k - \sum_{(i,j) \in A} w_{ij}^k = b_i^k && i \in N\ ,\ k \in K && (2.55)\\
& 0 \le w_{ij}^k \le u_{ij}^k && (i,j) \in A\ ,\ k \in K && (2.56)
\end{aligned}$$

is in fact fully decomposable in k single-commodity *min-cost flow* problems, for which plenty of efficient solution algorithms exist [23, 26]. The aggregated master problem

for the MMCF is, in turn,

$$\text{(MCCF-MP}_1\text{)} \qquad \min \quad \sum_{s\in S}\left(\sum_{k\in K} c^k w_s^k\right)\theta_s \tag{2.57}$$

$$s.t.$$

$$\sum_{s\in S}\left(\sum_{k\in K} w_{ij,s}^k\right)\theta_s \le u_{ij} \quad (i,j)\in A \tag{2.58}$$

$$\sum_{s\in S}\theta_s = 1 \tag{2.59}$$

$$\theta_s \ge 0 \qquad s\in S \tag{2.60}$$

where S is the set of the extreme points of the feasibility region defined by constraints (2.55) and (2.56) and w_s^k is the vector of the flows for commodity k in solution s. Such formulation, anyway, differs from the popular (and very first) *"arc-path"* reformulation ([48]) that rather has the structure of (2.40). In fact, identifying $\mathcal{K}$ (the set of independent blocks in a generic decomposable pricer) in K (the set of commodities), it comes more natural to formulate the reduced master problem in a *disaggregated form*

$$\text{(MCCF-MP}_0\text{)} \qquad \min \quad \sum_{k\in K}\sum_{s\in S^k}\left(c^k w_s^k\right)\theta_s \tag{2.61}$$

$$s.t.$$

$$\sum_{k\in K}\sum_{s\in S^k} w_{ij,s}^k\theta_s \le u_{ij} \quad (i,j)\in A \tag{2.62}$$

$$\sum_{s\in S^k}\theta_s = 1 \qquad \forall k\in K \tag{2.63}$$

$$\theta_s \ge 0 \qquad s\in S \tag{2.64}$$

where S^k is the set of extreme points for the flow of commodity k. Now, it is well-known (e.g. [36]) that for any MMCF problem one can construct an equivalent formulation (known as "*ODP formulation*") in terms only of single-Origin single-Destination commodities (*OD*). In fact for any given graph G and for any given set K of general multi-origin multi-destination commodities, it is always possible to add one or more super-sinks to G in order to create an equivalent "extended" graph in which all the commodities in K are multi-source single-destination (or "*destination specific*"). Then, any destination-specific commodity can then easily be split in a set of OD in such a way that the resulting ODP is equivalent to the original one in the sense that provide the same optimal solution. In this case, a commodity k can be uniquely characterized by its origin s_k, its destination t_k and the amount b_k of product p_k that has to be shipped between them. One could also assume without loss of generality that commodities can be uniquely defined by their OD pair, with no mention to the specific product (just add extra origin and/or destination nodes that only serve to differentiate for the product); if furthermore individual capacity constrains are *not* tight, i.e., $u_{ij}^k \ge u_{ij}$ in (2.56), the extreme points S^k correspond to s_k-t_k paths. Thus, denoting by P the set of all required ODs, a solution of the pricing problem now consists in a set of paths p_ς^π, $\pi\in P$; denoting by S^π the set of all paths for the OD-pair π, and by S their union,

one has the corresponding reformulation of MMCF

$$\begin{aligned}
(\text{MCCF-MP}_{-1}) \quad \min \quad & \textstyle\sum_{\pi\in P}\sum_{\varsigma\in S^\pi} c^\varsigma\theta_\varsigma && && (2.65)\\
s.t. \quad & && && \\
& \textstyle\sum_{\pi\in P}\sum_{\varsigma\in S^\pi} w_{ij}^\varsigma\theta_\varsigma \le u_{ij} && (i,j)\in A && (2.66)\\
& \textstyle\sum_{\varsigma\in S^\pi}\theta_\varsigma = 1 && \pi\in P && (2.67)\\
& \theta_\varsigma \ge 0 && \varsigma\in S && (2.68)
\end{aligned}$$

where c^ς is the cost of path ς and w_{ij}^ς is the amount of flow to be shipped for the OD π in the arc (i,j) if the arc belongs to the path p_ς^π, and 0 otherwise. In other words, possibly by a simple scaling of the variables one obtains the well-known *arc-path formulation* of MMCF. Note that this specific shape naturally occurs in a number of practical MMCF problems, such as these coming from routing in telecommunication networks, and for these then (MMCF-MP$_0$) coincides with (MMCF-MP$_{-1}$). In other words, one can see (MMCF-MP$_{-1}$) as the "truly disaggregated model, and (MMCF-MP$_0$) as a "partial aggregation" of (MMCF-MP$_{-1}$) in which all the OD sharing the same destination (and the same product) are aggregated to form a single component.

This leads to the consideration that any other possible partial aggregation of OD results in a valid model for $f(x)$. That is, one may arbitrarily define an arbitrary set $\mathcal{C}$ of (indexes of) subsets P^κ partitioning P; then, for each $\kappa\in\mathcal{C}$ one may define S^κ as the set of all possible sets of paths obtained by taking precisely one path in each S^π for each $\pi\in P^\kappa$. Then, for each $\zeta\in S^\kappa$ it is natural to define $c^\zeta \doteq \sum_{\varsigma\in\zeta} c^\varsigma$ and $w_{ij}^\zeta \doteq \sum_{\varsigma\in\zeta} w_{ij}^\varsigma$, leading to the *partly aggregated formulation*

$$\begin{aligned}
(\text{MCCF-MP}_{\mathcal{C}}) \quad \min \quad & \textstyle\sum_{\kappa\in\mathcal{C}}\sum_{\zeta\in S^\kappa} c^\zeta\theta_\zeta && && (2.69)\\
s.t. \quad & && && \\
& \textstyle\sum_{\kappa\in\mathcal{C}}\sum_{\zeta\in S^\kappa} w_{ij}^\zeta\theta_\zeta \le u_{ij} && (i,j)\in A && (2.70)\\
& \textstyle\sum_{\zeta\in S^\kappa}\theta_\zeta = 1 && \kappa\in\mathcal{C} && (2.71)\\
& \theta_\zeta \ge 0 && \zeta\in S && (2.72)
\end{aligned}$$

Note that once all the OD has been aggregated in the original set of $|K|$ commodities (if any such thing exist), is still possible to aggregate a group of multi-origin multi-destination commodities into a single component; clearly, (MMCF-MP$_1$) is the limit situation opposite to (MMCF-MP$_{-1}$).

However, it is important to remark that "super-disaggregating" the commodities in (MMCF-MP$_{-1}$) is only possible in a natural way under two conditions:

1. individual capacity constrains are *not* tight;
2. commodities are naturally destination-specific (or, for that matter, origin-specific) ones.

Both things are needed; indeed, while it is possible to make any problem a destination-specific one, this implies creating commodity-specific capacities on the "artificial" arcs thusly created. Thus, if the problem has multiple destinations, individual capacities need to be created even if they were not originally present. However, it is still possible to "super-disaggregate" a problem even if there are individual capacity constraints; it is "just" necessary to consider them as complicating constraints and move them to the master. This makes it possible to apply a reformulation like (MMCF-MP$_{-1}$), and more in general like (MMCF-MP$_{\mathcal{C}}$), to any MMCF. For (MMCF-MP$_{-1}$), for instance, assume that the flows belonging to any arbitrary subset P' of P (e.g. corresponding to one original commodity $k \in K$ that has been disaggregated) share a specific capacity $u'_{ij} < u_{ij}$ on any arc (i, j); then, one only have to add the constraint

$$\sum_{\pi \in P'} \sum_{\varsigma \in S^\pi} w_{ij}^{\varsigma} \theta_{\varsigma} \leq u'_{ij} \tag{2.73}$$

to the master problem. An example of this construction is provided in Fig. 2.4.

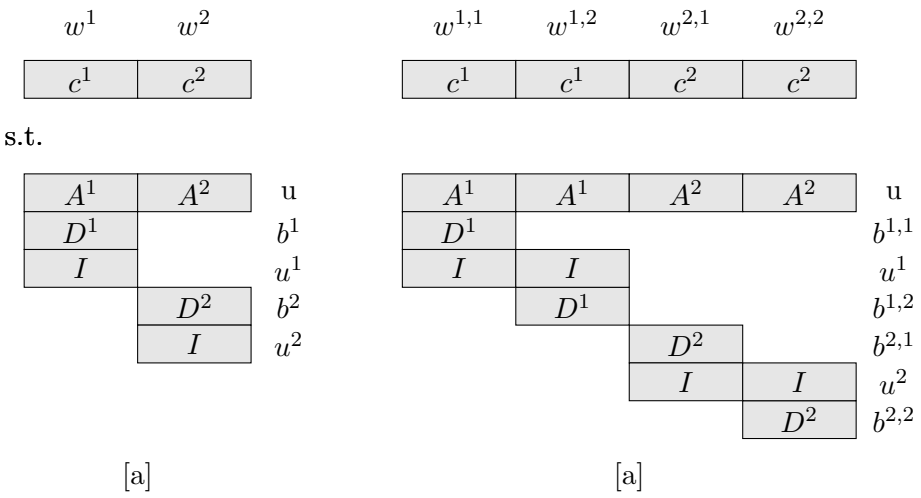

FIGURE 2.4: An visual example, the structure of LP for an MMCF with two commodities ([a]) and the correspondent, OD based, formulation ([b]). Assuming that each original commodity is split in two, the individual capacity constraints becomes complicating as they couples otherwise independent blocks composed by the flow conservation constraints.

Of course, in case one is disaggregating an original set K of commodities and each arc $(i, j) \in A$ has "tight" individual capacities u_{ij}^k for each $k \in K$, this would amount at adding $|K| \cdot |A|$ constraints to the master problem, which is likely to have a relevant impact from the computational viewpoint. However, one benefit is that in doing so the pricing problem become shortest-path ones, as opposed to the significantly more costly Min-Cost Flow ones that need to be solve if individual capacities are left in the sub-problem. This clearly has computational trade-offs that are hard to predict in theory and therefore need to be explored computationally (and to the best of our knowledge never have). Thus, the treatment of individual capacity constraints provide another way to generate different decomposition approaches for MMCF. Indeed, when "active"

individual capacity constraints are present one has the choice (but is not forced to) of increasing the "decomposability" (number of subproblems) of the formulation, and reducing the complexity of the subproblems, at the cost of increasing the master problem size. This is a general principle: easy constraints can be treated as complicating (but not viceversa) in a decomposition approach.

2.4 Computational Results

2.4.1 Preliminary Notes

Tests has been performed on an Intel(R) Xeon(R) CPU with 3.10GHz and 16GB of RAM, under a GNU/Linux and using the `C++` generalized bundle code developed by Antonio Frangioni (see [2, 6, 12, 25] for other usages) and available at: `http://www.di.unipi.it/~frangio/`.

The pricer consists in a min-cost flow solver from the `MCFClass` project ([26]), slightly modified as follows in order to generate several partially aggregated alternative formulations. Recalling that the set of commodities K corresponds to the number of subproblems among which the pricer is decomposed we name $\overline{K}$ the number of OD commodities that forms the equivalent ODP formulation (following the trasformation in [36]). A value $-1 \geq \eta \geq 1, \eta \in \mathbb{R}$ is given as input to the pricer and the number of partially aggregated components is calculated as:

$$|\mathcal{C}| = \begin{cases} \max\left\{ \lceil\ (1-\eta)|K|\ \rceil, 1 \right\} & if \quad \eta \geq 0 \\ \max\left\{ \lceil\ -\eta|\overline{K}|\ \rceil, |K| \right\} & if \quad \eta < 0 \end{cases} \tag{2.74}$$

Partial aggregation is then performed in order to ensure a number of components equal to $|\mathcal{C}|$. If $|\mathcal{C}| = |\overline{K}|$ then a disaggregated component is returned by the pricer of each OD and the reduced master problem is composed by $|\overline{K}|$ disaggregated bundles. If $|K| < |\mathcal{C}| < |\overline{K}|$ a number of components $|\mathcal{C}_k|, |\mathcal{C}|/|\overline{\mathcal{K}}| \leq |\mathcal{C}_k| \geq |\mathcal{C}|/|\overline{\mathcal{K}}| + 1$, is associated to each commodity. Subsets of OD belonging to the same commodity k are randomly aggregated in such a way that the number of OD in a given components is between $|\overline{K}_k|/|\mathcal{C}_k|$ and $|\overline{K}_k|/|\mathcal{C}_k| + 1$ where $|\overline{K}_k|$ is the number of OD in commodity k. A similar criteria is applied when $|\mathcal{C}| \leq |K|$, where commodities are grouped in a such a way that each components aggregate a number of commodities between $|K|/|\mathcal{C}|$ and $|K|/|\mathcal{C}| + 1$. Note that if the commodities are already defined as OD, any $\eta < 0$ will results in a number of component $|\mathcal{C}|$ equal to the original number of commodities. The `MPSolver`, as it is in the mentioned code, can deal with any partially aggregated bundle the pricer is able to provide.

Partially aggregated stabilized DW decomposition approaches has been compared with the direct resolution of the LP formulation by `CPLEX 12.51`. We refer to the latter

as "direct" approach. We tuned the parameter for the direct approach by performing tests among a subset of the instances presented here. The best performances has been obtained using the dual simplex algorithm with general scaling for network extraction (`CPX_PARAM_NETFIND` parameter set to `CPX_NETFIND_SCALE`). Quite surprisingly, the network simplex algorithm performances were poor if compared to the dual simplex as well as to the automatic parameter setting.

Same CPLEX version has been used to solve the reduced master problem. In this case, no parameter tuning has been done, and all the results presented here have been obtained with automatic choice of algorithm and parameter settings. A "boxstep" stabilization has been applied to all the (partially aggregated) reduced master problems and the resulting linear program has been solved using `CPLEX 12.51`. For all the methods, optimality tolerance is set to $1e^{-6}$.

2.4.2 Notations

If not otherwise specified, all times are reported in seconds. In the tables, columns "CPX a." and "CPX t." are referred to the direct approach. More precisely, numbers in column "CPX a." (CPLEX automatic settings) report the total computing time for the standard parameter settings while numbers in column "CPX t." (CPLEX with tuning of parameter) report the total computing time for dual simplex and general scaling. For each $\eta \in [-1, 1]$ included in the table is then showed the overall time for solving the reduced master problem, on column "Rmp", and for solving the pricer, in column "Pr.". Bold columns named "Tot." shows instead the overall time to solution. We refer to the times in column "Rmp" and "Pr.", and to their sums, as "*net*" times in the sense that those numbers directly measure the computation required by combinatorial optimization algorithm to provide the solutions of relative linear problems. The difference between the total time and the net time should tend to zero. Anyway, for some classes of instances a significant part of the time measured is spent outside the linear optimization. We assume that this "overhead" time can be strongly reduced by sharpening the data structures in the code. Therefore, keeping in mind that data storing and/or management outside the LP solver must be handled with attention especially when the number of commodities becomes high, we rather focus our attention on net times. Togheter with net and total execution times, the number of iteration is showed in columns named "It.". A single iteration time limit has be fixed to 3600 seconds for the decomposition based approaches, while a global time limit of 6400 seconds has been set for the direct approach. "TL" in the tables indicates that the time limit has been reached for the corresponding instance. Similarly, an iteration limit of 100000 has been set, with "IL" indicating that the limit has been reached. When instances appearing in a table with similar characteristics are grouped in a single row, columns named "Slv." eventually report the number of instances solved

to optimality for a given group. Averages also considers times and iteration numbers for instances not solved to optimality for time or iteration limit.

For what concerns the tables that summarize the characteristics of a given set of instances, numbers in column "Mut.%" display the percentage of mutual capacity constraints (2.52) that are "active" in the sense that are tight at the optimum for the direct approach.

Finally, for what concern networks Figures, if not differently specified black squares represents nodes with a non-zero balance for at lest one commodity, and the length of the arcs does not reflect their costs.

2.4.3 Results

MMCF problems cover a wide area of possible applications, typically with network topologies and commodities definition different one from the others. The test bed presented here aim to give an as much as possible comprehensive representation of MMCF problems arising in different contest.

2.4.3.1 The Mnetgen Instances

The *"Mnetgen"* is a well known random generator of MMCF instances and Mnetgen generated instances has been already used for testing the effectiveness of several solvers (see among the others [7, 20, 21, 10, 43]). We performed our tests using some of the instances belonging to the "standard" set also available at `"http://www.di.unipi.it/optimize/Data/MMCF.html"`. We included in our tests a total number of 204 instances with up to 256 nodes. In a Mnetgen instance all the commodities are multi-source multi-destination, in particular each commodity have a non-zero balance for all the nodes in the graph (see Fig. 2.5 on the left).

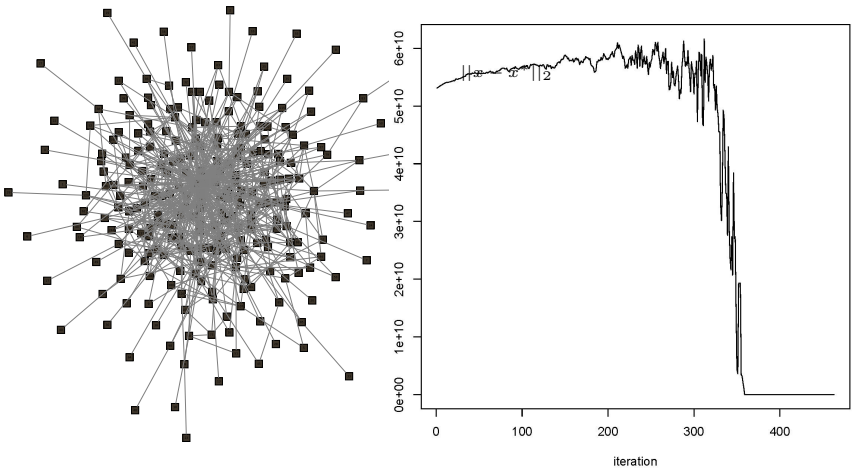

FIGURE 2.5: Relatively to a 256-16 Mnetgen instance: an automatically generated layout, on the left, and the normalized distance between the dual solution of the (unstabilized disaggregated) reduced master problem and the optimal dual solution, after every iteration, on the right.

As appear clear looking at Tables 2.1 and 2.2, this class of instances turns out to be particularly difficult if approached using decomposition methods. The structure of the graph, together with the presence of quite few and complex commodities, can be addressed as one of the reasons that make Mnetgen instances more adapt to a direct, brute-force, approach. Moreover, as displayed on the right side of Fig. 2.5, degeneracy may result here in reduced master problem dual solution strong oscillatory behaviours, when stabilization is absent. We already mentioned how this is a known origin o slow-converge in column generation. The extremely high number of iterations required by the aggregated model confirms such considerations and has been already showed (see e.g. [21]) how most effective stabilization terms may results in very strong improvements of the overall performances. Even considering the "boxstep" method, we feel that there is room for significant improvements simply by fine parameter tuning.

Anyway, the focus of this work is conversely to show how disaggregation can improve the trade off between convergence speed and the computing time for single iteration. Among the partial aggregation levels tested here ($\eta = 0, 0.2, 0.4, 0.6, 0.8, 1$), the best overall performances has been obtained in fact for $\eta = 0$. We measured a big "jump" going from $\eta = 0.8$ to $\eta = 1$. This has been observed also for other classes of instances, especially for what concern the number of iterations, and seems suggest that small improvements of the quality of model $f_{\mathcal{B}}$ are in some case enough for a consistent speed up of the convergence.

Note that almost the entire computing time is spent to solve the reduced master problem. This is at least partially due to an high percentage of tight mutual capacity constraints. We measured that the percentage of active mutual capacity constraints

rises together with the number of commodities, going from roughly the 8% for the 64-4 instance and reaching almost the 40% for the 256-256 instances.

	Average time to solution and number of iterations																
					$\eta = 0$					$\eta = 0.2$					$\eta = 0.4$		
Instance	CPX a.	**CPX t.**	Pr.	Rmp	**Tot.**	**It.**	Slv.	Pr.	Rmp	**Tot.**	**It.**	Slv.	Pr.	Rmp	**Tot.**	**It.**	Slv.
64-4	0.01	**0.01**	0	0.01	**0.02**	**12.92**	12	0	0.01	**0.02**	**13.25**	12	0.01	0.01	**0.02**	**15.17**	12
64-8	0.01	**0.01**	0.01	0.01	**0.03**	**14.42**	12	0.01	0.01	**0.04**	**17**	12	0.01	0.01	**0.04**	**20.25**	12
64-16	0.02	**0.02**	0.02	0.03	**0.11**	**19.25**	12	0.02	0.05	**0.14**	**24.09**	12	0.02	0.07	**0.17**	**30.34**	12
64-32	0.05	**0.05**	0.04	0.13	**0.37**	**22.34**	12	0.04	0.18	**0.48**	**29.25**	12	0.05	0.2	**0.51**	**33.5**	12
64-64	0.17	**0.17**	0.07	1.18	**1.88**	**21.84**	12	0.06	1.35	**2.16**	**26.25**	12	0.09	0.9	**1.72**	**31.34**	12
128-4	0.01	**0.01**	0.02	0.03	**0.09**	**32.92**	12	0.01	0.03	**0.1**	**34.25**	12	0.01	0.04	**0.11**	**41.75**	12
128-8	0.03	**0.03**	0.04	0.18	**0.36**	**47.59**	12	0.04	0.22	**0.42**	**54.42**	12	0.05	0.23	**0.47**	**67.59**	12
128-16	0.07	**0.07**	0.08	1.68	**2.16**	**50.09**	12	0.09	1.69	**2.33**	**69.75**	12	0.1	1.23	**1.83**	**79.09**	12
128-32	0.32	**0.33**	0.21	17.69	**19.24**	**56.34**	12	0.26	24.21	**26.15**	**74.42**	12	0.27	22.96	**24.89**	**89.09**	12
128-64	0.93	**0.88**	0.31	53.58	**56.65**	**45.09**	12	0.38	57.29	**61.04**	**57**	12	0.43	57.38	**60.85**	**67.67**	12
128-128	1.63	**1.7**	0.49	86.82	**91.47**	**34.92**	12	0.59	111.17	**117.47**	**43.09**	12	0.66	112.46	**118.44**	**51.59**	12
256-4	0.04	**0.04**	0.08	5.89	**6.47**	**115.84**	12	0.09	6.09	**6.68**	**118**	12	0.11	8.13	**8.84**	**169.17**	12
256-8	0.07	**0.07**	0.22	32.12	**33.91**	**140.42**	12	0.3	47.81	**50.09**	**183.84**	12	0.37	77.52	**80.09**	**251.92**	12
256-16	0.28	**0.28**	0.49	136.8	**141.7**	**147.59**	12	0.7	208.49	**215.28**	**217.84**	12	0.81	236.05	**243.12**	**270.42**	12
256-32	1.08	**1.08**	0.8	350.39	**359.12**	**116**	12	1.08	571.55	**583.3**	**159**	12	1.27	609.88	**621.68**	**199.5**	12
256-64	1.7	**1.86**	1.23	613.11	**629.28**	**82.34**	12	1.47	843.56	**862.25**	**106.09**	11	1.71	882.12	**900.04**	**130.5**	11
256-128	4.57	**3.99**	1.69	583.79	**606.59**	**53.67**	12	1.97	761.25	**786.78**	**68.42**	12	2.12	804.48	**825.11**	**77.42**	12
256-256	10.43	**8.4**	2.63	758.88	**797.85**	**42.46**	11	2.94	831.95	**874.08**	**52.19**	11	3.3	933.25	**973.65**	**62**	11
All	1.19	**1.06**	0.47	146.8	**152.63**	**58.52**	215	0.56	192.61	**199.38**	**74.71**	214	0.63	208.16	**214.53**	**93.56**	214

TABLE 2.1: Average times and number of iterations to optimality for Mnetgen instances: direct approach and partially aggregated bundles with $\eta = 0, 0.2, 0.4$.

	Average time to solution and number of iterations																
					$\eta = 0.6$					$\eta = 0.8$					$\eta = 1$		
Instance	CPX a.	**CPX t.**	Pr.	Rmp	**Tot.**	**It.**	Slv.	Pr.	Rmp	**Tot.**	**It.**	Slv.	Pr.	Rmp	**Tot.**	**It.**	Slv.
64-4	0.01	**0.01**	0.02	0.04	**0.02**	**18.42**	12	0.01	0.01	**0.02**	**29.25**	12	0.01	0.01	**0.02**	**40.59**	12
64-8	0.01	**0.01**	0.01	0.01	**0.05**	**24.17**	12	0.01	0.03	**0.07**	**39.42**	12	0.02	0.03	**0.08**	**80.25**	12
64-16	0.02	**0.02**	0.03	0.06	**0.18**	**34.84**	12	0.03	0.08	**0.21**	**48.75**	12	0.07	0.15	**0.35**	**120**	12
64-32	0.05	**0.05**	0.05	0.21	**0.54**	**43.17**	12	0.06	0.27	**0.58**	**60.34**	12	0.31	0.91	**1.63**	**280.42**	12
64-64	0.17	**0.17**	0.08	0.8	**1.74**	**39.25**	12	0.14	0.94	**1.69**	**57.92**	12	1.12	5.96	**8.43**	**583.75**	12
128-4	0.01	**0.01**	0.03	0.1	**0.14**	**56.25**	12	0.03	0.07	**0.16**	**75.59**	12	0.03	0.08	**0.16**	**77.92**	12
128-8	0.03	**0.03**	0.08	0.47	**0.49**	**72.25**	12	0.07	0.38	**0.68**	**117.84**	12	0.13	0.67	**1.1**	**227.25**	12
128-16	0.07	**0.07**	0.22	4.54	**2.14**	**99.09**	12	0.21	2.02	**2.85**	**156.09**	12	0.86	11.59	**13.9**	**812.67**	12
128-32	0.32	**0.33**	0.41	34.33	**28.1**	**124.75**	12	0.58	32.97	**35.92**	**215.75**	12	7.07	273.93	**289.8**	**2763.75**	12
128-64	0.93	**0.88**	0.64	79.68	**70.95**	**88.92**	12	0.94	101.35	**106.39**	**164.75**	12	22.5	889.14	**935.87**	**4779.84**	9
128-128	1.63	**1.7**	0.59	123.91	**153.69**	**68.09**	12	1.21	158.34	**165.31**	**106**	12	34.88	993.25	**1058.76**	**2848.92**	9
256-4	0.04	**0.04**	0.53	112.63	**12.55**	**219**	12	0.27	25.77	**27.04**	**471.59**	12	0.24	24.27	**25.48**	**520.17**	12
256-8	0.07	**0.07**	0.48	101.35	**91.59**	**281.67**	12	0.96	294.11	**298.71**	**632.92**	12	2.05	708	**714.91**	**1685.34**	10
256-16	0.28	**0.28**	0.43	90.66	**459.03**	**377.5**	12	1.98	853.13	**864.64**	**669.92**	11	4.87	1124.3	**1136.62**	**1998.42**	10
256-32	1.08	**1.08**	0.39	80.51	**880.81**	**280.5**	11	2.29	958.74	**970.81**	**392.59**	9	11.21	1782.09	**1805.73**	**1937.75**	6
256-64	1.7	**1.86**	0.36	71.02	**1431.71**	**180.92**	8	2.88	1132.63	**1148.88**	**230.92**	9	14.86	1777.76	**1806.03**	**1336.17**	6
256-128	4.57	**3.99**	0.33	61.54	**962.49**	**95.59**	10	3.48	1151.04	**1170.21**	**142.84**	9	20.12	1776.07	**1811.94**	**1003.84**	6
256-256	10.43	**8.4**	0.3	52.05	**1091.96**	**66.82**	9	4.64	1093.33	**1119.17**	**100**	9	33.4	1760.53	**1818.35**	**860.19**	6
All	1.19	**1.06**	0.78	281.87	**288.23**	**120.38**	206	1.1	322.51	**328.52**	**205.89**	203	8.54	618.26	**634.95**	**1216.03**	182

TABLE 2.2: Average times and number of iterations to optimality for Mnetgen instances: direct approach and partially aggregated bundles with $\eta = 0.6, 0.8, 1$.

2.4.3.2 "JLF" instances

This set of instances, grouped in six subsets with similar topologies (see Fig. 2.6), replicates the test bed introduced by Jones et al. in [36] (see also [21, 43]).
For this set of instances we performed the experiments for η going from -1 to 1, and as shown in Tables 2.3 and 2.4 the fully disaggregated model leads to better overall performances in terms of net computational time, with respect to the other levels of aggregation. This seems to confirm that conclusions in [36] are valid even disposing of more powerful machines and even applying boxstep stabilization. Nevertheless, consideration similar to what said for Mnetgen case can be done for what concern the possibility of improving the effectiveness of the stabilization tool. Most of the JLF instances are rather small for the today standards, with execution times that in some cases are below the centisecond. Is anyway impressive the gap between aggregated ($\eta = 1$) and disaggregated $\eta = -1$ models performances, especially for some cases as the "veh8" instance. Note that, the number of iteration tends to remain similar for some η windows while tends to "jump" for some particular values, most of the time going from $\eta = 0.8$ to $\eta = 1$ or going from $\eta = -0.2$ to $\eta = 0$. We recall that the number of disaggregated components in the bundles grows linearly with η. This behaviour is particularly interesting as one may tray to exploit the fact that η can be increased within a "flat" window, obtaining a reduction of the complexity of the reduced master problem without compromising the convergence speed. In fact also looking this particular set of instances, in some sporadic cases as for chen5, we observed a slight but encouraging better net time for some $\eta > -1$.

Anyway, as happening for the Mnetgen test bed and with similar considerations about the potential of stabilization tools, the direct approach appears to be the best choice. The reasons behind such outcome can be found in a limited number of commodities, at least for some instances, and in an extremely high percentage of active mutual capacity constraints. In fact, recalling that the JLFproblems does not define individual capacities, on average approximatively 65% of the mutual capacity constraints are tight, and in some cases (e.g. the "chen" and "psp" instances) this value reaches values near to 100%.

	Average time to solution and number of iterations																	
			$\eta = -1$				$\eta = -0.8$				$\eta = -0.6$				$\eta = -0.4$			
Instance	CPX a.	**CPX t.**	Pr.	Rmp	**Tot.**	**It.**	Pr.	Rmp	**Tot.**	**It.**	Pr.	Rmp	**Tot.**	**It.**	Pr.	Rmp	**Tot.**	**It.**
term	0.03	**0.03**	0	0.17	**0.75**	**11**	0.01	0.11	**0.33**	**11.5**	0.01	0.1	**0.33**	**11.5**	0.01	0.07	**0.22**	**12**
alk.HALF	0.2	**0.2**	0.16	2.22	**17.03**	**28**	0.21	3.45	**12.8**	**42**	0.28	3.69	**13.04**	**42**	0.29	5	**19.16**	**58**
alk.TWO	3.01	**2.91**	3.12	374.49	**1332.22**	**107**	4.28	1778.53	**2335.18**	**146**	4	1799.92	**2315.17**	**141**	4.25	2189.81	**2485.43**	**151**
assad	0.01	**0.01**	0	0.01	**0.01**	**11.67**	0	0	**0.01**	**11.67**	0	0.01	**0.01**	**11.67**	0.01	0	**0.01**	**12.34**
veh8	0.06	**0.06**	0.03	0.97	**2.73**	**23**	0.05	1.5	**2.79**	**30**	0.06	1.46	**2.74**	**30**	0	1.34	**2.27**	**31**
chen	0.01	**0.04**	0.01	2.98	**4.08**	**70.29**	0.01	3.19	**4.03**	**99.58**	0.01	3.15	**3.98**	**99.58**	0.02	2.25	**2.87**	**117.86**
All	0.19	**0.19**	0.18	21.03	**72.9**	**39.53**	0.25	95.08	**125.32**	**53.64**	0.24	96.2	**124.25**	**53.37**	0.25	116.44	**133.07**	**61.79**

TABLE 2.3: Average times in seconds for JLF instances, partial aggregation with η from -1 to -0.4.

	Average time to solution and number of iterations																	
			$\eta = -0.2$				$\eta = 0$				$\eta = 0.2$				$\eta = 0.4$			
Instance	CPX a.	**CPX t.**	Pr.	Rmp	**Tot.**	**It.**	Pr.	Rmp	**Tot.**	**It.**	Pr.	Rmp	**Tot.**	**It.**	Pr.	Rmp	**Tot.**	**It.**
term	0.03	**0.03**	0.01	0.05	**0.12**	**13**	0.01	0.04	**0.1**	**23.5**	0.01	0.03	**0.1**	**27.34**	0.01	0.04	**0.11**	**29.5**
alk.HALF	0.2	**0.2**	0.38	23.34	**36.25**	**87**	3.88	1426.96	**1472.86**	**743**	7.31	TL	**TL**	**TL**	9.09	TL	**TL**	**TL**
alk.TWO	3.01	**2.91**	4.12	TL	**TL**	**TL**	6.81	TL	**TL**	**TL**	7.52	TL	**TL**	**TL**	7.88	TL	**TL**	**TL**
assad	0.01	**0.01**	0.01	0	**0.01**	**12.34**	0	0.01	**0.01**	**12.34**	0.01	0.01	**0.02**	**21**	0	0.01	**0.02**	**22**
veh8	0.06	**0.06**	0.1	4.1	**6.02**	**60**	1.29	487.08	**506.35**	**1232**	1.28	488.71	**508.53**	**1232**	2.81	866.68	**893.55**	**2562**
chen	0.01	**0.04**	0.02	1.9	**2.36**	**212**	0.03	2.19	**2.72**	**266.15**	0.03	2.53	**3.22**	**354.15**	0.04	2.91	**3.49**	**405**
psp	0.02	**0.02**			-		0.14	3.07	**3.98**	**603.5**	0.13	2.93	**3.86**	**610.67**	0.17	3.02	**3.97**	**862.5**
All	0.15	**0.15**	0.25	186.25	**193.26**	**223.63**	0.53	220.88	**225.37**	**318.21**	0.69	283.6	**289.35**	**373.88**	0.85	320.64	**326.55**	**470.22**

TABLE 2.4: Average times in seconds for JLF instances, partial aggregation with η from -0.2 to -0.4.

	Average time to solution and number of iterations													
			$\eta = -0.6$				$\eta = 0.8$				$\eta = 1$			
Instance	CPX a.	**CPX t.**	Pr.	Rmp	**Tot.**	**It.**	Pr.	Rmp	**Tot.**	**It.**	Pr.	Rmp	**Tot.**	**It.**
term	0.03	**0.03**	0.01	0.06	**0.12**	**39.5**	0.01	0.07	**0.17**	**59.17**	0.04	0.19	**0.39**	**130.67**
alk.HALF	0.2	**0.2**	8.95	TL	**TL**	**TL**	10.09	TL	**TL**	**TL**	12.4	TL	**TL**	**TL**
alk.TWO	3.01	**2.91**	9.01	TL	**TL**	**TL**	11.24	TL	**TL**	**TL**	26.44	TL	**TL**	**TL**
assad	0.01	**0.01**	0	0.01	**0.02**	**20.34**	0	0.02	**0.02**	**29.67**	0.01	0.02	**0.04**	**61**
veh8	0.06	**0.06**	2.87	826.92	**853.36**	**2562**	3.73	961.71	**984.31**	**3246**	3.44	1007	**1030.4**	**3246**
chen	0.01	**0.04**	0.06	4.41	**5.09**	**572.15**	0.08	8.39	**9.29**	**920.43**	0.3	32.95	**35.36**	**2709.58**
psp	0.02	**0.02**	0.2	2.96	**4.18**	**1142**	3.23	73.84	**86.69**	**1251.8**	3.3	77.07	**90.63**	**1444.8**
All	0.15	**0.15**	0.9	319.94	**325.15**	**596.4**	1.8	343.63	**351.15**	**745.1**	2.58	352.21	**361.17**	**1382**

TABLE 2.5: Average times in seconds for JLF instances, partial aggregation with η from 0.6 to 1.

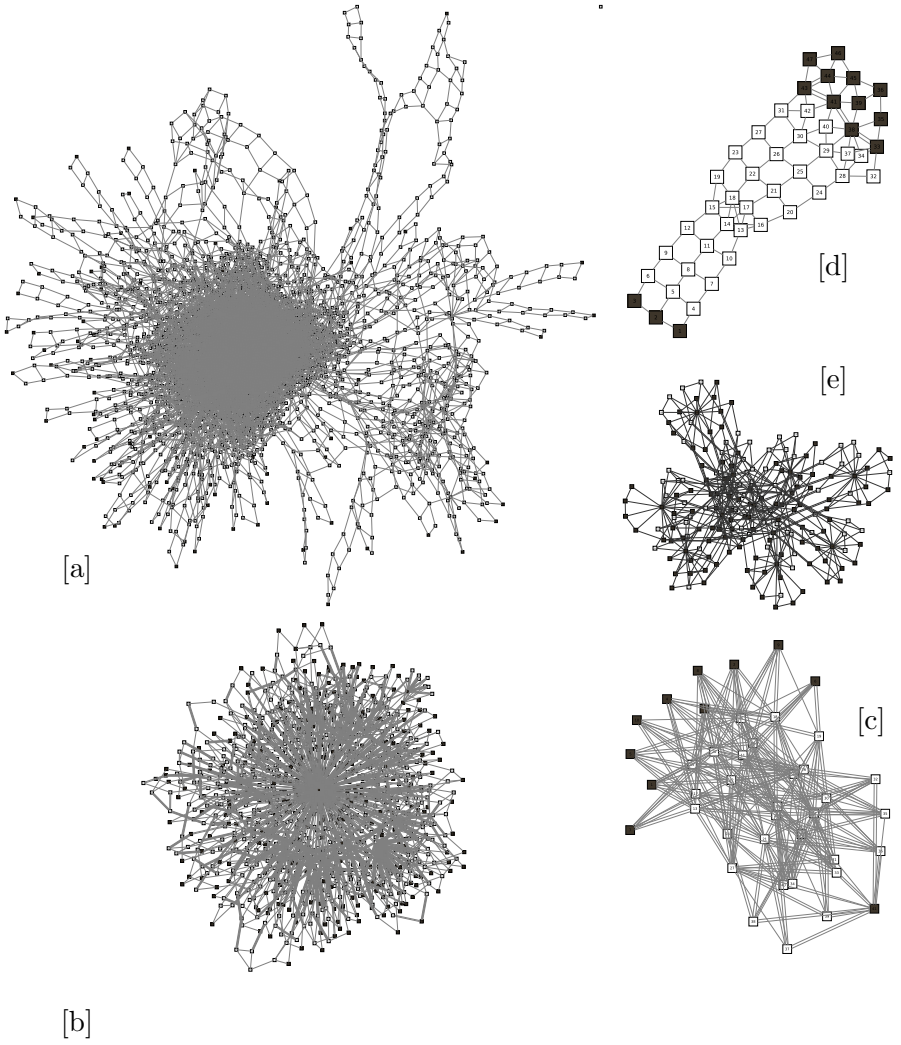

FIGURE 2.6: Automatically generated layouts for some of the JLF instances: [a] veh8, [b] alkHALF, [c] chen2 (same topology for "PSP" and "DSP"), [d] assad1.6k, and [e] 10term.50.

2.4.3.3 "Dimacs2pprn" Instances

This two groups of instances has been generated using the *Dimacs2pprn Generator* developed by Jordi Castro (see ([9, 11]) and available at http://www.di.unipi.it/optimize/Data/MMCF.html#d2p. The generator takes as input a single-commodity min-cost flow instance and constructs a MMCF instance with k copies of the original commodity having deficits and capacities "scaled" of a random quantity and having randomly twisted costs. Commodities are therefore here defined as OD, and sources and destinations corresponds to the same nodes for all the commodities (see also Fig. 2.7). In turn, the single commodity instances has been synthesized by using two different generators, hence we have:

- GoTo-Dimacs2pprn: instances belonging to this group has been built upon two single-commodity flow problems created with the "*Grid-on-Torus*" network generator implemented by A. Goldberg and available at `http://www.zib.de/en/services/web-services/mp-testdata/generators.html` (see [31, 45, 9] for a more detailed description). GoTo-Dimacs2pprn randomly generated MMCF instances taking part in our test bed have 100, 400 or 800 commodities with default `size` parameter (see `dmx2pprn.C`) and with `factor` parameter set as in Table 2.6. On the whole, a number of 35 GoTo-Dimacs2pprn instances composes the test bed.

Instances	Arcs	Nodes	Commodities	`fctr`	Instances #	Av. Mut.%
Goto6_6_1-100	1024	64	100	1000	5	13.09
Goto6_6_1-400	1024	64	400	4000	5	13.7
Goto6_6_1-800	1024	64	800	8000	5	13.97
Goto8_8_1-10	2048	256	10	100	5	14.12
Goto8_8_1-100	2048	256	100	1000	5	14.42
Goto8_8_1-400	2048	256	400	4000	5	14.83
Goto8_8_1-800	2048	256	800	16000	5	5.01

TABLE 2.6: The GoTo-Dimacs2pprn instance set.

- RMFGen-Dimacs2pprn instances: the RMFGen is a random max flow, single-commodity, instance generator developed by Goldfarb and Grigoriadis (see [32]). The generated network is divided in "frames", in each frame all the nodes are connected with their neighbours. Nodes of a frame are then connected one to one with the nodes of next frame using a random permutation of those nodes. The origin is the lower left node of the first frame while the destination is the upper right node of the last frame (see Fig. 2.7). Demands, capacities and costs of the arcs are integer and randomly distributed between two limit values (see also the `readme` distributed along with the generator source code at `http://www.di.unipi.it/optimize/Data/MMCF.html#d2p`). Dimacs2pprn generator has been fed with four RMFGen instances having 64, 128, 256 or 512 nodes (see Table 2.7). For each RMFGen instance, four groups of MMCF problems has been generated with increasing number of commodities. Each group counts three instances with same parameters setting but with different seeds for Dimacs2pprn random choices. On the whole, a number of 48 RMFGen-Dimacs2pprn instances composes the test bed.

Instances	Arcs	Nodes	Commodities	fctr	Instances #	Av. Mut.%
R-4-4-512	240	64	512	10	3	7.04
R-4-4-1024	240	64	1024	10	3	10.42
R-4-4-2048	240	64	2048	20	3	10.42
R-4-4-4096	240	64	4096	20	3	29.17
R-4-8-2048	496	128	2048	100	3	4.23
R-4-8-4096	496	128	4096	200	3	4.29
R-4-8-8192	496	128	8192	200	3	7.67
R-4-8-16384	496	128	16384	400	3	8.78*
R-7-6-2048	1008	256	2048	100	3	10.66
R-7-6-4096	1008	256	4096	500	3	5.13
R-7-6-8192	1008	256	8192	500	3	8.7
R-7-6-16384	1008	256	16384	1000	3	9.43
R-8-8-2048	2240	512	2048	10	3	6.19
R-8-8-4096	2240	512	4096	100	3	0.57
R-8-8-8192	2240	512	8192	100	3	2.02*
R-8-8-16384	2240	512	16384	200	3	1.94*

TABLE 2.7: The RMFGen-Dimacs2pprn instance set. The symbol (*) means that optimal solutions are unknown so "Mut. %" has been evaluated for the best (suboptimal) solution at time limit.

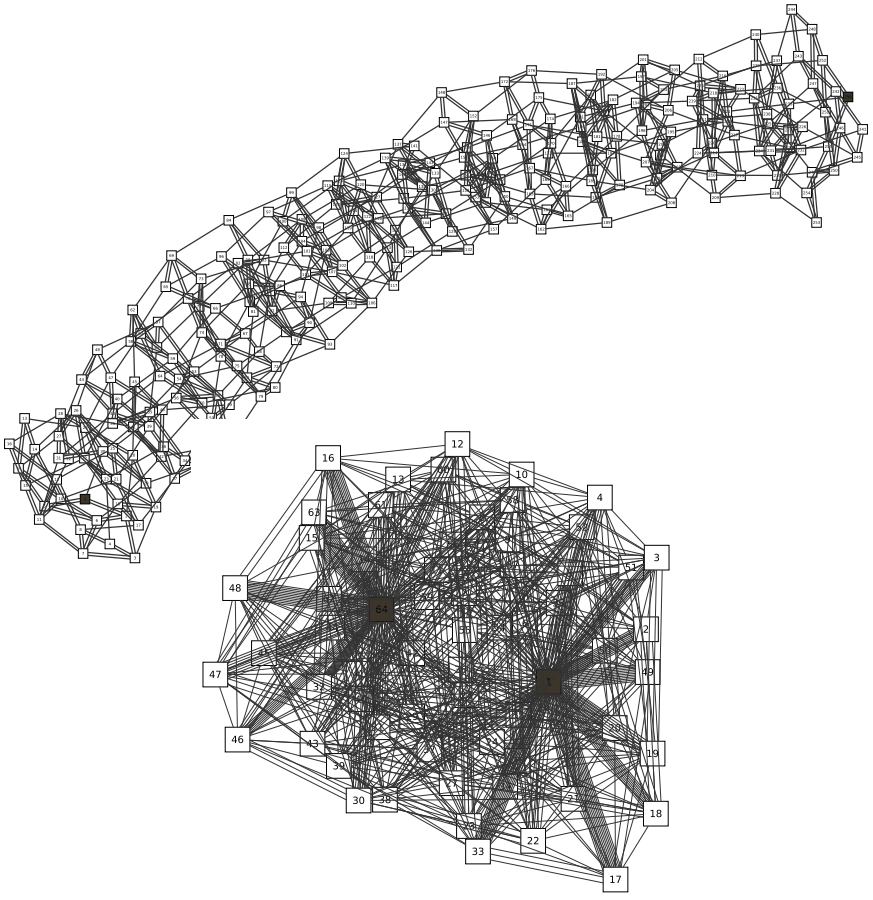

FIGURE 2.7: Automatically generated layout for a "R-7-6-2048" instance, on the top and a "goto6_6_1" instance, at the bottom.

For what concern the GoTo-Dimacs2pprn instances, results are summarized in Table 2.8 and 2.9. In general, we can see that for this set of instances a decomposition method on a properly disaggregated bundle ensures rather good performances if compared with the direct approach. This can find a motivation in the rather easy structure of the relatively many commodities. As for the JLF and Mnetgen case, a more disaggregated

bundle gives in general better performances. Nevertheless, the overall (net) execution times are often quite similar for $\eta \leq 0.5$, and we feel that further investigation on larger instances might be interesting to better understand which level of partial aggregation allows in general better performances.

Seems anyway remote the possibility of obtaining competitive results by using a fully aggregated bundle, as the number of iterations is in this case extremely higher then any other η setting. Is quite interesting the case of the Goto8-8-1-800 instance set. The high number of commodities forming the instances of this set leads to high computational time for the direct approach. On the other hand, the as well high "`fctr`" parameter, and consequently the low percentage of active mutual capacity constraints, make this instances rather easy for decomposition based approaches.

	Average time to solution and number of iterations													
			$\eta = 0$				$\eta = 0.2$				$\eta = 0.4$			
Instance	CPX a.	**CPX t.**	Pr.	Rmp	**Tot.**	**It.**	Pr.	Rmp	**Tot.**	**It.**	Pr.	Rmp	**Tot.**	**It.**
Goto6_6_1-100	0.72	**0.68**	0.12	0.83	**1.29**	**24.6**	0.14	1.03	**1.54**	**29.8**	0.16	1.1	**1.65**	**34.6**
Goto6_6_1-400	15.9	**14.22**	0.3	0.79	**2.23**	**15.25**	0.32	0.94	**2.23**	**16.5**	0.36	1.16	**2.39**	**18.5**
Goto6_6_1-800	112.43	**64.1**	0.5	1.02	**3.96**	**12.4**	0.55	1.24	**3.96**	**14**	0.61	1.6	**3.94**	**15.2**
Goto8_8_1-10	0.13	**0.12**	0.09	2.79	**3.26**	**75.2**	0.11	4.51	**5.15**	**104.4**	0.18	6.46	**7.28**	**136.6**
Goto8_8_1-100	9.04	**5.64**	0.29	2.91	**3.71**	**21**	0.37	4.27	**5.33**	**27.2**	0.41	4.43	**5.6**	**31**
Goto8_8_1-400	194.22	**105.14**	0.88	4.12	**6.74**	**15.6**	1	6.61	**9.37**	**17.6**	1.1	9.5	**12.32**	**19.6**
Goto8_8_1-800	331.75	**326.01**	1.84	2.17	**7.08**	**11.4**	1.94	2.42	**6.98**	**12**	2.13	3.36	**7.83**	**13.2**
All	97.21	**75.45**	0.58	2.13	**4.09**	**25.36**	0.64	3.06	**5.02**	**32.09**	0.72	4.03	**5.96**	**38.98**

TABLE 2.8: Average times to optimality for GoTo instances: direct approach and partially aggregated bundles with $\eta = 0, 0.2, 0.4$

	Average time to solution and number of iterations													
			$\eta = 0.6$				$\eta = 0.8$				$\eta = 0.1$			
Instance	CPX a.	**CPX t.**	Pr.	Rmp	**Tot.**	**It.**	Pr.	Rmp	**Tot.**	**It.**	Pr.	Rmp	**Tot.**	**It.**
Goto6_6_1-100	0.72	**0.68**	0.21	1.38	**2.05**	**43.6**	0.36	0.99	**2.05**	**69.2**	3.01	10.46	**20.51**	**925.8**
Goto6_6_1-400	15.9	**14.22**	0.42	1.77	**2.98**	**21.75**	0.67	4.95	**6.6**	**31.75**	16.66	26.55	**79.56**	**1272**
Goto6_6_1-800	112.43	**64.1**	0.69	2.56	**4.62**	**16.8**	0.98	7.47	**9.96**	**24.6**	44.3	43.83	**183.14**	**1709.2**
Goto8_8_1-10	0.13	**0.12**	0.2	7.66	**8.59**	**164**	0.38	17.11	**18.71**	**300.6**	0.69	20.81	**23.88**	**722**
Goto8_8_1-100	9.04	**5.64**	0.6	4.95	**6.59**	**45**	1.02	11.1	**14.14**	**79.2**	18.79	287.15	**340.27**	**2113.8**
Goto8_8_1-400	194.22	**105.14**	1.37	14.21	**17.38**	**22.8**	2.02	20.73	**25.49**	**35.6**	92.87	364.88	**613.74**	**2690.4**
Goto8_8_1-800	331.75	**326.01**	2.38	5.24	**9.7**	**15**	3.14	9.96	**15.52**	**19.8**	115.25	43.58	**250.27**	**729.2**
All	97.21	**75.45**	0.85	5.5	**7.55**	**47.74**	1.24	10.49	**13.4**	**81.53**	42.39	116.46	**219.92**	**1457.06**

TABLE 2.9: Average times to optimality for GoTo instances: direct approach and partially aggregated bundles with $\eta = 0.6, 0.8, 1$

For what concern the RMFGen-Dimacs2pprn instances, results are summarized in Table 2.10 and 2.11. The number of iterations for this class of instances results particularly similar among different values of $\eta < 1$. As already observed for other classes of instances, slow convergence issues are instead particularly evident for the fully aggregated bundle. We observe an interesting reduction for reduced master problem optimization times as the η increases. This means that in this case the smaller sizes of the corresponding program, and consequently the reduced amount of computation required to solve it, overcompensate the increased number of iterations. This is surprisingly true also for $\eta = 1$. Nonetheless, the net times related with the optimization of the pricer play here a central role, covering in some cases even more than 90% of the overall net time. As the number of iteration increases, being the time for single iteration quite constant, the required time for pricing increases in turn. We can therefore observe that the best trade-off between reduced master problem time (that decreases when η is increases) and pricing time (that on the opposite increases when η is increased) can be obtained in some situation by partial aggregation of the bungle. We recall that being the commodities already defined as single-source single-destination, makes no sense any $\eta < 0$.

	Average time to solution and number of iterations													
			$\eta = 0$				$\eta = 0.2$				$\eta = 0.4$			
Instance	CPX a.	**CPX t.**	Pr.	Rmp	**Tot.**	**It.**	Pr.	Rmp	**Tot.**	**It.**	Pr.	Rmp	**Tot.**	**It.**
R-4-4-512	1.96	**1.55**	0.05	0.09	**0.43**	**6.67**	0.04	0.04	**0.3**	**6.67**	0.06	0.02	**0.21**	**6.67**
R-4-4-1024	9.17	**6.38**	0.1	0.19	**1.4**	**7**	0.11	0.22	**1.14**	**7.34**	0.13	0.2	**0.93**	**8**
R-4-4-2048	43.1	**30.38**	0.21	0.48	**5.1**	**7.34**	0.25	0.47	**4.16**	**8**	0.25	0.36	**2.65**	**8**
R-4-4-4096	321.51	**220.22**	0.49	1.36	**21.95**	**8**	0.55	1.42	**18.47**	**9**	0.57	1.32	**11.43**	**9**
R-4-8-2048	11.43	**7.4**	0.4	0.34	**2.97**	**5**	0.38	0.26	**2.2**	**5**	0.39	0.21	**1.61**	**5**
R-4-8-4096	33.96	**21.66**	0.76	0.52	**9.57**	**5**	0.77	0.5	**6.78**	**5**	0.76	0.46	**4.57**	**5**
R-4-8-8192	2077.04	**1337.62**	2.14	1.95	**65.44**	**7**	2.24	1.95	**48.11**	**7.34**	2.42	1.9	**34.37**	**8**
R-4-8-16384	TL	**6064.75**	4.22	3.7	**336.53**	**7**	4.43	4.03	**223.15**	**7.34**	4.84	4.12	**124.46**	**8**
R-7-6-2048	203.51	**149.05**	1.27	0.6	**5.32**	**5.67**	1.34	0.61	**4.64**	**6**	1.33	0.52	**3.71**	**6**
R-7-6-4096	248.69	**186.89**	2.22	0.73	**12.09**	**5**	2.21	0.67	**9.14**	**5**	2.22	0.67	**6.89**	**5**
R-7-6-8192	2765.73	**1992.45**	4.46	1.69	**39.17**	**5**	4.45	1.62	**28.08**	**5**	4.45	1.5	**19.3**	**5**
R-7-6-16384	TL	**TL**	8.9	3.14	**143.82**	**5**	8.92	3.14	**94.62**	**5**	8.9	2.9	**60.25**	**5**
R-8-8-2048	3836.17	**2697.98**	3.27	1.6	**15.13**	**10**	3.4	1.77	**12.95**	**10.34**	3.5	1.87	**11.05**	**10.67**
R-8-8-4096	1453.48	**1021.34**	3.14	0.69	**15.81**	**5.34**	3.13	0.67	**12.29**	**5.34**	3.13	0.63	**9.51**	**5.34**
R-8-8-8192	TL	**TL**	8.42	2.6	**82.15**	**7**	8.84	2.6	**63.44**	**7.34**	8.85	2.36	**43.23**	**7.34**
R-8-8-16384	TL	**TL**	16.88	5.01	**381.36**	**7**	16.86	5.44	**233.62**	**7**	18.51	5.4	**147.67**	**7.67**
Average	2288.92	**2059.55**	3.56	1.54	**71.14**	**6.44**	3.62	1.59	**47.69**	**6.67**	3.77	1.53	**30.12**	**6.86**

TABLE 2.10: Average times to optimality for RMFGen-Dimacs2pprn instances: direct approach and partially aggregated bundles with $\eta = 0, 0.2, 0.4$

	Average time to solution and number of iterations													
			$\eta = 0.6$				$\eta = 0.8$				$\eta = 0.1$			
Instance	CPX a.	**CPX t.**	Pr.	Rmp	**Tot.**	**It.**	Pr.	Rmp	**Tot.**	**It.**	Pr.	Rmp	**Tot.**	**It.**
R-4-4-512	1.96	**1.55**	0.06	0.01	**0.17**	**7**	0.06	0.02	**0.14**	**7.67**	0.24	0.01	**0.47**	**35**
R-4-4-1024	9.17	**6.38**	0.11	0.12	**0.63**	**8.34**	0.14	0.03	**0.39**	**9.34**	0.89	0.02	**1.61**	**58.67**
R-4-4-2048	43.1	**30.38**	0.26	0.36	**1.85**	**8.67**	0.29	0.21	**1.06**	**9.34**	2.1	0.02	**3.78**	**68.34**
R-4-4-4096	321.51	**220.22**	0.6	1.37	**7.08**	**9.67**	0.68	1.47	**4.24**	**11**	8.99	0.06	**14.99**	**139.67**
R-4-8-2048	11.43	**7.4**	0.38	0.13	**1.11**	**5**	0.38	0.08	**0.81**	**5**	0.59	0.01	**0.95**	**7.67**
R-4-8-4096	33.96	**21.66**	0.76	0.39	**2.94**	**5**	0.75	0.21	**1.8**	**5**	0.86	0	**1.38**	**5.67**
R-4-8-8192	2077.04	**1337.62**	2.41	1.83	**18.75**	**8**	2.72	1.71	**10.55**	**9**	18.8	0.04	**29.95**	**61.67**
R-4-8-16384	TL	**6064.75**	4.85	3.98	**62.4**	**8**	5.23	3.53	**27.45**	**8.67**	37.63	0.05	**60.52**	**62**
R-7-6-2048	203.51	**149.05**	1.42	0.44	**3.17**	**6.34**	1.57	0.34	**2.9**	**7**	12.06	0.06	**16.9**	**54**
R-7-6-4096	248.69	**186.89**	2.36	0.62	**5.63**	**5.34**	2.37	0.45	**4.33**	**5.34**	3.71	0.01	**5.23**	**8.34**
R-7-6-8192	2765.73	**1992.45**	4.76	1.48	**14.28**	**5.34**	5.36	1.43	**11.16**	**6**	27.06	0.03	**38.21**	**30.34**
R-7-6-16384	TL	**TL**	9.5	2.81	**39.62**	**5.34**	10.69	2.8	**25.79**	**6**	68.33	0.04	**96.78**	**38.34**
R-8-8-2048	3836.17	**2697.98**	3.97	2.12	**10.59**	**12**	4.67	3.34	**11.65**	**14**	92.66	8.69	**157.17**	**271**
R-8-8-4096	1453.48	**1021.34**	3.53	0.61	**8.46**	**6**	3.53	0.46	**6.78**	**6**	10.21	0.43	**17.78**	**17.34**
R-8-8-8192	TL	**TL**	9.66	2.61	**32.43**	**8**	10.92	2.61	**24.98**	**9**	109.02	1.96	**186.21**	**88.67**
R-8-8-16384	TL	**TL**	19.38	5.43	**91.24**	**8**	21.86	6.29	**59.21**	**9**	248.05	5.08	**468.97**	**100.67**
Average	2288.92	**2059.55**	4	1.52	**18.77**	**7.28**	4.45	1.56	**12.08**	**8**	40.08	1.03	**68.81**	**66.53**

TABLE 2.11: Average times to optimality for RMFGen-Dimacs2pprn instances: direct approach and partially aggregated bundles with $\eta = 0.6, 0.8, 1$

2.4.3.4 The Planar and Grid instances

This set of instances, available at http://www.di.unipi.it/optimize/Data/MMCF.html#Plnr has been introduced in [39] and are built upon two similar classes of topologies, planar and grid, inspired by telecommunication networks (see also [3]).

- Planar instances: nodes are in this case randomly chosen as points in a plane and arcs link neighbour nodes in such a way that the resulting graph is planar (see Fig.2.8). The origin and destination nodes are randomly chosen, arc costs are euclidean distances while demands and capacities are uniformly distributed in given intervals.

- Grid instances: nodes in the graph are connected to form a a regular grid. Thus, there are four incoming and four outgoing arcs for every internal node. The costs of the arcs, the commodities, and the demands are generated as for the planar networks.

Results for this class of instances are summarized in Tables 2.12 and 2.13. Net execution times confirm the convenience of disaggregated bundle approaches with respect to the direct approach and with respect to partially or totally aggregated bundles. The peculiar behaviour of this class of instances consists in the fact that the number of iterations, going from $\eta = 0.8$ to $\eta = 0$, does not decrease enough to justify the reduction in terms of computational time. In some cases, we even observe a weird situation in which aggregated bundles reduces the number of iterations to optimality. This may be due to the presence of the stabilization that may be more effective for some aggregation level than others. In any case, we may find of interest, as future work, a further investigation on this class of instances to better understand this behaviour. Note finally how the tuned direct approach, with dual simplex and network scaling, does not seem to be very effective when compared to the use of automatic setting.

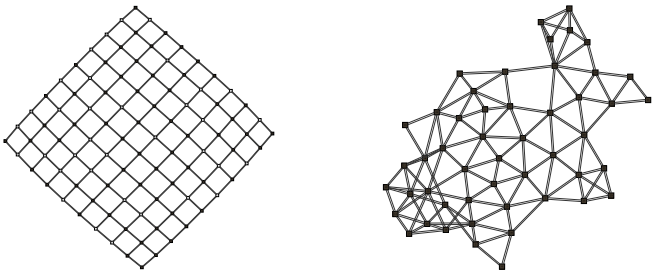

FIGURE 2.8: Automatically generated layout for "grid3" instance on the left and "planar50" instance on the right.

	Time to solution													
			$\eta = 0$				$\eta = 0.2$				$\eta = 0.4$			
Instance	CPX a.	**CPX t.**	Pr.	Rmp	**Tot.**	**It.**	Pr.	Rmp	**Tot.**	**It.**	Pr.	Rmp	**Tot.**	**It.**
grid1	0.01	**0.01**	0	0.01	**0.01**	**8**	0	0	**0.01**	**8**	0	0	**0.01**	**8**
grid2	0.03	**0.02**	0	0	**0.03**	**8**	0	0	**0.02**	**8**	0	0.01	**0.02**	**8**
grid3	0.04	**0.04**	0.01	0.01	**0.03**	**7**	0.01	0	**0.02**	**7**	0	0	**0.02**	**8**
grid4	0.11	**0.11**	0.02	0	**0.06**	**9**	0	0.02	**0.08**	**11**	0.03	0	**0.06**	**11**
grid5	0.41	**0.41**	0.04	0.01	**0.14**	**12**	0.03	0.01	**0.16**	**13**	0.02	0	**0.11**	**11**
grid6	8.78	**10.37**	0.09	0.73	**1.17**	**15**	0.1	0.6	**0.96**	**14**	0.11	0.5	**0.83**	**17**
grid7	54.74	**55.96**	0.24	1.83	**2.93**	**12**	0.27	2.56	**3.66**	**13**	0.28	1.87	**2.94**	**15**
grid8	1745.65	**1954.88**	0.7	17.03	**20.62**	**18**	0.78	18.29	**21.53**	**19**	0.76	18.78	**21.64**	**20**
grid9	TL	**TL**	1.15	34.34	**41.59**	**15**	1.23	48.2	**54.6**	**16**	1.36	54.56	**60.31**	**18**
grid10	TL	**TL**	2.32	51.08	**73.63**	**15**	2.52	59.43	**78.16**	**16**	2.59	68.68	**83**	**17**
grid12	TL	**TL**	7.44	42.71	**145.02**	**11**	7	56.31	**118.49**	**10**	7.25	48.93	**97.1**	**11**
grid14	TL	**TL**	24.37	72.16	**830.92**	**10**	28.36	133.03	**776.8**	**11**	28.79	129.52	**532.14**	**12**
planar30	0.03	**0.03**	0	0	**0.04**	**10**	0.01	0	**0.04**	**10**	0	0.01	**0.03**	**10**
planar50	0.49	**0.51**	0.07	0.12	**0.41**	**12**	0.05	0.06	**0.3**	**12**	0.05	0.02	**0.21**	**13**
planar80	10.33	**10.81**	0.2	0.77	**2.56**	**17**	0.19	0.92	**2.24**	**16**	0.19	0.78	**1.69**	**17**
planar100	43.9	**45.62**	0.36	1.07	**5.96**	**14**	0.38	1.14	**4.24**	**13**	0.34	1.03	**3.07**	**13**
planar150	4239.98	**4346.06**	1.62	21.23	**50.72**	**17**	1.65	24.53	**43.8**	**17**	1.61	23.58	**36.27**	**17**
planar300	TL	**5127.75**	6.77	13.04	**64.92**	**13**	7.31	15.36	**54.28**	**14**	7.01	15.13	**41.74**	**14**
Averages	2117.36	**2064.03**	2.53	14.23	**68.94**	**12.39**	2.78	20.03	**64.42**	**12.67**	2.8	20.19	**48.96**	**13.34**

TABLE 2.12: Average times to optimality for Planar and Grid instances: direct approach and partially aggregated bundles with $\eta = 0, 0.2, 0.4$

	Time to solution													
			$\eta = 0.6$				$\eta = 0.8$				$\eta = 0.1$			
Instance	CPX a.	**CPX t.**	Pr.	Rmp	**Tot.**	**It.**	Pr.	Rmp	**Tot.**	**It.**	Pr.	Rmp	**Tot.**	**It.**
grid1	0.01	**0.01**	0	0	**0.01**	**8**	0	0	**0.01**	**10**	0	0.01	**0.02**	**28**
grid2	0.03	**0.02**	0.01	0	**0.02**	**10**	0	0	**0.02**	**13**	0.01	0.03	**0.08**	**84**
grid3	0.04	**0.04**	0	0.01	**0.02**	**8**	0.01	0	**0.02**	**10**	0.04	0.02	**0.09**	**44**
grid4	0.11	**0.11**	0.01	0	**0.04**	**11**	0.01	0.01	**0.05**	**13**	0.1	0.06	**0.38**	**100**
grid5	0.41	**0.41**	0.02	0.01	**0.14**	**14**	0.05	0.01	**0.15**	**15**	0.32	0.24	**1.18**	**132**
grid6	8.78	**10.37**	0.11	0.33	**0.71**	**20**	0.1	0.39	**0.81**	**26**	3.94	10.4	**20.14**	**751**
grid7	54.74	**55.96**	0.27	1.7	**2.54**	**14**	0.36	2.35	**3.28**	**18**	22.89	245.5	**299.73**	**1169**
grid8	1745.65	**1954.88**	0.88	23.2	**25.93**	**23**	1.25	36.72	**40.3**	**33**	TL	TL	**TL**	**TL**
grid9	TL	**TL**	1.52	76.69	**81.85**	**20**	2.47	166.72	**174.07**	**32**	TL	TL	**TL**	**TL**
grid10	TL	**TL**	2.75	93.18	**104.02**	**18**	3.65	190.97	**202.05**	**24**	TL	TL	**TL**	**TL**
grid12	TL	**TL**	7.28	56.31	**86.48**	**11**	8.59	107.32	**131.84**	**13**	TL	TL	**TL**	**TL**
grid14	TL	**TL**	26.39	135	**292.97**	**11**	28.73	227.1	**321.16**	**12**	TL	TL	**TL**	**TL**
planar30	0.03	**0.03**	0	0	**0.02**	**10**	0.01	0	**0.03**	**16**	0.05	0.03	**0.11**	**70**
planar50	0.49	**0.51**	0.04	0.04	**0.15**	**13**	0.07	0	**0.13**	**14**	0.36	0.07	**0.79**	**105**
planar80	10.33	**10.81**	0.18	0.7	**1.34**	**16**	0.22	0.4	**0.91**	**18**	11.73	5.55	**30.18**	**1071**
planar100	43.9	**45.62**	0.37	1.48	**2.88**	**13**	0.44	2.1	**3.15**	**15**	37.64	5.41	**82.22**	**1400**
planar150	4239.98	**4346.06**	1.81	26.87	**35.6**	**19**	2.19	33.76	**39.97**	**23**	TL	TL	**TL**	**TL**
planar300	TL	**5127.75**	7.56	16.9	**36.73**	**15**	7.36	20.04	**33.14**	**14**	1554.18	628.19	**3090.86**	**2967**
Averages	2117.36	**2064.03**	2.74	24.03	**37.31**	**14.12**	3.09	43.78	**52.84**	**17.73**	235.23	119.65	**293.82**	**660.09**

TABLE 2.13: Average times to optimality for Planar and Grid instances: direct approach and partially aggregated bundles with $\eta = 0.6, 0.8, 1$

2.4.3.5 PRT instances

The four "PRT" instances in this set derive from system traffic assignment problems for Personal Rapid Transit networks (see 3) . Linear MMCF PRT instances are obtained by fixing the otherwise flow-dependant cost of the arcs to a specific value. A generic instance is composed by $|K|-1$ single-origin single-destination commodities, representing full vehicle travels, and one multi-source multi-destination commodity representing the demands of empty vehicles. Table 2.14 summarizes the characteristics of the instances (see also Fig. 2.9 for a visual representation of the network topologies).

Instances	Network	Nodes	Arcs	Commodities	Mut.%
Koln307-65	Cologne	558	1039	308	4.33
Koln1578-10	Cologne	558	1039	1579	2.11
Portland1803-10	Portland	120	355	1804	5.07
Portland3583-5	Portland	120	355	3584	4.78

TABLE 2.14: The PRT instance set.

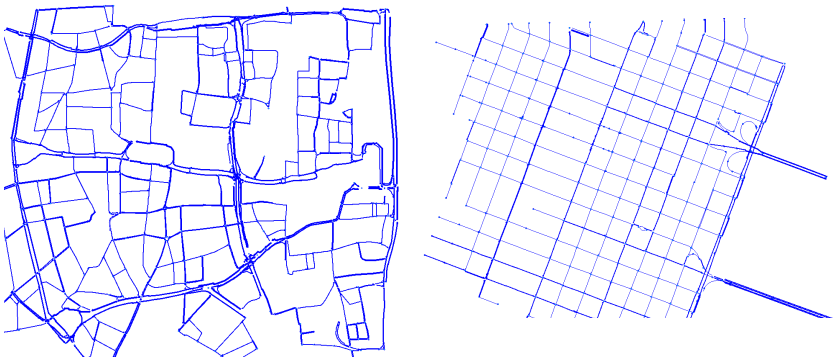

FIGURE 2.9: Geographical layout of Cologne and Portland networks, respectively on the left and on the right of the figure. Sources and destinations are randomly distributed among the nodes of the network. The lengths of the arcs reflects their costs.

Looking at tables 2.15, 2.16 and 2.17 we can see how for this set of instances bundle approaches are able to provide strong time reductions with respect to the direct approach. There is not a clear convenience behind full bundle disaggregation. In fact, the rather constant number of iterations measured for η going from -1 to 0.8 reflects into a quite significant reduction of reduced master problem net optimization time. Despite the very high number of iteration required to reach optimality, fully aggregated bundles are the most convenient choice if only master time is considered, . Anyway a large amount of the total net time is spent to solve the pricer. Such time increases together with the number of iterations, and a full aggregate bundle turns out to be not

the most convenient choice. A partial aggregation with $\eta = 0.8$ is instead very interesting, ensuring a number of iteration that is just slightly above the number obtained for $\eta = 0$.

	Time to solution							
			$\eta = 0$			$\eta = 0.2$		
Instance	CPX a.	**CPX t.**	Pr.	Rmp	**Tot.**	Pr.	Rmp	**Tot.**
Koln307-65	61.35	**48.755**	0.98	1.08	**3.6**	0.88	0.97	**2.91**
Koln1578-10	1054.78	**890.35**	3.61	2.64	**17.48**	3.61	2.39	**13.96**
Portland1803-10	173.564	**138.57**	0.58	1.49	**9.71**	0.49	1.16	**6.21**
Portland3583-5	431.53	**399.01**	1.09	3.18	**30.65**	1.08	3.26	**23.97**
Averages	430.31	**369.18**	1.57	2.1	**15.36**	1.52	1.95	**11.77**

TABLE 2.15: Average times to optimality for PRT instances: direct approach and partially aggregated bundles with $\eta = 0, 0.2$.

	Time to solution								
	$\eta = 0.5$			$\eta = 0.8$			$\eta = 1$		
Instance	Pr.	Rmp	**Tot.**	Pr.	Rmp	**Tot.**	Pr.	Rmp	**Tot.**
Koln307-65	1.01	1.05	**2.89**	1.04	0.96	**2.63**	26.4	2.82	**36.91**
Koln1578-10	3.59	1.95	**9.6**	4.11	2.07	**8.3**	49.95	0.82	**61.94**
Portland1803-10	0.56	0.88	**3.72**	0.58	0.61	**2.01**	8.19	0.28	**12.37**
Portland3583-5	1.04	2.18	**11.5**	1.25	1.57	**5.3**	14.39	0.19	**21.44**
Averages	1.55	1.52	**6.93**	1.75	1.31	**4.56**	24.74	1.03	**33.17**

TABLE 2.16: Average times to optimality for PRT instances: aggregated bundles with $\eta = 0.5, 0.8, 1$.

	Number of iterations				
Instance	$\eta = 0$	$\eta = 0.2$	$\eta = 0.5$	$\eta = 0.8$	$\eta = 1$
KOLN10-65	22	19	22	23	604
KOLN50-10	14	14	14	16	203
Portland500-10	11	10	11	12	173
Portland1000-5	11	11	11	13	154
Averages	14.5	13.5	14.5	16	283.5

TABLE 2.17: Computational results for PRT instances: average number of iterations for all the level of partial aggregation tested.

2.4.3.6 "Waxman" instances

Several MMCF instances has been obtained starting from "Waxman" network topologies (the name derives from [49]), created by using the Fast Network Simulation Setup (FNSS) toolchain available at `"http://fnss.github.io/"`. A number of 15 "Waxman-01" topologies (see FNSS documentation at `http://fnss.github.io/doc/core/index.html` for more details) has been generated in three sets that groups the networks by number of nodes; `alpha` parameter has always kept equal to 0.4, `beta` equal 0.1 and `L` to 1. Topologies of the same group differs only for the random seed given as input to the `waxman_1_topology` method. For each topology, the FNSS method `static_traffic_matrix`, is then invoked with `mean=0.8`, `stddev=0.05`, `max_u=0.9` to randomly generate single-origin single destination commodities. The number of commodities in the generated MMCF instance is very high, often close to to all the possible origin-destination pairs. Other instances are then created by randomly grouping single-origin single-destination commodities to form multi-origin multi-destination commodities. This procedure is completely unrelated to what mentioned before about the work of Jones et al. ([36]), hence the derived multi-origin multi-destination instances are not equivalent to the initial single-origin single destination instance. See Table 2.18 for more details about the instances.

Instances	Inst.s. #	Av. Comm.s #	Nodes	Av. Arcs	Av. Mut.%
WMOMDt_50_32	5	32	50	101.6	1.32
WMOMDt_50_128	4	128	50	98.5	1.96
WODt_50	5	1647.20	50	107.01	1.96
WMOMDt_100_128	3	128	100	400	1.96
WMOMDt_100_512	4	512	100	415	4.06
WMOMDt_100_2048	5	2048	100	410.4	4.49
WMOMDt_100_8192	3	8192	100	410.67	5.37
WODt_100	5	9624.40	100	410.93	4.1
WMOMDt_150_128	4	128	150	875	2
WMOMDt_150_512	4	512	150	875	0.93
WMOMDt_150_2048	4	2048	150	875	0.06
WMOMDt_150_8192	2	8192	150	897	0.79
WODt_150	5	22250.66	150	901.29	0.84

TABLE 2.18: The Waxman instance set.

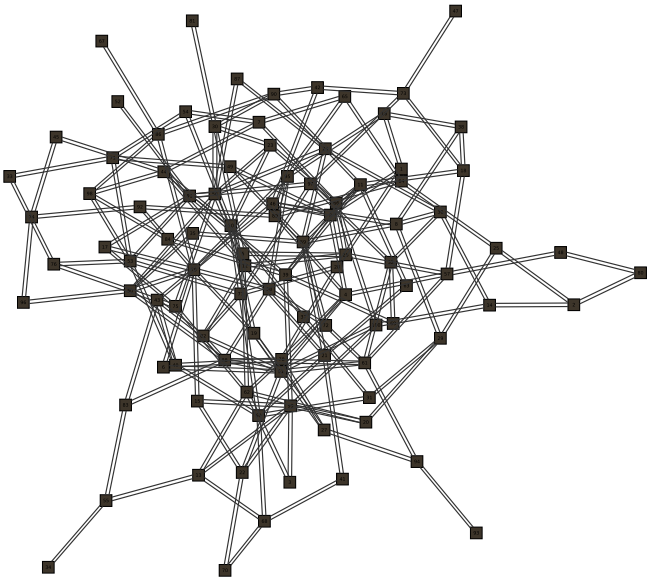

FIGURE 2.10: Automatically generated layout for a 'wMOMDt_100_128" instance.

Looking at Tables 2.19 and 2.20 we observe how the relatively small sizes of the graphs, combined with the high, or in some cases very high, number of commodities make this class of instances particularly adapt for aggregated bundles. We observe how the fully aggregated bundle ($\eta = 1$) always provides the best average performances for what concern the net reduced master problem time. In fact, the high number of iterations is well compensated by the presence of a smaller and easier reduced master problem. Nevertheless, as happening also for other classes of instances, the time needed for pricing the solution plays a central role. The choice of using a partial aggregation is often convenient considering the reduction of time spent for pricing. Especially in presence of a very high number of commodities, as for the instance WODt_150, partial aggregation with $\eta = 0.4$ or $\eta = 0.6$ gives very good balances between number of iterations, and consequently pricer overall times, and reduced master problem overall times.

	Average time to solution and number of iterations													
			$\eta = 0$				$\eta = 0.2$				$\eta = 0.4$			
Instance	CPX a.	**CPX t.**	Pr.	Rmp	**Tot.**	**It.**	Pr.	Rmp	**Tot.**	**It.**	Pr.	Rmp	**Tot.**	**It.**
WMOMDt_50_32	0.01	**0.01**	0	0	**0.01**	**1.6**	0.01	0	**0.01**	**1.6**	0	0	**0**	**1.6**
WMOMDt_50_128	0.04	**0.04**	0.01	0	**0.02**	**2.5**	0.01	0	**0.02**	**2.5**	0.01	0.01	**0.02**	**2.5**
WODt_50	1.13	**1.16**	0.02	0.11	**0.42**	**2.6**	0.02	0.08	**0.3**	**2.6**	0.02	0.04	**0.18**	**2.6**
WMOMDt_100_128	0.36	**0.37**	0.02	0.02	**0.06**	**3**	0.03	0.01	**0.06**	**3**	0.03	0	**0.06**	**3**
WMOMDt_100_512	2.54	**2.22**	0.09	0.05	**0.26**	**3**	0.08	0.04	**0.24**	**3**	0.08	0.03	**0.2**	**3**
WMOMDt_100_2048	9.29	**9.46**	0.23	0.29	**1.6**	**3**	0.24	0.26	**1.26**	**3**	0.25	0.22	**0.96**	**3**
WMOMDt_100_8192	28.82	**32.01**	0.34	0.25	**4.09**	**1.67**	0.35	0.26	**2.96**	**1.67**	0.37	0.28	**2.13**	**1.67**
WODt_100	31.12	**31.81**	0.31	0.31	**13**	**2.6**	0.32	0.32	**8.75**	**2.6**	0.33	0.31	**5.5**	**2.6**
WMOMDt_150_128	0.91	**0.93**	0.04	0.01	**0.06**	**1.25**	0.04	0.01	**0.06**	**1.25**	0.04	0.01	**0.06**	**1.25**
WMOMDt_150_512	4.84	**5.17**	0.14	0.07	**0.32**	**2**	0.15	0.08	**0.31**	**2**	0.14	0.05	**0.28**	**2**
WMOMDt_150_2048	18.3	**19.03**	0.44	0.37	**1.74**	**2.5**	0.46	0.36	**1.56**	**2.5**	0.45	0.31	**1.28**	**2.5**
WMOMDt_150_8192	89.3	**101.88**	2.13	0.78	**30.31**	**5**	2.17	0.82	**22.5**	**5**	2.1	0.76	**12.44**	**4.5**
WODt_150	136.74	**187.66**	1.16	0.52	**47.62**	**2.4**	1.21	0.5	**32.13**	**2.4**	1.21	0.44	**19.51**	**2.4**
Averages	23.85	**29.45**	0.32	0.2	**7.47**	**2.28**	0.33	0.2	**5.19**	**2.28**	0.33	0.17	**3.2**	**2.27**

TABLE 2.19: Average times to optimality for Waxman instances: direct approach and partially aggregated bundles with $\eta = 0, 0.2, 0.4$

	Average time to solution and number of iterations													
			$\eta = 0.6$				$\eta = 0.8$				$\eta = 0.1$			
Instance	CPX a.	**CPX t.**	Pr.	Rmp	**Tot.**	**It.**	Pr.	Rmp	**Tot.**	**It.**	Pr.	Rmp	**Tot.**	**It.**
WMOMDt_50_32	0.01	**0.01**	0	0	**0.01**	**1.6**	0	0	**0.01**	**1.6**	0.01	0	**0.01**	**2**
WMOMDt_50_128	0.04	**0.04**	0.01	0	**0.02**	**2.5**	0.01	0	**0.02**	**2.75**	0.02	0.01	**0.03**	**6**
WODt_50	1.13	**1.16**	0.02	0.01	**0.1**	**2.6**	0.02	0.01	**0.06**	**2.6**	0.07	0.01	**0.16**	**9**
WMOMDt_100_128	0.36	**0.37**	0.04	0.01	**0.07**	**3.34**	0.03	0.01	**0.06**	**3.67**	0.07	0.01	**0.14**	**13.34**
WMOMDt_100_512	2.54	**2.22**	0.09	0.02	**0.17**	**2.75**	0.08	0.02	**0.16**	**2.75**	0.28	0.01	**0.45**	**12.25**
WMOMDt_100_2048	9.29	**9.46**	0.25	0.12	**0.69**	**3**	0.25	0.07	**0.52**	**3.2**	0.69	0.01	**1.15**	**9.8**
WMOMDt_100_8192	28.82	**32.01**	0.35	0.29	**1.45**	**1.67**	0.36	0.23	**1**	**1.67**	0.99	0.01	**1.94**	**5**
WODt_100	31.12	**31.81**	0.32	0.31	**3.05**	**2.6**	0.32	0.26	**1.52**	**2.6**	5.41	0.08	**14.6**	**39**
WMOMDt_150_128	0.91	**0.93**	0.04	0.01	**0.06**	**1.25**	0.04	0	**0.06**	**1.25**	0.04	0	**0.05**	**1.5**
WMOMDt_150_512	4.84	**5.17**	0.15	0.04	**0.26**	**2**	0.16	0.03	**0.28**	**2.25**	0.18	0.01	**0.25**	**2.75**
WMOMDt_150_2048	18.3	**19.03**	0.46	0.25	**1.11**	**2.5**	0.48	0.15	**0.95**	**2.5**	0.59	0.01	**0.93**	**3.25**
WMOMDt_150_8192	89.3	**101.88**	2.24	0.72	**9.31**	**5**	2.21	0.68	**5.85**	**5**	7.7	0.02	**15.05**	**16.5**
WODt_150	136.74	**187.66**	1.31	0.43	**12.4**	**2.6**	1.39	0.43	**6.24**	**2.6**	17.36	0.08	**54.79**	**34.6**
Averages	23.85	**29.45**	0.34	0.15	**2.09**	**2.3**	0.35	0.13	**1.18**	**2.37**	2.66	0.02	**7.49**	**11.04**

TABLE 2.20: Average times to optimality for Waxman instances: direct approach and partially aggregated bundles with $\eta = 0.6, 0.8, 1$

2.5 Conclusions and Future Works

It is a common strategy to reduce the overall computing time of an iterative approach based on decomposition by finding a good compromise between convergence speed and time for single iteration. The attention is generally focussed on the reduced master problem, while the pricer is often an easy optimization problem efficiently solved by existing algorithms. This is a valid strategy for the MMCF, in which the pricer can be decomposed among easily solvable single-commodity flow problems. Conversly, *long-tail effects* and slow convergence issues typically affects column generation for this class of problem. The inclusion of stabilization tools generally aim to strongly reduce the number of iteration without compromising the ability of efficiently solving the reduced master problem. Similarly disaggregate formulation corresponds to larger and more difficult reduced master problem but allows the construction of a better model for $f(x)$. This in general translates to a lower number of cutting planes to reach optimality.

Partial aggregation opens new alternatives in this contest. Acknowledged the possibility of formulating the reduced master using any level of aggregation without changing de definition of the (decomposable) pricer, a wide range of gradual alternative between complete disaggregation and full aggregation can be explored. A combined use of partial aggregation and stabilization tools provide a vast spectrum of trade-offs between convergence speed and time for iteration.

We explored only part of the possible alternatives allowed by the definition of partially aggregated bundles. The results shows how a correct choice of the formulation for the reduced master problem is crucial in most of the cases. Often, the choice of disaggregating the bundle is the best choice. A full disaggregation is not anyway the best choice in all the situation. In some cases partial aggregation reduces the overall execution time with respect to the two extreme levels of aggregation ($MCCF - MP_{-1}$) and ($MCCF - MP_1$). Boxstep stabilization is used without fine parameter tuning. More accurate stabilization strategies may change the presented scenario, more likely in favour of aggregated formulation.

We thus feel that further studies are necessary to fully understand the potential of this new alternatives. In particular, more sophisticated stabilization methods, or even a better parameter tuning for the boxstep tool used here, may have a big impact on the performances of the decomposition approaches proposed. Parameter tuning for the direct approach allowed a marked reduction of the computing time. We fell that parameter tuning can likewise improve the ability of solving the reduced master problem and consequently the overall performances of decomposition approaches. The possibility of changing the solver itself should be taken into account as well. Besides, the possibility of dynamically aggregate components of the bundle on the basis of information collected during the iterative process may open new interesting opportunities.

Finally, the test bed could be further expanded to give a complete overview of all the possible situations in which practitioner may want to use the approaches proposed.

Bibliography

[1] Ravindra K Ahuja, Thomas L Magnanti, and James B Orlin. Network flows: theory, algorithms, and applications. 1993.

[2] A. Astorino, A. Frangioni, M. Gaudioso, and E. Gorgone. Piecewise Quadratic Approximations in Convex Numerical Optimization. *SIAM Journal on Optimization*, 21(4):1418–1438, 2011.

[3] Frédéric Babonneau, Olivier Du Merle, and J-P Vial. Solving large-scale linear multicommodity flow problems with an active set strategy and proximal-accpm. *Operations Research*, 54(1):184–197, 2006.

[4] L. Bacaud, C. Lemaréchal, A. Renaud, and C. Sagastizábal. Bundle Methods in Stochastic Optimal Power Management: A Disaggregated Approach Using Preconditioners. *Computational Optimization and Applications*, 20:227–244, 2001.

[5] Martin Bergner, Alberto Caprara, Fabio Furini, Marco E Lübbecke, Enrico Malaguti, and Emiliano Traversi. Partial convexification of general mips by dantzig-wolfe reformulation. In *Integer Programming and Combinatoral Optimization*, pages 39–51. Springer, 2011.

[6] A. Borghetti, A. Frangioni, F. Lacalandra, and C.A. Nucci. Lagrangian Heuristics Based on Disaggregated Bundle Methods for Hydrothermal Unit Commitment. *IEEE Transactions on Power Systems*, 18(1):313–323, February 2003.

[7] P. Cappanera and A. Frangioni. Symmetric and Asymmetric Parallelization of a Cost-Decomposition Algorithm for Multi-Commodity Flow Problems. *INFORMS Journal on Computing*, 15(4):369–384, 2003.

[8] Alberto Caprara, Fabio Furini, and Enrico Malaguti. Uncommon dantzig-wolfe reformulation for the temporal knapsack problem. *INFORMS Journal on Computing*, 25(3):560–571, 2013.

[9] Jordi Castro. A specialized interior-point algorithm for multicommodity network flows. *SIAM journal on Optimization*, 10(3):852–877, 2000.

[10] Jordi Castro. Solving difficult multicommodity problems with a specialized interior-point algorithm. *Annals of Operations Research*, 124(1-4):35–48, 2003.

[11] Jordi Castro and Narcis Nabona. An implementation of linear and nonlinear multicommodity network flows. *European Journal of Operational Research*, 92(1):37–53, 1996.

[12] T.G. Crainic, A. Frangioni, and B. Gendron. Bundle-based Relaxation Methods for Multicommodity Capacitated Fixed Charge Network Design Problems. *Discrete Applied Mathematics*, 112:73–99, 2001.

[13] George B Dantzig, Alex Orden, and Philip Wolfe. The generalized simplex method for minimizing a linear form under linear inequality restraints. *Pacific Journal of Mathematics*, 5(2):183–195, 1955.

[14] George B Dantzig and Philip Wolfe. Decomposition principle for linear programs. *Operations Research*, pages 101–111, 1960.

[15] George B Dantzig and Philip Wolfe. The decomposition algorithm for linear programs. *Econometrica: Journal of the Econometric Society*, pages 767–778, 1961.

[16] O. du Merle, J.-L. Goffin, and J.-P. Vial. On Improvements to the Analytic Center Cutting Plane Method. *Computational Optimization and Applications*, 11(1):37–52, 1998.

[17] Jr. Ford, L. R. and D. R. Fulkerson. A suggested computation for maximal multicommodity network flows. *Management Science*, 5(1):pp. 97–101, 1958.

[18] LR Ford Jr, DR Fulkerson, and Arthur Ziffer. Flows in networks. *Physics Today*, 16:54, 1963.

[19] A. Frangioni. *Dual-Ascent Methods and Multicommodity Flow Problems.* PhD thesis, TD 5/97, Dipartimento di Informatica, Università di Pisa, Pisa, Italy, 1997.

[20] A. Frangioni. Generalized Bundle Methods. *SIAM Journal on Optimization*, 13(1):117–156, 2002.

[21] A. Frangioni and G. Gallo. A Bundle Type Dual-Ascent Approach to Linear Multicommodity Min Cost Flow Problems. *INFORMS Journal on Computing*, 11(4):370–393, 1999.

[22] A. Frangioni and B. Gendron. A Stabilized Structured Dantzig-Wolfe Decomposition Method. *Mathematical Programming*, to appear, 2013.

[23] A. Frangioni and C. Gentile. Experiments with a Hybrid Interior Point/Combinatorial Approach for Network Flow Problems. *Optimization Methods and Software*, 22(4):573 – 585, 2007.

[24] A. Frangioni and E. Gorgone. Generalized Bundle Methods for Sum-Functions with "Easy" Components: Applications to Multicommodity Network Design. *Mathematical Programming*, to appear, 2014.

[25] A. Frangioni, A. Lodi, and G. Rinaldi. New Approaches for Optimizing Over the Semimetric Polytope. *Mathematical Programming*, 104(2-3):375–388, 2005.

[26] A. Frangioni and A. Manca. A Computational Study of Cost Reoptimization for Min Cost Flow Problems. *INFORMS Journal on Computing*, 18(1):61–70, 2006.

[27] Matthew Galati. *Decomposition methods for integer linear programming.* Lehigh University, 2010.

[28] Gerald Gamrath and Marco E Lübbecke. Experiments with a generic dantzig-wolfe decomposition for integer programs. In *Experimental algorithms*, pages 239–252. Springer, 2010.

[29] Arthur M Geoffrion. Generalized benders decomposition. *Journal of optimization theory and applications*, 10(4):237–260, 1972.

[30] J.-L. Goffin and J.-P. Vial. Convex Nondifferentiable Optimization: a Survey Focussed on the Analytic Center Cutting Plane Method. *Optimization Methods and Software*, 17(5):805–867, 2002.

[31] Andrew V Goldberg and Michael Kharitonov. On implementing scaling push-relabel algorithms for the minimum-cost flow problem. *Network Flows and Matching: First DIMACS Implementation Challenge*, 12:157–198, 1993.

[32] Donald Goldfarb and Michael D Grigoriadis. A computational comparison of the dinic and network simplex methods for maximum flow. *Annals of Operations Research*, 13(1):81–123, 1988.

[33] J. Gondzio, P. González-Brevis, and P. Munari. New Developments in the Primal-dual Column Generation Technique. Technical Report ERGO-11-001, School of Mathematics, The University of Edinburgh, 2011.

[34] J.-B. Hiriart-Urruty and C. Lemaréchal. *Convex Analysis and Minimization Algorithms* I—*Fundamentals*, volume 306 of *Grundlehren Math. Wiss.* Springer-Verlag, New York, 1993.

[35] TC Hu. Multi-commodity network flows. *Operations Research*, 11(3):344–360, 1963.

[36] K.L. Jones, I.J. Lustig, J.M. Farwolden, and W.B. Powell. Multicommodity Network Flows: The Impact of Formulation on Decomposition. *Mathematical Programming*, 62:95–117, 1993.

[37] Jeff L Kennington. A survey of linear cost multicommodity network flows. *Operations Research*, 26(2):209–236, 1978.

[38] K.C. Kiwiel. A Bundle Bregman Proximal Method for Convex Nondifferentiable Optimization. *Mathematical Programming*, 85:241–258, 1999.

[39] Torbjörn Larsson and Di Yuan. An augmented lagrangian algorithm for large scale multicommodity routing. *Computational Optimization and Applications*, 27(2):187–215, 2004.

[40] C. Lemaréchal, A. Nemirovskii, and Y. Nesterov. New variants of bundle methods. *Mathematical Programming*, 69:111–147, 1995.

[41] C. Lemaréchal, A. Ouorou, and G. Petrou. A Bundle-type Algorithm for Routing in Telecommunication Data Networks. *Computational Optimization and Applications*, 44(3):385–409, 2009.

[42] Claude Lemaréchal and Robert Mifflin. *Nonsmooth optimization*, volume 3. Pergamon Press Oxford, UK, 1978.

[43] John W Mamer and Richard D McBride. A decomposition-based pricing procedure for large-scale linear programs: an application to the linear multicommodity flow problem. *Management Science*, 46(5):693–709, 2000.

[44] R. E. Marsten, W. W. Hogan, and J. W. Blankenship. The boxstep method for large-scale optimization. *Operations Research*, 23(3):pp. 389–405, 1975.

[45] S THOMAS MCCORMICK and LI LIU. An experimental implementation of the dual cancel and tighten algorithm. *Network Flows and Matching: First DIMACS Implementation Challenge*, 12:247, 1993.

[46] A. Ouorou. A Proximal Cutting Plane Method Using Chebychev Center for Nonsmooth Convex Optimization. *Mathematical Programming*, 119(2):239–271, 2009.

[47] A. Ouorou. The Proximal Chebychev Center Cutting Plane Algorithm for Convex Additive Functions. *Mathematical Programming*, to appear, 2013.

[48] JA Tomlin. Minimum-cost multicommodity network flows. *Operations Research*, 14(1):45–51, 1966.

[49] Bernard M Waxman. Routing of multipoint connections. *Selected Areas in Communications, IEEE Journal on*, 6(9):1617–1622, 1988.

[50] Roman L Weil and Paul C Kettler. Rearranging matrices to block-angular form for decomposition (and other) algorithms. *Management Science*, 18(1):98–108, 1971.

Chapter 3

Solving the optimum system traffic assignment problem for Personal Rapid Transit networks using the Frank-Wolfe algorithm[1]

3.1 Introduction

Personal Rapid Transit (PRT) is a novel form of fully automated, guided, public transportation. With PRT, up to 6 persons travel in small, individually controlled and electrically driven vehicles on an exclusive network of narrow guideways. Vehicles are available at stations on-demand and trips can be booked individually, similar to a taxi.

Stations are usually off-line as to allow a non-stop trip between origin and destination. On-vehicle switching and globally-controlled routing allows such systems to operate as true, seamless networks, without the need to transfer. PRT may provide a genuine alternative to the private car as individual passengers or small groups can travel together by choice in a private atmosphere and in seated comfort.

From the user's perspective, PRT is accessible at dedicated stations. Destination selection and ticket purchase is done at stations (or beforehand through the Internet). At the station, either vehicles are already available or the system routes empty vehicles to a station, on demand. Once a vehicle becomes available, the passenger enters and reaches his/her destination non-stop. As soon as the vehicle is cleared at the destination, it will be re-used if passengers are present; otherwise, empty vehicles are routed to stations with passengers or to a nearby depot.

[1] **Technical Report** "*Solving the optimum system traffic assignment problem for Personal Rapid Transit networks using the Frank-Wolfe algorithm*", J. Schweizer, T. Parriani, E. Traversi, F. Rupi, *Tech. Rep. DEI, OR-13-17.*

Even though the concepts of PRT have been known since the 60's, it took until 2010 before the first two small-scale PRT systems became operational: the 2getthere PRT in Masdar, Abu Dhabi, UEA and the ULTra PRT at terminal 5 at Heathrow airport, London, UK. In Suncheon Bay, south Korea, Vectus-PRT is expected to open a 40km long track by fall 2013.

The analyses of PRT networks have been mainly performed by micro-simulators which mimic the movement of individual vehicles and users [8, 18, 17]. Also conventional static traffic assignment methods have been used in studies, such as the all-or-nothing assignment, see [7] for an overview of traffic assignment methods.

However, the traffic assignment for PRT must take into account the specific characteristics of a PRT system: (i) PRT systems require an *occupied vehicle routing* and an *empty vehicle routing*. The occupied vehicle management (OVM) routes the vehicles from origin to destination along the fastest route and the empty vehicle management (EVM) counterbalances the full vehicle flows while minimizing trip times. (ii) As both *vehicle managements are centralized* it is legitimate to assume a *system optimal* solution for the vehicle routing in case the managements have a priori information of the transport demand and the demand is uniformly distributed over a considered time interval. This means, the total travel time of all trips of both, empty and full vehicles, can be minimized. (iii) In general, the travel *time over a transport link depends on the vehicle flow*, similar to road traffic. If the vehicle flow on a link is far below the capacity limit then vehicles do not interact and run at the line-speed limit. But approaching the capacity limit, the PRT vehicles will try to maintain a *minimum safe distance*, that is the distance necessary to stop without colliding with the vehicle in front, which is braking at a predefined maximum failure deceleration rate.

Note that the proposed traffic assignment will find the optimal traffic flows of a PRT network by minimizing the total travel time, while taking into account the reuse of all vehicles (no parking space required) and the flow-dependent, physically minimum trip time on each link.

The assumption that the entire transport demand is a priori known and uniformly distributed is idealistic because neither the destination nor the exact departure time is a priori known. In a running PRT system, the routing is performed in real time, when the passenger decided to travel and communicates its destination to the system. However, for planning purposes and in particular during the network design process, the static traffic assignment is a useful tool as it quickly identifies bottlenecks and estimates the minimum number of required vehicles, which is a significant costs factor. Once the network topology is identified the performance needs to be verified by micro-simulations. It has been shown that random arrival times and real-time routing of a microscopic simulation increase the empty vehicle flows with respect to the static traffic assignment [19].

In literature a large number of deterministic and stochastic assignment methods is known that could be adapted to assign the traffic of full and empty vehicles to a PRT network. As for instance the probabilistic assignment methods described in [15, 16] model the temporal and spacial occupation of resources as Markovian random processes. In theory, this framework would be ideally suited for a transport network that is "driven" by a random demand. But there are very high dimensions and convergence problems for larger networks, in particular when loaded at capacity limits, see discussion in [19]. The optimization model by [17] has been developed to determine the total minimum static empty vehicle flows on a PRT network for a given constant demand. The authors were interested in determining the maximum transport demand that a PRT network can absorb with a limited number of vehicles. But the flows and capacity limits on single links have not been taken into consideration, see [19].

The PRT traffic assignment model used in this chapter has been formulated in [19, 13]. It is a model that minimizes total travel time for empty and occupied vehicle trips and is fast to resolve in case of uncongested networks, using constant trip times (or fixed link costs) for all network links. In order to verify the feasibility of the results, the assigned link flows can be compared against the maximum possible flow at a fixed line velocity. However, this assignment methods does not take into account that physically possible flow on a link can vary with the velocity.

In the present work we propose an extension to the assignment problem described in [19] for the case of a congested network, where vehicles try to maintain a minimum safe distance: a central control can route more vehicles through particular links in order to shorten the path for many origin-destination-pairs, but at the cost of decreasing vehicle speed and increases travel time due to congestion effects. We solve this trade-off by using the so called "system optimal" assignment method, following the second principle of Wardop [1]. While this assignment method has been used to calculate optimal tolls for road networks, here the link-costs (i.e travel time) are solely determined by the maximum line-speeds and flow-dependent line speeds of the network links. In contrast with the toll system for road vehicles, where link-flows can be reduced by increasing their respective monetary costs [3, 4, 5, 6], in a fully automated system the flows can be imposed by the centralized routing. The changing link costs, due to a change in flow-dependent link velocities, is therefore a consequence, and not the cause.

We show that this approach is indeed suitable to minimize the total travel time of all passengers by exploiting the physical capacities limits of the network. The resulting linear programming (LP) problem can then be solved with the Frank-Wolfe algorithm [2] as suggested in [7].

In the next section we derive the cost function of a PRT link where vehicles run at a minimum safe distance, define the linear programming model (LP model) of the assignment and present the Frank Wolfe algorithm used to solve the specific LP problem.

Section 3.3 shows the results of traffic assignments to different networks and in particulate demonstrates the convergence speed of the proposed algorithm . In Sec. 3.4 we summarize the main findings of this work and comment the advantages and limits of the proposed methods.

3.2 Methodology

In this section we develop first the flow dependent function of the link travel times before presenting the LP model and the modified Frank Wolfe algorithm, including the prove of convergence.

3.2.1 Link costs with minimum safe distance spacing

If the automatically controlled vehicles are able to maintain a minimum safe distance in all situations then the vehicle flows on a link can reach the physical link capacity. Here we will derive the link cost, which is the time that it takes a vehicle to cross a link while maintaining a minimum safe distance to the vehicle in front. As intermediate step we determine first the speed as a function of the link flow. Thereafter we calculate the exact travel time and finally we approximate it with a piecewise linear cost function which will be used in the traffic assignment model. Two successive vehicles keep a minimum safe distance if the follower vehicle can stop without collision, even if the preceding vehicle stops instantly.

This condition is often referred to as the "brick wall stopping criteria". Assuming a break actuation delay time τ, an emergency brake deceleration a_E and a vehicle of length L running at speed v, then this condition is satisfied if the nose-to-nose time headway is greater than $\tau + \frac{v}{2a_E} + \frac{L}{v}$. Given this minimum time interval, the speed dependent vehicle flow $f(v)$ in vehicles per second becomes

$$f(v) = \frac{1}{\frac{L}{v} + \frac{v}{2a} + \tau}. \tag{3.1}$$

The link-capacity q is the maximum flow at the critical speed $v = v_{\text{crit}} = \sqrt{2a_E L}$, hence

$$q = f(v_{\text{crit}}) = \frac{1}{\tau + \sqrt{\frac{2L}{a_E}}}.$$

The lower bound of the vehicle speed is $v > v_{\text{crit}}$, because lower velocities would cause instabilities in the traffic flow which can lead to still stands. Solving Eq. 3.1 for $v > v_{\text{crit}}$

we obtain

$$v(f) = \frac{1}{f}\left(\sqrt{-2a_E\, f^2\, L + a_E^2 f^2\, \tau^2 - 2a_E^2 f\, \tau + a_E^2} - a_E f \tau + a_E\right). \tag{3.2}$$

The upper velocity bound is given by the maximum line speed V_a allowed on link a. Finally the link cost $\hat{c}_a(f_a)$ to cross a link of length l_a with a uniform maximum velocity V_a and a link flow of f_a is given by

$$\hat{c}_a(f_a) = \begin{cases} \dfrac{l_a}{V_a} & \text{for} \quad f_a \leq f(V_a) \\ \dfrac{l_a}{v(f_a)} & \text{for} \quad f_a > f(V_a) \end{cases} \tag{3.3}$$

with $f(\cdot)$ from Eq. 3.1 and $v(\cdot)$ from Eq. 3.2. Note that the maximum velocity on a link is not necessarily uniform, for example in presence of sharp curves. Such details can be taken into consideration by using an average link velocity or by splitting the link by inserting additional nodes. An example cost function with realistic vehicle- and link parameters is shown in Fig. 3.1.

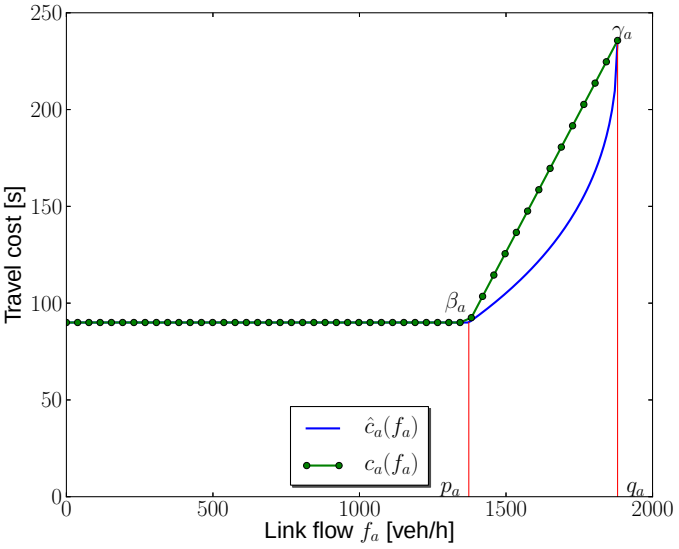

FIGURE 3.1: Exact link costs $\hat{c}_a(f_a)$ and piecewise linear link costs $c_a(f_a)$ versus link flow f_a for link a of length $l_a = 1000m$ and a maximum line-velocity of $V_a = 40km/h$. The vehicle parameters are: $\tau = 0.50s$, $a_E = 3.00m/s^2$, $L = 3m$ the critical speed $v_{\text{crit}} = 15.27km/h$, the capacity $q = 1880veh/h$. For this set of parameters the constants characterizing the piecewise linear function are $p_a = 1373veh/h$, $q_a = 1880veh/h$, $\beta_a = 90s$ and $\gamma_a = 235s$.

The piecewise-linear approximation $c_a(f_a)$ of the cost function $\hat{c}_a(f_a)$ can be constructed by using the vehicle flow at line speed as break point p_a and the capacity

q_a as upper flow limit. With $p_a = f(V_a)$, $q_a = f(v_{\text{crit}})$, $\beta_a = l_a/V_a$, $\gamma_a = l_a/v_{\text{crit}}$ the bilinear cost function can be defined as

$$c_a(f) = \begin{cases} \beta_a & \text{for} \quad f_a \leq p_a \\ \beta_a + \dfrac{\gamma_a - \beta_a}{q_a - p_a}(f_a - p_a) & \text{for} \quad f_a > p_a \end{cases} \tag{3.4}$$

See again Fig. 3.1 for comparing the approximation $c_a(f_a)$ with exact solution $\hat{c}_a(f_a)$. For sake of conciseness, we introduce inclination ε_a of the congested part:

$$\varepsilon_a = \frac{\gamma_a - \beta_a}{q_a - p_a}\,.$$

In reality, if vehicle flows exceed q_a then the traffic would collapse into a traffic jam, corresponding to a cost that tends to infinity. However, the bi-linear function in Eq. 3.4 would result in a proportional increase of travel time.

3.2.2 Bilinear programming model

Here we recall the system optimum bilinear programming model as reported in [19]. As previously mentioned, the vehicle flows on the links are found by minimizing the trip time of all passengers, taking into account the empty- and occupied vehicle flows as well as the link costs derived in Sec. 3.2.1. The proposed mathematical programming model for the the centralized occupied- and empty vehicle routing must satisfy the following conditions:

- The full vehicles flows must be assigned such that a given demand between each origin-destination pair of the network is satisfied.
- The full vehicle flow must be counterbalanced by an empty vehicle flow.

For the latter we add in each station vehicles-flow conservation constraints by defining a fictitious demand of empty vehicles for each station. This fictions demand represents the algebraic difference between the number of exiting and entering full vehicles at each station, modeled as network nodes with demand. Then we ensure the flow-equilibrium by adding a multi-origin/multi-destination empty vehicle flow to the model.

The transport problem is defined as follows: Let $G = (V, A)$ be the directed network graph where V and A are the sets of PRT network nodes and links, respectively. Each link $a = (i, j) \in A$ is associated with the bilinear travel cost function c_a. Let $S \subset V$ be the sub-set of nodes associated with stations and let $R \subseteq V \times V$ be a set

of routes $r = (s_r, t_r) \in R$, each represented by the pairs of station nodes. The total travel demand within the observation period are D trips and d_r is called the fractional demand, representing the number of passengers traveling from origin s_r to destination t_r along route r, such that $D = \sum_{r \in R} d_r$.

We now define D_i^{res} as the residual demand of node i defined as follows:

$$D_i^{res} = \sum_{r \in R: s_r = i} d_r - \sum_{r \in R: t_r = i} d_r, \forall i \in V \tag{3.5}$$

where $D_i^{res} > (<) 0$ means that there is a demand (surplus) of vehicles in node i. Let us further introduce the variable y_a^r, representing the fractional part of the flow on route r using link a. The occupied vehicle flow on link a is the sum of all fractional flows y_a^r multiplied by the route demand d_r. The total vehicle flow f_a on link a is the sum of of the occupied vehicle flow and the empty vehicle flow w_a, thus

$$f_a = \sum_{r \in R} (d_r y_a^r) + w_a \ . \tag{3.6}$$

It can be shown that the number of vehicles N on the network which is necessary to serve a particular demand D is determined by

$$N = \sum_{a \in A} c_a(f_a) f_a \ .$$

The objective function used in the system-optimum (SO) bilinear programming model represents the total costs, the travel time of all full and empty vehicles:

$$\min \textstyle\sum_{a \in A} c_a f_a \tag{3.7}$$

$$\textstyle\sum_{a \in \delta^+(i)} y_a^r - \sum_{a \in \delta^-(i)} y_a^r = \begin{cases} 1 & \text{if } i = s_r \\ -1 & \text{if } i = t_r \\ 0 & \text{otherwise} \end{cases}, \quad \forall i \in V, \forall r \in R \tag{3.8}$$

$$\textstyle\sum_{a \in \delta^+(i)} w_a - \sum_{a \in \delta^-(i)} w_a = D_i^{res}, \qquad \forall i \in V \tag{3.9}$$

$$\textstyle\sum_{r \in R} d^r y_a^r + w_a = f_a, \qquad \forall a \in A, \tag{3.10}$$

$$c_a \geq \beta_a + \varepsilon_a (f_a - p_a), \qquad \forall a \in A \tag{3.11}$$

$$c_a \geq \beta_a, \qquad \forall a \in A \tag{3.12}$$

$$c_a \leq \gamma_a, \qquad \forall a \in A \tag{3.13}$$

$$w_a, c_a, f_a \geq 0 \qquad \forall a \in A \tag{3.14}$$

$$y_a^r \geq 0, \qquad \forall a \in A, \forall r \in R \tag{3.15}$$

The constraints (3.8) (resp. (3.9)) guarantee flow conservation of full (rep. empty) vehicles. Constraints (3.11) and (3.12) are used to describe the function (3.4) and constraints (3.13) fix the upper bound for c_a, as derived in Sec. 3.2.1.

This bilinear representation is typical in operational research and it is useful to develop Linear Programming based Heuristic methods. But the problem can be rewritten and solved via the reduced gradient method of Frank and Wolfe as shown below.

3.2.3 Solution method: Frank Wolfe algorithm

The (continuous) bilinear problem (3.7)-(3.15) can be solved directly by a generic solver for non linear continuous problems, like IPOPT[23]. Preliminary tests showed that our method is drastically better and allows to quickly solve the instances studied. In order to solve them to optimality we exploit the following property:

Observation 1. Problem (3.7)-(3.15) is a convex problem (i.e. the objective function is convex over the feasible region).

Proof. the statement can be proven by showing that every component $c_a f_a$ is convex over the feasible region defined by the constraints (3.8)-(3.15).
Each term c_a is the maximum value among the two values given by β_a and $\beta_a + \varepsilon_a(f_a - p_a)$. Because f_a is non-negative, we can rewrite

$$c_a f_a = \max\{\beta_a f_a, \varepsilon_a f_a^2 + (\beta_a - \varepsilon_a p_a) f_a\} \ .$$

The first term is linear and hence convex. Also the second term is convex because ε_a is non-negative. This allows to conclude that also $c_a f_a$ is convex, because it is the maximum of two convex functions. □ □

Observation 1 allows us to use a Frank-Wolfe (FW) algorithm [2] for solving the above stated SO assignment problem: For each iteration n of the FW algorithm the flows f_a^n and costs c_a^n are determined. As $n \to \infty$ the flows f_a^n converge to the optimal solution, minimizing (3.7), while satisfying the constraints (3.8)-(3.15). The initialization (step 0) and iteration n (steps 1-4) are as follows:

Step 0: Set iteration counter $n = 0$. Set upper bound $UB = +\infty$ and scale parameter $\lambda_0 = +\infty$.

Compute an initial feasible solution (f_a^0, c_a^0) which can be done efficiently by fixing $c_a^0 = \beta_a$ and solving the linear problem (3.7)-(3.10), (3.14)-(3.15). This is equivalent to performing an all-or-nothing assignment, assuming an uncongested network (all flows $f_a < p_a$).

Step 1: Updating the best solution obtained so far: $UB = \sum_{a \in A} c_a^n f_a^n$.

Compute the first order approximation of the objective function in (c_a^n, f_a^n), see note [2], and if it is equal to zero then STOP.

[2] Note that the first order approximation of the function $\sum_{a \in A} c_a f_a$ in the point (c_a^*, f_a^*) corresponds to $\sum_{a \in A} c_a^* f_a^* + \sum_{a \in A} (c_a^* f_a + f_a^* c_a)$.

If $\lambda_n = 0$ then STOP.
Set $n = n + 1$.

Step 2: Use the first order approximation of the objective function of the previous iteration f_a^{n-1} and c_a^{n-1} as objective function of the following approximated problem:

$$\begin{aligned} &\min \textstyle\sum_{a \in A} c_a^{n-1} f_a + f_a^{n-1} c_a \\ &(3.8) - (3.15) \ . \end{aligned} \tag{3.16}$$

Note that problem (3.16),(3.8)-(3.15) is a multi-commodity flow problem and hence can be solve with ad-hoc algorithm, in addition to using a generic LP solver. At this step the optimal solution to problem (3.16), (3.8)-(3.15) gives also a valid lower bound to the optimal solution. Let LB be this bound, at each iteration we can hence compute the open gap remained:

$$Gap = 100 \times \frac{UB - LB}{UB} \ .$$

Step 3: Let $(\overline{c}_a^n \overline{f}_a^n)$ be the optimal solution to problem (3.16),(3.8)-(3.15), find the optimum step length λ_n by solving the problem

$$\lambda_n = \min_{0 \leq \lambda \leq 1} \sum_{a \in A} (c_a^{n-1} + \lambda(\overline{c}_a^n - c_a^{n-1}))(f_a^{n-1} + \lambda(\overline{f}_a^n - f_a^{n-1})) \ . \tag{3.17}$$

Solving problem (3.17) corresponds to optimize the original objective function over the segment joining the two points $(\overline{c}_a^n \overline{f}_a^n)$ and $(c_a^{n-1} f_a^{n-1})$. Since the feasible region is a convex space, this means that all the points in the segment are feasible, which in turn reduces problem (3.17) to solve a second degree polynomial, which can be done analytically.

Step 4: Set $c_a^n = c_a^{n-1} + \lambda(\overline{c}_a - c_a^{n-1})$, $f_a^n = f_a^{n-1} + \lambda(\overline{f}_a - f_a^{n-1})$ and go to Step 1.

3.3 Computational Experiments

3.3.1 Instances description

The assignment algorithm has been applied to two network instances, Köln and the central district of Portland. The two cities have been chosen because they represent two different network topologies, and it has been interesting how the SO assignment distributes the flows in both cities. Portland has an almost regular street grid as it is typical for US cities, while the Köln graph is irregular as it represents a historically grown European city. The transport graph of both cities has been extracted from the

major road network of each city using OpenStreet as data source. The OSMOSIS package has been employed to extract the main streets subgraph and SUMO (Simulation of Urban Mobility) generated the the directional transport network graph.

The link speeds have been assumed to be the same as speed limits for cars (taken from Openstreet data), the demand has been assumed to be uniform between randomly selected origins and destinations. Table 3.1 summarizes the information about the instances used: The Köln graph hae 1039 links and 558 nodes, while the much smaller Portland graph has only 355 links and 190 nodes. In order to simulate different network saturation levels, three sets of (uniform) demand-levels have been tested for each network instance. In absents of realistic information on the travel demand and in order to test the assignment for different demand scenarios, we have generated OD demand matrix in the following way: first the OD-pairs have been randomly selected from the set of all nodes of each graph. Then, a constant number of D trips between all selected OD-pairs has been assumed in order to mimic different demand levels.

In particular, the demand levels have been adjusted as to obtain a reasonable mix of links with congested and non-congested links.

Instance	OD Pairs	D
KOLN-162-50	162	50
KOLN-307-65	307	65
KOLN-1578-10	1578	10
PORT-361-50	361	50
PORT-1803-10	1803	10
PORT-3583-5	3583	5

TABLE 3.1: Instances characteristics.

3.3.2 Implementation of algorithm

We implemented the algorithm described in Section 3.2.3 in `C++`. In order to obtain the first feasible solution needed in Step 0, a minimun-cost linear Multi-Commodity Flow (MCF) problem is solved using the well known Dantzig-Wolfe decomposition approach, together with column generation (see e.g. [20, 21, 22]) where CPLEX 12.5 is used to solve both, the reduced master problem and the pricing problem. We use CPLEX 12.5 also to solve problem (3.16), (3.8)-(3.15). We have been running the algorithm on a computer with Intel® Pentium® 4, with 3.20GHz and 2GB of RAM, single core.

In Table 3.2 and Table 3.3 the obtained results are analyzed in more detail. We report the values of the best solution ("UB"), open Gap at each iteration and the cumulative computational time. The row corresponding to the iteration "0" reports the results obtained after solving the initial MCF problem.

	KOELN-162-50			KOELN-307-65			KOELN-1578-10		
It.	UB	Gap %	Time [s]	UB	Gap %	Time [s]	UB	Gap %	Time [s]
0	9905.8		1.5	49050.6		9.2	27131.3		27.6
1	9298.5	32.49	19.2	41158	45.45	562.7	18489.1	53.85	10845.7
2	7884.2	9.57	21.7	34984.2	6.2	590.7	17150.2	6.54	11240.9
3	7725.5	0.26	23.9	34635.1	3.3	606.8	16646.4	4.03	11397.9
4	7720	0.2	24.9	34169.5	0.41	620.3	16576.1	0.83	11518.7
5	7717.8	0.06	25.5	34158.5	0.38	630.7	16541.8	0.18	11593.1
6	-	-	-	34152.8	0.14	633	16535.2	0.21	11679.1
7	-	-	-	34146.8	0.07	637.5	16528.8	0.02	11721.5

TABLE 3.2: Köln Network, computational results.

	PORT-361-50			PORT-1803-10			PORT-3583-5		
It.	UB	Gap %	Time [s]	UB	Gap %	Time [s]	UB	Gap %	Time [s]
0	4441.1		2.6	3129.8		13	3205.4		26.6
1	3831.9	88.61	59	2645.1	77.77	894.2	2687.5	81.02	4177.6
2	2739.2	34.21	70.7	1958.7	31.98	1099	1965.2	33.56	5012.8
3	2635.2	24.99	80.8	1792	4.21	1261.7	1789.3	4.42	5650.1
4	2341.9	6.73	86.9	1775.8	3.25	1393.4	1775.4	3.2	6066
5	2326.2	2.12	91.4	1769.6	0.7	1488.8	1766.4	0.64	6344
6	2319.5	0.68	93.9	1767.9	0.66	1553.1	1764.8	0.8	6586.6
7	2316.2	0.58	96.2	1766.9	0.23	1581	1763.8	0.18	6732.5
8	2314.5	0.35	98	1766.3	0.28	1603.6	1763.5	0.16	6875.8
9	2313.5	0.2	99.4	1765.7	0.15	1633.9	1763.2	0.09	7002.1
10	2312.6	0.19	101	1765.6	0.15	1666.8	-	-	-
11	2312.3	0.13	102	1765.4	0.08	1691.2	-	-	-
12	2312.2	0.08	103.1	-	-	-	-	-	-

TABLE 3.3: Portland Network, computational results.

In each column of Figure 3.2, "Init" represents the amount of time spent to find the first feasible solution in Step 0. "It.1" and "It.s>1" represent, respectively, the amount of time spent for the first iteration of the Frank Wolfe algorithm and the amount of time spent for all successive iterations. Times are reported in percentage of the total time required to solve the problem.

It is easy to see that the first iteration of the Frank Wolfe algorithm is the most critical one, both in terms of gap reduction and in terms of computational time.

The strong improvement in the first iterations is typical for a first order method like Frank Wolfe algorithm. On the other hand, it is more interesting to analyze the time needed or solving each iteration. The complexity of each iteration is always the same and can be reduced to solve the problem (3.16),(3.8)-(3.15) with a generic LP solver. The difference in terms of time can be explained by the fact that starting from the second iteration, only the objective function (3.16) changes (accordingly to the new first order approximations), which enables the re-optimization of the LP solver to be more effective after the first iteration. Additional tests show clearly that after deactivating the pre-solver this advantage in terms of computing time disappears, showing a more balanced, but slower behavior .

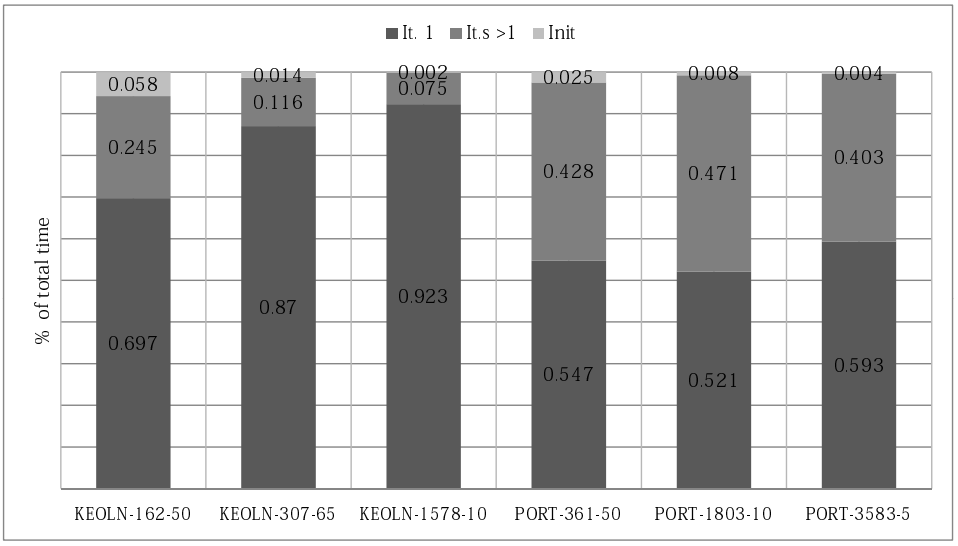

FIGURE 3.2: Percentage of total time spent for providing the first feasible solution, for running the first iteration of the Frank Wolfe algorithm and for running all the other iterations.

Finally, Table 3.4 shows the percentage of congested links (i.e. with $f_a > p_a$) for each iteration. The non negligible share of saturated links means that many demand flows from origin to destination have been effectively split into one or more route flows.

Iteration	1	2	3	4	5	6	7	8	9	10	11	12
KOELN-162-50	5.1	5.1	3.3	3.6	3.6	-	-	-	-	-	-	-
KOELN-307-65	19.6	14.9	15.2	14.1	15.3	15.3	15.4	-	-	-	-	-
KOELN-1578-10	10.8	10.1	10.3	10.8	9.8	10.3	10.1	-	-	-	-	-
PORT-361-50	69.1	51.1	62.1	51.1	62.8	65.9	62.1	63.6	63.6	63.6	62.8	63.6
PORT-1803-10	54.4	46.1	38.7	53	56.4	57.1	57.8	59.2	59.2	59.2	59.2	-
PORT-3583-5	51.1	47.3	35.5	53.7	59.2	56.4	56.4	57.1	57.1	-	-	-

TABLE 3.4: Percentage of congested links by iteration for both networks.

3.4 Conclusions

The particular requirements of a system optimal traffic assignment for PRT networks has been motivated and specified. A Bi-linear programming problem and algorithm has been proposed to assign traffic flows in a PRT network, taking into account the occupied and empty vehicle flows as well as the minimum safe headways vehicle control policy. As the PRT control is centralized, the system optimum traffic assignment minimizes the travel time of both, empty and occupied vehicles.

The convergence of the algorithm has been demonstrated analytically. The algorithm has been evaluated with two city-wide PRT network instances in terms of convergence and computing time. The algorithm allows for an optimal and rapid traffic assignment

which is useful for the network design. Furthermore it allows to determine the minimum number of vehicles necessary to generate the optimum flows and satisfy a given demand.

There are more potential applications for the proposed algorithm: it can be used to optimize the topology or graph orientation of large scale PRT networks as proposed in [13]. The traffic assignment could also be used as flow predictor in a closed loop real time vehicle-management system in order to optimize the dispatching of vehicles.

Bibliography

[1] Wardop, J.G., Some theoretical aspects of road traffic research. Proc. Inst. Civ. Eng. 2:325-378.

[2] Frank H. and Wolfe P. (1956). An Algorithm for quadratic programming, Naval Research Logistics Quarterly, 3 (1956), pp. 95-110.

[3] Beckmann M.J. (1965). On optimal tolls for highways, tunnels and bridges Vehicular Traffic Science, American Elsevier, New York (1965), pp. 331-341.

[4] Dafermos S.C. and Sparrow F.T.(1971). Optimal resource allocation and toll patterns in user-optimized transportation network Journal of Transportation Economics and Policy, 5 (1971), pp. 198-200.

[5] Sheffi Y. (1985). Urban Transportation Networks: Equilibrium Analysis with Mathematical Programming Methods Prentice-Hall, Englewood Cliffs, NJ (1985)

[6] Yang H., Huang H.J. (1998). Principle of marginal-cost pricing: How does it work in a general network? Transportation Research, 32A (1998), pp. 45-54

[7] Cascetta E (2001). Transportation systems engineering: theory and methods. Kluwer Academic Publisher

[8] Andréasson, I (1994). Vehicle Distribution in Large Personal Rapid Transit Systems. Transportation Research Record, No. 1451, pp 95-99, Transportation Research Board, Washington, D.C.

[9] Kaspi, M. and Tanchoco, J.M.A. (1990). Optimal flow path design of unidirectional AGV systems. International Journal of Production Research, 28, 1023-1030.

[10] Langevin, A., Riopel, D., Savard G., Bachmann, R. (2004). A multi-commodity network design approach of automated guided vehicle systems. INFOR, Vol 2, 2, 113-123.

[11] Benders, J.F. (1962). Partitioning procedures for solving mixed variables programming problems. Num. Math, 4, 238-252.

[12] Magnanti, T.L., Mireault, P., Wong, R.T.(1986) Tailoring Benders domposition for uncapacitated network design. Mathematical Programming Study 26, 112-154

[13] Traversi, E, Caprara, A, Schweizer, J (2009). An Application of Network Design with Orientation Constraints. In: Proceedings of the 7th Cologne-Twente Workshop on Graphs and Combinational Optimization. Gargnano, Italy, 13-15 May 2008, s.l: s.n, p. 16 - 21

[14] Danesi, A, Lupi, M, Rudi, A, Rupi, F, Schweizer, J, (2009) Economical feasibility study of a Personal Automated Transport for leisure, holidays and special events. In SIDT 2009, International Conference, Milano, Italy, 29/30 June 2009.

[15] Spivey M.Z, Powell W.B, The Dynamic Assignment Problem, TRANSPORTATION SCIENCE, Vol. 38, No. 4, November 2004, pp. 399-419

[16] Anil Y., Kaan O., Evacuation Network Modeling via Dynamic Traffic Assignment with Probabilistic Demand and Capacity Constraints, Journal of the Transportation Research Board, No. 2196, pp 11-20, ISSN: 0361-1981

[17] Lees-Miller J.D, Hammersley J.S, Wilson R.E, Theoretical Maximum Capacity as a Benchmark for Empty Vehicle Redistribution in Personal Rapid Transit, Journal of the Transportation Research Board, No. 2146, pp 76-83, Transportation Research Board, ISSN: 0361-1981

[18] Koskinen K., Luttinen T., Kosonen I., Developing a microscopic simulator for personal rapid transit (PRT) systems, Transportation Research Board 86th Annual Meeting, Washington, D.C., 2007

[19] Schweizer J., Danesi A., Rupi F., Comparison of static vehicle flow assignment methods and microsimulations for a personal rapid transit network, J. Adv. Transp. 2012; 46:340-350

[20] Tomlin J. A., Minimum-Cost Multicommodity Network Flows. Operations Research, 1966, 45-51.

[21] Lübbecke M. E., Desrosiers J., Selected topics in column generation. Operations Research, 2005, 53.6: 1007-1023.

[22] Frangioni A., Gallo G., A bundle type dual-ascent approach to linear multicommodity min-cost flow problems. INFORMS Journal on Computing, 1999, 11.4: 370-393.

[23] Wächter, A., Biegler, L. T., On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. Mathematical Programming, 2006, 106.1: 25-57.

Chapter 4

Single-commodity Robust Network Design Problem: Complexity, Instances and Heuristic Solutions[1]

4.1 Introduction

Network design problems arise in many different areas, such as transportation and telecommunication. Recently, the class of *robust network design* problems has received increasing attention. The term robust can represent the capability of the network to cope with disruptions or to deal with different traffic scenarios in different times of the day, as is the case of our work.

We study the *single-commodity* Robust Network Design problem (RND) defined as follows. We are given an undirected graph $G = (V, E)$, a cost vector (c_e) $(e \in E)$ and an integer *balance* matrix $B = (b_i^q)$ $(i \in V,\ q = 1, \ldots, K)$. The q-th row b^q of B is called the q-th *scenario*.

For a given scenario, we call a node with nonzero balance a *terminal*. More specifically, a node i with positive balance is called a *source* and we call the balance of i its *supply*. A node with negative balance is called a *sink* and its balance is called *demand*.

[1] **Published** - "*Models and Algorithms for Robust Network Design with Several Traffic Scenarios*", E. Álvarez-Miranda, V. Cacchiani, T. Dorneth, M. Jünger, F. Liers, A. Lodi, T.Parriani, D. Schmidt, *Combinatorial Optimization, LNCS series, Volume 7422, 2012, pp 261-272*; **Under review** - "*Single-commodity Robust Network Design Problem: Complexity, Instances and Heuristic Solutions*", E. Álvarez-Miranda, V. Cacchiani, A. Lodi, T.Parriani, D. Schmidt, "*European Journal of Operational Research*".

Let us denote by (i,j) and (j,i) the arcs (directed from i to j and from j to i, respectively) corresponding to edge $e = \{i,j\} \in E$. In addition, let us call $f^q_{i,j} \in \mathbb{Z}_+$ the integral amount of flow that is sent along arc (i,j) from i to j in scenario q and by f^q the corresponding flow vector.

RND calls for determining integer capacities $(u_e) \in \mathbb{Z}_+^{|E|}$ $(e \in E)$ with minimal costs $c^T u$ such that, for each q $(q = 1, \dots, K)$, there is a directed network flow f^q in G that is feasible with respect to the capacities and the balances of the q-th scenario. In particular, the flow f^q $(q = 1, \dots, K)$ must fulfill the following constraints:

1. $f^q_{i,j} + f^q_{j,i} \leq u_e$ for all edges $e = \{i,j\} \in E$, which imposes that the sum of the flows going along every edge (in both directions) must respect the installed edge capacity, for every scenario,
2. $\sum_{\{i,j\} \in E} (f^q_{i,j} - f^q_{j,i}) = b^q_i$ for all nodes $i \in V$, which implies that the flow must satisfy the required integer balances.

An overall natural model for RND reads as follows

$$\min \sum_{\{i,j\} \in E} c_{ij} u_{ij} \tag{4.1}$$

$$\sum_{j:\{j,i\} \in E} f^q_{ji} - \sum_{j:\{i,j\} \in E} f^q_{ij} = b^q_i \qquad \forall i \in V,\ q = 1, \dots, K \tag{4.2}$$

$$f^q_{ij} + f^q_{ji} \leq u_{ij} \qquad \forall \{i,j\} \in E,\ q = 1, \dots, K \tag{4.3}$$

$$f^q_{ij} \geq 0 \qquad \forall \{i,j\} \in E,\ q = 1, \dots, K \tag{4.4}$$

$$u_{ij} \in \mathbb{Z}_+, \qquad \forall \{i,j\} \in E \tag{4.5}$$

where the objective function (4.1) is to minimize the total cost of the installed capacities. Constraints (4.2) ensure flow-conservation in each scenario and impose to satisfy the required balances. Constraints (4.3) model that the capacity of an edge is at least as large as the flow it carries. Integral flows are enforced through integrality of the capacity variables, as all balances are integral [17].
As described in [9], an example of a practical application of the considered problem is the following: some clients wish to download some program stored on several servers. For a client, it is not important which server he or she is downloading from, as long as the demand is satisfied. In other words, we consider servers that store identical data: examples are video on demand or large datacentre in which one mirrors his data over several locations. This is opposed to multi-commodity network design, in which point-to-point connections are considered, i.e. each client requests a specific server. In addition, we consider the robust version of the problem: at different times of the day, the demands may change (e.g. different clients show up), and the goal is to design a network that is able to route all flow in all different scenarios. In particular, we consider a finite list of demands, i.e. we sample different times of the day.

Contribution. Preliminary computational investigations have been performed on classical graphs from the literature with random balances [9] and on special hypercubes with $\{-1, 0, 1\}$ balances [2]. The results in both papers have shown that the former instances are surprisingly easy for a general-purpose Mixed-Integer Programming (MIP) solver on the natural flow-formulation (4.1)–(4.5), while the latter instances are structurally difficult. The first contribution is in studying the complexity of some RND special cases[2] associated with the above instances and enlightening the reasons of the observed computational behavior. Second, based on the complexity results, we propose a new family of randomly generated RND instances that are computationally challenging for the natural flow formulation already for $|V| = 50$ and $K = 10$. Third, motivated by those instances (available upon request from the authors), we propose new and general heuristic approaches that provide high-quality approximated solutions for large graphs (tests are reported for $|V|$ up to 500) in short computing times[3].

Organization of the chapter. Section 4.2 reviews the (vast) related literature by pointing out differences and similarities. In Section 4.3 we present the complexity results we achieved on special classes of instances, while Section 4.4 describes the proposed heuristic algorithm and its performance is reported in Section 4.5. Finally, in Section 4.6 we draw conclusions and describe ideas for future research.

4.2 Related Literature

The work on classical (i.e., non-robust) network design goes back as far as the early 1960s where it was studied by Chien [11] and Gomory and Hu [16, 15]. Since then, network design has evolved to a vast field of research which we cannot fully discuss in the scope of this article. We rather refer to [10] for a complete overview and restrict ourselves to a few exemplary related works that are of direct importance for us here.

The common theme of network design problems is installing optimum-cost capacities in a given network topology such that a set of traffic requests can be routed through the network. In practice, however, the traffic requests are not exactly known in advance. This can be due to measuring errors or simply because they cannot be predicted [6]. Here, the robustness comes in: Following an idea by Soyster [29], Ben-Tal and Nemirowski [5] coined the term of an *uncertainty set* that is added to the model and contains all possible (or likely) scenarios against which the robustness should protect. Since then, robust network design has been very actively studied. The notions of *network topology*, *cost*, *capacity*, *traffic request* and *routing* can vary – as well as the exact way in which the problem is *robustified*.

[2] The RND problem is strongly NP-hard [27].

[3] A preliminary version of the heuristic approaches described here was introduced in [2] where the first phase of the investigation on RND, which was the topic of the "Vigoni 2011-2012" project between the University of Köln and the University of Bologna, was summarized.

We here study a *worst-case* robust model in the sense of [5]. This means that our solutions must be feasible for all the scenarios from the uncertainty set. The uncertainty set is *finite* and *explicitly given* as part of the input (an idea that goes back to [23]). We use an *undirected* graph as the network topology and allow *dynamic routing* (each scenario may be routed on different paths). Furthermore, we assume *linear costs* for the capacities and integer multiples of a unit capacity may be installed on each edge. Each node specifies its traffic request by a scalar number that gives its supply or demand and each such traffic request may be routed on an arbitrary number of paths (the routing is *splittable*) as long as each edge carries an integer amount of flow in total. Therefore, the underlying flow model is a standard *single-commodity*, splittable network flow in our case.

To the best of our knowledge, only two prior publications on this specific problem exist. The problem was first studied by Buchheim, Liers and Sanità [9]. They gave an exact branch-and-cut algorithm that solves a flow-model MIP through sophisticated general-purpose cutting planes. Lately, Álvarez-Miranda et al. [2] introduced a capacity-based MIP-model, and discussed a preliminary set of results of the biennial "Vigoni 2011-2012" between the universities of Köln and Bologna.

Atamtürk [3] considers a variant of the non-robust single-commodity network design problem where integer multiples of a facility with fixed capacity can be installed on each arc. Ortega and Wolsey [24] report on the performance of general MIP solvers on various network design problems and develop an exact algorithm for the single-commodity fixed-charge network design problem (all arcs may be bought at a fixed-charge and then be used at full capacity).

A close variant of single-commodity RND is the *multi-commodity* robust network design problem. Here, the traffic requests specify the amount d_{ij} of flow that should be exchanged among all pairs of nodes i and j. In particular, this defines fixed source/sink pairs – which is not the case in our problem. Also, each commodity has a single source (or sink). While this condition can also be established in the single-commodity case, it requires the use of fixed-capacity edges and therefore, our single-commodity variant is not a true special case of the multi-commodity problem. Sanità showed in her doctoral thesis [27] that the multi-commodity variant is NP-hard even if there are only three scenarios, all scenarios use a unique source node and all demands are from $\{-1, 0, 1\}$. This immediately implies that the single-commodity variant is NP-hard as well. The thesis contains many further complexity results; among others Sanità gives a $O(\log |V|)$-approximation for the multi-commodity robust network design problem with unsplittable routing and shows that removing the integrality constraint from the capacities makes the problem polynomial time solvable. This is also true for the single-commodity RND. The multi-commodity RND was also first considered as a classical (non-robust) problem [8].

A vast variety of problems exists in the multi-commodity case. The case where the uncertainty set is finite was studied by Minoux [23], though fractional capacities are assumed in [23], and in Labbé et al. [20]. Duffield et al. [12] introduced the Hose uncertainty model in which the uncertainty set is defined by inflow and outflow bounds on all nodes. Ben-Ameur and Kerivin [4] observed that this type of uncertainty set is a polytope and developed an exact approach that additionally assumes *static routing* (i.e., in all scenarios, the flow must be routed along the same subset of paths). This configuration is also known as the *Virtual Private Network* problem. An exact approach for this problem was given in [1] under the additional constraint that each commodity may only use a single path (unsplittable routing).

In the case of dynamic routing, an exact approach by Mattia [21] exists. Bertsimas and Sim [7] introduced Γ-robustness as a general model for robustification. Exact approaches that apply this type of robustness to multicommodity network design are presented in [19].

Finally, one of the most basic network design problems, the *Steiner Tree* problem, is the special case of the single-commodity robust network design problem where for each pair i, j of Steiner nodes, there exists a scenario in which exactly i and j are terminals with supply/demand of $1/-1$. If not all the Steiner node scenarios are present, the single-commodity RND instance is instead a special case of the survivable network design problem. Note, however, that, in general, RND does not consider the requirement of disjointness that is in Survivable Network Design. We refer the reader to [18] for an extensive survey on this subject.

4.3 Complexity

In this section, we characterize the complexity of some RND special cases. The RND case in which we have a single scenario ($K = 1$) corresponds to a standard polynomial time minimum cost flow problem. Already for $K = 3$, RND is NP-hard (see [27]): the reduction comes from the 3-Dimensional Matching Problem for the special case of RND in which there is the same source in each scenario and balances are $\{-1, 0, 1\}$.

Motivated by the computational investigations in [9, 2], in the following, we analyze some special cases:

- RND with balances different from 1 and -1;
- RND on hypercubes with all balances equal to 1, 0, or -1;
- RND on hypercubes with all balances equal to r, 0, or $-r$, with r integer and > 1.

The analysis is intended to show some classes of hard instances and some classes of easier instances. According to the results that we present in the following subsections, we are able to get a better understanding of empirical results in [9, 2], and we propose a family of randomly generated instances that are challenging for the natural flow formulation already for $|V| = 50$ and $K = 10$.

4.3.1 All balances different from 1 and -1

Because instances defined on random graphs with random integer balances on the (randomly chosen) terminals turn out to be surprisingly easy for a general-purpose MIP solver on the natural flow-formulation (4.1)–(4.5), a natural question to ask is if this special case remains NP-hard. The following theorem answers positively through a reduction from Hamiltonian cycle ([28]).

In order to prove that RND, defined on graph $G = (V, E)$ ($|V| \geq 3$), with balances different from 1 and -1, is NP-hard, let us define the following RND instance I_R. We use G without modification and install a cost of 1 on each edge. We choose some arbitrary numbering of the nodes. We install $|V| - 1$ scenarios. In scenario i, only nodes 1 and $i + 1$ are terminals; the node 1 gets a balance of 2 while the node $i + 1$ has a balance of -2.

Theorem 1. *A graph $G = (V, E)$ (with $|V| \geq 3$) has a Hamiltonian cycle C if and only if the described RND instance I_R has a solution with cost equal to $|V|$.*

Proof. If G has a Hamiltonian cycle C, we build a feasible solution for I_R by installing a capacity of 1 on each edge of C. In each scenario i, both unique terminals 1 and $i+1$ lie on C. The node $i+1$ decomposes C into two paths P_1, P_2 from 1 to $i+1$ (one clockwise, one counterclockwise). We can route one unit of flow on P_1 and one unit of flow on P_2, satisfying the demands of scenario i. Thus, our solution for I_R is feasible and additionally, it has cost of $|C| = |V|$.

On the other hand, suppose we have a solution for I_R of cost $|V|$. By our choice of scenarios (we have a single source at node 1 and all other nodes are terminals in some scenario), each node must be connected to node 1. Therefore, any feasible solution for I_R must have a support S that induces a connected component of G containing all nodes. S must contain at least $|V| - 1$ edges, otherwise it cannot be connected. If S contains exactly $|V| - 1$ edges, a capacity of 2 must be installed on each edge in S in order to route all demands. However, such a solution has cost of $2 \cdot |V| - 2 > |V|$ and therefore S must contain at least $|V|$ edges. If some node in $G[S]$ has a degree of 1, then we must install a capacity of 2 on its unique incident edge. By the same argument as before, the remaining nodes $|V| - 1$ nodes must be connected by at least $|V| - 1$ edges. Then again, the cost of the solution is at least $|V| - 1 + 2 > |V|$. Therefore, all nodes in $G[S]$ must have a degree of at least 2 and because we can have at most $|V|$

edges in S, each node must have exactly degree 2. Together with our observation that $G[S]$ is connected and contains all nodes, we have a Hamiltonian cycle. □

4.3.2 Hypercubes

The authors defined a structurally difficult class of instances in [2], based on d-dimensional hypercubes. In the following we repeat the construction.

Definition 2. *A* d-dimensional hypercube $\mathcal{H}_d$ *is the result of the following recursive construction:* $\mathcal{H}_0$ *is the graph that consists of a single node. For* $d > 0$, $\mathcal{H}_d$ *is obtained by duplicating the nodes and edges of* $\mathcal{H}_{d-1}$ *and connecting each node* v *to its copy* v' *with an additional edge* $\{v, v'\}$.

Definition 3. *We say that two nodes* v, w *are* diagonally opposite *on* $\mathcal{H}_d$ *iff the shortest path from* v *to* w *in* $\mathcal{H}_d$ *has maximum length, i. e., length* d.

Notice that for every node v in $\mathcal{H}_d$ there is exactly one node v^o that is diagonally opposite to v. It is well-known that $\mathcal{H}_d$ has $N_d := 2^d$ nodes and $M_d := d \cdot 2^{d-1}$ edges.

We can now define a class of instances on d-dimensional hypercubes as follows. For $d \in \mathbb{Z}_+$, consider the following instance I_d of the RND problem on $\mathcal{H}_d$. Observe that $\mathcal{H}_d$ is composed of two hypercubes H^s, H^t of dimension $d-1$. Add 2^{d-1} scenarios to $\mathcal{H}_d$. In scenario $1 \leq q \leq 2^{d-1}$, assign a supply of 1 to the q-th node v_q (in some fixed numbering) of H^s and a demand of -1 to its diagonally opposite node v_q^o which lies in H^t by our construction. Set all other balances of scenario q to zero and set the costs for each edge to 1. Figure 4.1 shows the construction.

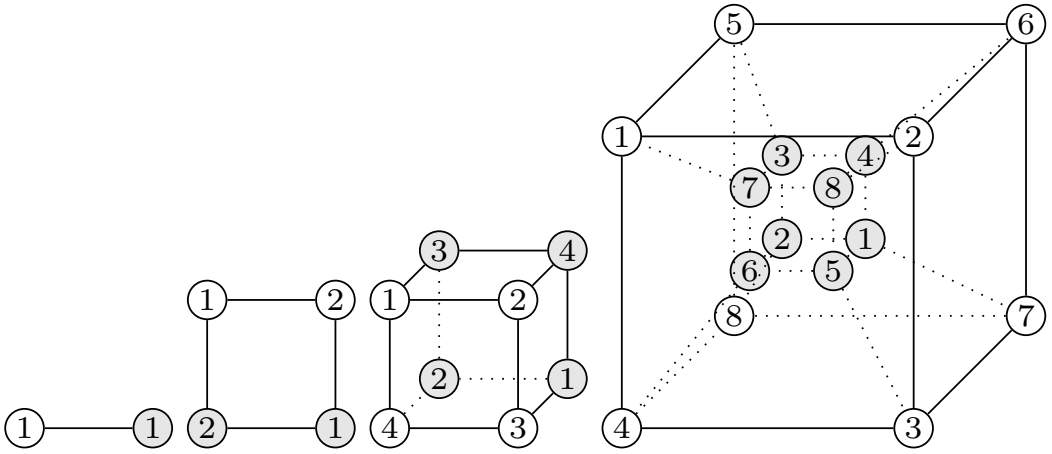

FIGURE 4.1: The hypercubes in 1, 2, 3 and 4 dimensions. Copied nodes are displayed in gray. The node numbering refers to the scenarios.

We denote the instance obtained in this way by $\mathcal{H}_d^1$. Scaling all balances in $\mathcal{H}_d^1$ by $r \in \mathbb{Z}_+$, we obtain the instance $\mathcal{H}_d^r$.

4.3.2.1 All balances equal to 1, 0, or -1

It is shown in [2] that this class of instances is difficult for MIP-based solution approaches as the integrality gap (i.e., the ratio of an optimum integral solution value and an optimum fractional solution value) of $\mathcal{H}_d^1$ converges to 2 as $d \to \infty$. We refer the reader to [2] for details.

4.3.2.2 All balances equal to r, 0, or $-r$, r integer and > 1

We characterize the integrality gap for $r > 1$. The optimum values for integer and fractional solutions are the same, i.e. the integrality gap is 1. We need a series of Lemmata to prove this result, stated and proven at the end of this section.

It is a well-known fact that $\mathcal{H}_d$ is hamiltonian for any $d \geq 2$ and we shall use this fact on several occasions. In particular, we can obtain a feasible integer solution for $\mathcal{H}_d^2$ by installing a capacity of 1 on each edge of a Hamiltonian cycle in $\mathcal{H}_d^2$.

Lemma 4. *For any $d \geq 2$, there is a feasible integer solution for $\mathcal{H}_d^2$ with costs 2^d.*

To derive the cost of this solution, recall that $\mathcal{H}_d$ has 2^d nodes. Similarly, we can state a feasible integer solution for $\mathcal{H}_d^3$.

Lemma 5. *For any $d \geq 3$, there is a feasible integer solution for $\mathcal{H}_d^3$ with costs $3 \cdot 2^{d-1}$.*

Proof. Let $d \geq 3$. Then $\mathcal{H}_d$ decomposes into two copies H_1, H_2 of $\mathcal{H}_{d-1}$ and a set of edges F connecting H_1 and H_2. We install a capacity of 1 on each edge in F. Since $d - 1 \geq 2$, we find Hamiltonian cycles C_1, C_2 in H_1 and H_2, respectively, and install a capacity of 1 on each edge of C_1 and of C_2.

This solution is feasible: For any scenario $i \in \{1, \ldots, q\}$, let s_i, t_i be the corresponding terminal pair. We need to route three units of flow from s_i to t_i. To do that, let $s_i' \in H_2$ and $t_i' \in H_1$ be the unique nodes such that $e_1 = \{s_i, s_i'\} \in F$ and $e_2 = \{t_i', t_i\} \in F$. Also, let $e_3 = \{u, v\} \in F$ with $u \in H_1$ and $v \in H_2$ be an arbitrary connecting edge that is different from e_1 and e_2. Mark here that F contains at least four edges because $d \geq 3$. Figure 4.2 shows an example for the situation on $\mathcal{H}_4^3$. Now, by sending one unit of flow over each of e_1, e_2, e_3, we have reduced the instance to two instances on $\mathcal{H}_{d-1}$: The first instance is defined on H_1; here, s_i has a balance of 2 and both u and t_i' have a balance of -1. However, these balances can be routed along the Hamiltonian C_1. In the second instance, which is defined on H_2, the sink t_i has a balance of -2 and both s_i' and v have a balance of 1. Again, these balances can be routed along the Hamiltonian cycle C_2.

Both C_1 and C_2 contain exactly 2^{d-1} edges, each with capacity 1. There are 2^{d-1} edges in F, all of them having capacity 1. This gives a total cost of $3 \cdot 2^{d-1}$. □

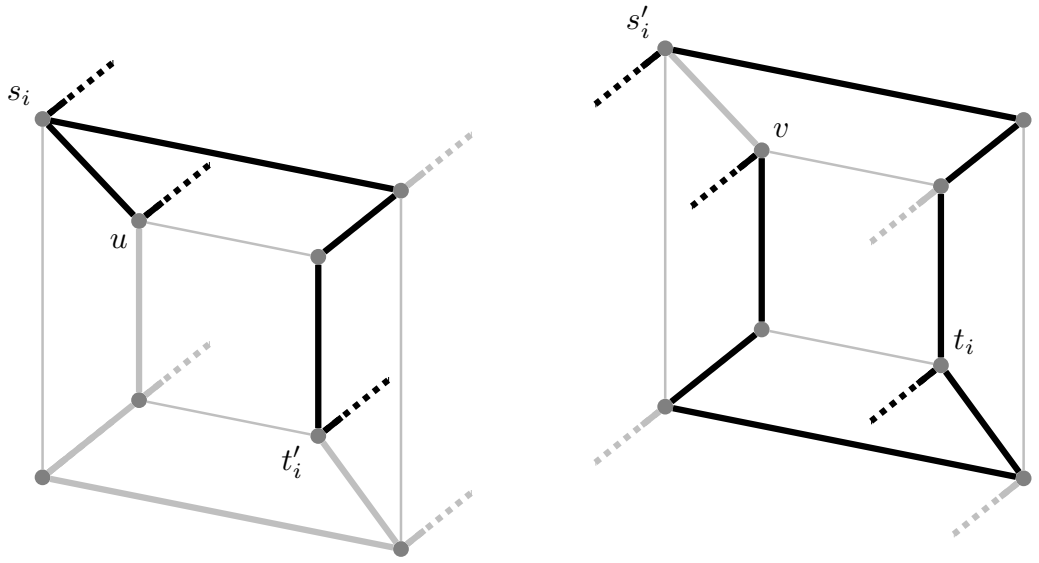

FIGURE 4.2: An example for $\mathcal{H}_4^3$.

We show next that we can construct an integer feasible solution for any $\mathcal{H}_d^r$ using the two previous ones.

Lemma 6. *Let $d \geq 2$ and let $r = 2m + 3n$ with $m \in \mathbb{Z}_+$ and $n \in \{0, 1\}$. If there exists an integer feasible solution for $\mathcal{H}_d^2$ with cost at most c_2 and an integer feasible solution for $\mathcal{H}_d^3$ with cost at most c_3, then there exist an integer feasible solution for $\mathcal{H}_d^r$ with cost at most*

$$m \cdot c_2 + n \cdot c_3.$$

Proof. We can decompose $\mathcal{H}_d^r$ into m copies of $\mathcal{H}_d^2$ and, if r is odd, a single copy of $\mathcal{H}_d^3$. The copies have costs of c_2 and c_3 each, respectively. For the i-th copy and $i = 1, \ldots, m + n$, we have an integer capacity vector u_i that allows for routing all scenarios. Then, $u = \sum_{i=1}^{m+n} u_i$ is an integer capacity vector that admits a routing of all scenarios of $\mathcal{H}_d^r$ and has exactly cost $mc_2 + nc_3$. □

To calculate the integrality gap for our solutions, we also need the value of an optimum fractional solution. Such a solution can be obtained by installing r/d units of capacity on each edge of $\mathcal{H}_d$ and since $\mathcal{H}_d$ has $d \cdot 2^{d-1}$ edges, this gives the following result.

Lemma 7. *An optimum fractional solution for $\mathcal{H}_d^r$ has a value of $r \cdot 2^{d-1}$.*

Proof. Check that a solution that installs r/d units of capacity on each edge satisfies the complementary slackness optimality conditions of the *cut-set formulation* [2] for the problem. The corresponding dual solution has a variable ξ_S for all meaningful cuts $S \subseteq V[\mathcal{H}_d]$. It sets $\xi_S := 1/2$ if $|\delta(S)| = d$ and $\xi_S = 0$. The remaining part of the proof is straight-forward if we observe that for $d \geq 3$, we have $|\delta(S)| = d$ if and only if $|S| = 1$. The full proof is shown in the Appendix. □

We can now prove that the optimum values for integer and fractional solutions are the same:

Theorem 8. *For $d \geq 3$ and $r \geq 2$, an optimum integer solution for $\mathcal{H}_d^r$ has value $r \cdot 2^{d-1}$. In particular, the integrality gap for $\mathcal{H}_d^r$ is 1.*

Proof. Let $r = 2m + 3n$ with $m \in \mathbb{Z}_+$ and $n \in \{0, 1\}$. Putting together Lemma 6 with Lemma 4 and Lemma 5, we obtain that there is an integer solution for $\mathcal{H}_d^r$ with value $c_r := m \cdot 2^d + n \cdot 3 \cdot 2^{d-1}$. If r is even, we have $n = 0$ and $m = r/2$. Therefore, $c_r = r \cdot 2^{d-1}$. On the other hand, if r is odd, we have $n = 1$ and $m = (r-3)/2$. Then, $c_r = (r-3)/2 \cdot 2^d + 3 \cdot 2^{d-1} = r \cdot 2^{d-1} - 3 \cdot 2^{d-1} + 3 \cdot 2^{d-1} = r \cdot 2^{d-1}$. By Lemma 7, this is optimal. □

4.3.3 Challenging Instances

In the previous sections we have shown that, although *computationally easy* [9, 2], RND instances defined on random graphs with random balances are difficult in theory. The explanation of this is suggested by the fact that structurally hard instances like those defined on hypercubes and $\{-1, 0, 1\}$ balances become *theoretically easy* when balances are in r, 0, or $-r$, with r integer and $r > 1$. have an integrality gap of value one. Building on top of those results, we concentrate on instances on random graphs with balances $\{-1, 0, 1\}$ that turn out to be computationally challenging for the natural flow formulation already for $|V| = 50$ and $K = 10$. An effective heuristic approach for this family is described and computationally evaluated in Section 4.4 and Section 4.5, respectively.

4.4 Heuristic Algorithm

In this section, we present our heuristic algorithm, which, although general, is designed having in mind the class of hard instances introduced in the precious section, i.e., random graphs with balances of $\{-1, 0, 1\}$. It consists of three phases. In the first phase (*constructive phase*, CP), the graph is reduced by heuristically deleting a subset of the arcs, and a feasible solution is built. The second phase (*neighborhood search phase*, NSP) consists of a neighborhood search on the reduced graph in order to improve the solution found: in particular, the MIP flow-formulation is solved, within a time limit, by the general-purpose MIP solver Cplex. Finally, the third phase (*proximity search phase*, PSP) consists of iteratively applying a local search (by solving a carefully constructed MIP) to further improve the solution, taking into account the original graph, and is based on the recent work [13].

In the following, we describe the three phases in detail.

4.4.1 Constructive Phase

Initially, the graph we are dealing with is reduced, and then a solution is built. Our goal is to reduce the graph so that we are able to quickly compute a feasible solution, and we can warm start the NSP described in 4.4.2. At the same time, the graph reduction should not be too "aggressive", because the NSP should be able to improve the solution found. In other words, we need to find a trade-off between reducing the computing time and reducing the solution space. Note that, since the (nonzero) balances are 1 or -1 in our problem, it is not common to have large capacity values installed on the edges. Therefore, solutions differ mainly because of the different set of edges on which capacity is installed. Our goal is to select a "large enough" set of edges for our reduced graph.

The following steps are executed in the CP:

1. Consider the scenarios from 1 to K and multiply all balances by a given constant F;
2. construct a feasible solution for the new obtained RND instance (see Section 4.4.1.1);
3. reduce the graph by deleting all the edges that are not used in the solution found (and the nodes such that they do not have any incident edge after edge deletion) and obtain graph $\overline{G} = (\overline{V}, \overline{E})$;
4. set back the balances to 1 and -1, and construct a feasible solution (see Section 4.4.1.1) for the original RND instance on the reduce graph $\overline{G}$.

Step 1 is used to define the search space that we want to use in the NSP. Indeed, by increasing the absolute value of the balances, more edges are likely to be used in the solution computed in step 2 and they constitute the neighborhood of the solution computed in step 4. The next section describes how to compute a feasible solution for an RND instance.

4.4.1.1 Construction of a Feasible Solution

In the case of a single scenario, an algorithm for the Minimum Cost Flow (MCF) problem can be used to solve RND as follows: we define a directed graph having the same set of nodes as G and two arcs for each edge of G (one for each direction) with infinite upper bounds on the capacities. The flows that we obtain by solving the MCF problem on the defined graph determine the edge capacities, i.e., the RND solution.

In the case of K scenarios, ordered from 1 to K, in a generic scenario q we can use for free the capacities that have already been installed on the edges in scenarios $1, \ldots, q-1$.

A straightforward heuristic algorithm consists of iteratively solving a MCF problem for each scenario (in the order from 1 to K), updating the capacities that can be used for free after each MCF execution. In particular, we define an auxiliary directed graph $G_{dir} = (V, A)$ having the same set of nodes of G and the set of arcs defined as follows. For each edge $e = \{i, j\} \in E$, we introduce four arcs a_1^e, a_2^e, a_3^e and a_4^e: a_1^e and a_2^e are directed from i to j, while a_3^e and a_4^e are directed from j to i. Two arcs are needed for each direction in order to take into account, in a generic scenario q, the previous scenarios $1, \ldots, q-1$: one arc has an upper bound on its capacity equal to the capacity already installed on the corresponding edge in the previous scenarios $1, \ldots, q-1$ and has zero cost; the other arc has an infinite upper bound on its capacity and has cost equal to the cost of the corresponding edge. More precisely, for each arc $a \in A$, we initialize the upper bounds UB_a on the capacities and the costs c_a as $UB_{a_1^e} := \infty$, $UB_{a_2^e} := 0$, $UB_{a_3^e} := \infty$ and $UB_{a_4^e} := 0$; $c_{a_1^e} := c_e$, $c_{a_2^e} := 0$, $c_{a_3^e} := c_e$, $c_{a_4^e} := 0$. A MCF problem is then solved for each scenario and the upper bounds are updated according to the capacities installed on each arc.

The described algorithm follows a greedy approach. It would be useful if, when solving scenario q, we could know what happens in the next scenarios $q+1, \ldots, K$ so that we could choose accordingly the best capacity installation. In addition, a MCF solution for a generic scenario q that installs capacity on more edges (at the same cost) should be preferred: indeed, it is more likely that free capacity can be used in scenarios $q+1, \ldots, K$. Therefore, a MCF solution with integer flows split over disjoint paths should be preferred with respect to a MCF solution that sends flows along a single path.

Based on these two observations, we derive an improvement of the described heuristic algorithm. We apply a preprocessing in which we divide each scenario $q = 1, \ldots, K$ in R sub-scenarios $g_1^q, \ldots, g_R^q$, where R is an integer positive number. We consider the sub-scenarios in the order g_1^q, $(q = 1, \ldots, K)$, g_2^q, $(q = 1, \ldots, K)$, up to g_R^q, $(q = 1, \ldots, K)$. In this way, the generic sub-scenario g_l^q of scenario q can already take into account the partial solution computed for all the scenarios $1, \ldots, K$. The balances are defined as follows: $b_v^{g_1^q} = \lfloor b_v^q / R \rfloor$, $b_v^{g_2^q} = \lfloor b_v^q / (R-1) \rfloor$, up to $b_v^{g_R^q} = b_v^q$, $v \in V$. This means that the complete MCF solution of a generic scenario q will more likely have a split integer flow over disjoint paths, because each sub-scenario might use different subsets of arcs.

The improved heuristic algorithm iteratively solves a MCF problem for each sub-scenario g_l^q $(l = 1, \ldots, R,\ q = 1, \ldots, K)$. Let us call u^{RND} the RND solution that we compute with the improved heuristic algorithm. At $h = 0$, u^{RND} is initialized to be the zero vector. Let f^{h*} be the MCF solution obtained at iteration h corresponding to sub-scenario g_l^q. The flows in f^{h*} along the arcs with infinite upper bound determine the additional capacities that must be installed on the corresponding edges: for each $e = \{i, j\} \in E$, $u_e^{RND} = u_e^{RND} + f_{a_1^e}^{h*} + f_{a_3^e}^{h*}$. Note that, in each sub-scenario, there will

always be an optimal solution using, for each edge, only arcs in one of the two directions: it is a single commodity flow, so we could simply do flow cancellation on cycles. In addition, the values u^{RND} are used to update the upper bounds on the capacities, before considering the following sub-scenario: $UB_{a_2^e} := u_e^{RND}$ and $UB_{a_4^e} := u_e^{RND}$. When all the sub-scenarios have been considered, the algorithm returns the solution found u^{RND}.

Note that in step 2 the described algorithm is used to define the reduced graph $\overline{G}$: all the edges such that $u^{RND} = 0$ are deleted from G and the nodes that do not have anymore incident edges are removed as well. Step 4 is instead used to obtain a first feasible solution to our problem and is executed on the reduced graph $\overline{G}$. Let us call u^{CP} the solution obtained at the end of the constructive phase.

4.4.2 Neighborhood Search Phase

This phase consists of solving an MIP flow-formulation for RND (4.1)–(4.5) on the reduced graph $\overline{G}$ defined in the previous section.

Then, NSP explores the neighborhood of solution u^{CP} by allowing the use of different edges belonging to $\overline{G}$ and by allowing the installation of different capacities on the edges. The neighborhood is explored by solving the proposed model, initialized with u^{CP}, by Cplex within a given time limit. Let us call u^{NSP} the obtained improved solution and c^{NSP} its cost. Since we consider the reduced graph, this phase is able to quickly obtain an improved solution, as it will be seen in Section 4.5.

4.4.3 Proximity Search Phase

Recently, Fischetti and Monaci [13] investigated the effects of replacing the objective function of a 0-1 Mixed-Integer Convex Programming problem with a "proximity" one, i.e., with minimizing the distance from a feasible solution of the problem, with the aim of enhancing the heuristic behavior of a black-box solver. In particular, they consider the Hamming distance:

$$\Delta(x, \tilde{x}) := \sum_{j \in J: \tilde{x}=0} x_j + \sum_{j \in J: \tilde{x}=1} (1 - x_j), \tag{4.6}$$

where $x_j \in \{0, 1\}$, $\forall j \in J$, and $\tilde{x}$ is a feasible solution to the considered problem. The idea consists of starting with an initial feasible solution $\tilde{x}$ with cost $f(\tilde{x})$, and iteratively searching for an improved solution by adding a cutoff constraint that imposes the cost of the improved solution to be smaller than $f(\tilde{x})$ by at least a quantity θ. The search is performed by solving with a black-box solver the new model with objective function that minimizes the Hamming distance from $\tilde{x}$, until a termination condition is reached, namely, until the first improved solution has been found. If no improved solution is

found, θ is reduced. The process is then iterated by using the improved solution found as new $\tilde{x}$. The algorithm is terminated when a given time limit is reached. The method can be enhanced by providing an incumbent solution to each iteration of proximity search. This is obtained by adding an auxiliary continuous variable z which is used to keep the cutoff constraint feasible:

$$f(x) \leq f(\tilde{x}) - \theta + z \tag{4.7}$$

and has a large cost M in the objective function. In this way, $\tilde{x}$ is a (very costly) feasible solution for the MIP it defines. As soon as z becomes 0, an improved solution is found.

We apply this idea to RND, i.e. we deal with an MIP. We start with initial solution u^{NSP} and we consider the original graph G (instead of the reduced one) in order to have a higher probability of improving u^{NSP}. Since capacities assume integer (and not only binary) values, we need to modify the definition of distance presented in [13]. Instead of expressing the distance as $|u - u^{NSP}|$, u_{ij} integer $\forall\{i,j\} \in E$, we fix upper bounds on the capacity variables, based on the values of u_{ij}^{NSP}, as follows. For each edge $\{i,j\} \in \overline{E}$ such that $u_{ij}^{NSP} > 0$, the upper bound is set to u_{ij}^{NSP}. For all the remaining edges the upper bound is the set to be infinite. The distance is then defined as

$$\sum_{\{i,j\} \in E : u_{ij}^{NSP} = 0} u_{ij} + \sum_{\{i,j\} \in E : u_{ij}^{NSP} > 0} (u_{ij}^{NSP} - u_{ij}). \tag{4.8}$$

By imposing upper bounds on the capacity variable, we limit the search space and, consequently, the computing time, by using the solution found u^{NSP}. At the same time we leave the possibility of installing capacity on edges that were not used in the previous solution. Note that, by imposing upper bounds on the capacities, the proximity search becomes a heuristic method for RND. Given this distance measure definition, we iteratively solve the following proximity search model

$$\min \sum_{\{i,j\}\in E:u_{ij}^{NSP}=0} u_{ij} - \sum_{\{i,j\}\in E:u_{ij}^{NSP}>0} u_{ij} + Mz \tag{4.9}$$

$$\sum_{\{i,j\}\in E} c_{ij}u_{ij} - z\theta \leq c^{NSP} - \theta, \tag{4.10}$$

$$\sum_{j:\{j,i\}\in E} f_{ji}^q - \sum_{j:\{i,j\}\in E} f_{ij}^q = b_i^q \qquad \forall i \in V,\ q = 1,\dots,K \tag{4.11}$$

$$f_{ij}^q + f_{ji}^q \leq u_{ij} \qquad \forall \{i,j\} \in E,\ q = 1,\dots,K \tag{4.12}$$

$$u_{ij} \leq u_{ij}^{NSP} \qquad \forall \{i,j\} \in \overline{E} : u_{ij}^{NSP} > 0 \tag{4.13}$$

$$f_{ij}^q \geq 0 \qquad \forall \{i,j\} \in E,\ q = 1,\dots,K \tag{4.14}$$

$$u_{ij} \in \mathbb{Z}_+ \qquad \forall \{i,j\} \in E \tag{4.15}$$

$$z \in \{0,1\}, \tag{4.16}$$

where the auxiliary variable z is to guarantee feasibility of u^{NSP}. The objective function (4.9) calls for minimizing the distance from the previous solution u^{NSP} and for obtaining a solution with $z = 0$, i.e., an improved solution that respects the cutoff constraint (4.10). Constraint (4.10) imposes to obtain a reduction in the cost of the improved solution of at least θ. Constraints (4.11) and (4.12) correspond to the RND problem constraints. Constraints (4.13) impose the upper bounds on the capacity variables. Finally, constraints (4.14)–(4.16) impose variables' bounds. Note that z is defined as binary as it turned out in our computational experiments that it is very effective to impose branching priority on z, in order to quickly obtain a solution with $z = 0$.

Model (4.9)-(4.16) is solved by Cplex until the first feasible solution with $z = 0$ is obtained. In our experiments θ was set to 1. Therefore, if $z = 1$ the process is stopped. On the first feasible solution found, Cplex *polishing* (see, Rothberg [25]) is applied until the first improved solution is found. Formulation (4.9)–(4.16) is then solved again by replacing u^{NSP} with the improved solution. The proximity search phase is executed until a given time limit is reached. When the time limit is reached, PS returns the best solution found u^{PSP}.

4.5 Computational Results

In this section, we report the computational results that we achieved on instances generated on random graphs with balances $\{-1, 0, 1\}$. Instances are generated as follows: n nodes are randomly located in a unit Euclidean square. Two nodes are connected with an edge if the Euclidean distance is less than $\alpha/\sqrt{n}$ where α is a parameter set to 2 in our generator. The edge cost for capacity installation is proportional to the

Euclidean distance. For each scenario, 25%, 50% or 100% of the nodes are randomly selected to be terminals. We consider 5 or 10 scenarios.

The heuristic was developed in C language, and Cplex version 12.5 with 4 threads was used as a general purpose solver. The tests were executed on a PC 1.73 GHz, 6 GB Ram. The computing times are expressed in seconds. The algorithm CS2 by Goldberg [14] was used for solving the Minimum Cost Flow. The following parameter setting is used for the heuristic: a time limit of 300 seconds is given to NSP and a time limit of 600 seconds is given to PSP. The total time limit for the heuristic is fixed to 900 seconds, because the computation time of the CP is negligible. We fix $F = 100$, $R = 10$, $\theta = 1$ and $M = 100u^{NSP}$, based on parameter tuning.

An important feature of our heuristic algorithm is that it is robust to parameter setting, i.e. the efficacy of the algorithm is not really dependent on the specific parameter values, as long as balances are increased and scenarios are split in sub-scenarios (i.e., $F > 1$ and $R > 1$). In particular, the difference between average gaps, computed with respect to the solutions obtained by Cplex 5h, for different combinations of F and R (with $F > 1$ and $R > 1$) are negligible (below 1%). Among all combinations, the one that has the best balance between gap and standard deviation is given by $R = 10$ and $F = 100$. The other parameter used in our heuristic algorithm is θ. We choose $\theta = 1$ as a conservative value, i.e. a value that allows us to obtain good solutions on average on all the instances. In particular, we observed that, on the small instances (with 50 to 100 nodes), larger values for θ do not produce high quality solutions. When the instances get larger, a more aggressive policy (e.g. with $\theta = 100$) can give better results. We decided to keep a conservative value, in order to avoid parameter overtuning.

In Figure 4.3, we show the results obtained, with a time limit of 900 seconds, by the proposed method after each of the three phases described in Section 4.4. In particular, we show one graphic for each class of instances (from 50 nodes to 500 nodes). In this graphic, the comparison is presented with respect to the solutions obtained after the constructive phase and we show in black the percentage improvement of the solutions obtained after the neighborhood search phase (NSP) and in gray the percentage improvement of the solutions obtained after the proximity search phase (PSP). On the right of the figure, we also show the average and standard deviation for each class of instances. As it can be seen, both the NSP and the PSP are effective in obtaining improvements for all instances but six, on which only PSP is able to improve the solution found u^{CP} after the constructive phase. The detailed results are reported in Table 4.2 in the Appendix.

In the following, we present a comparison of the results obtained by the proposed heuristic (indicated as RND Heur.) with those obtained by Cplex applied to the MIP flow model (4.1)-(4.5) on the original graph G. In particular, we show the results obtained when Cplex is run with a time limit of five hours (Cplex 5h) in default setting, and the results obtained with Cplex in the effective heuristic configurations

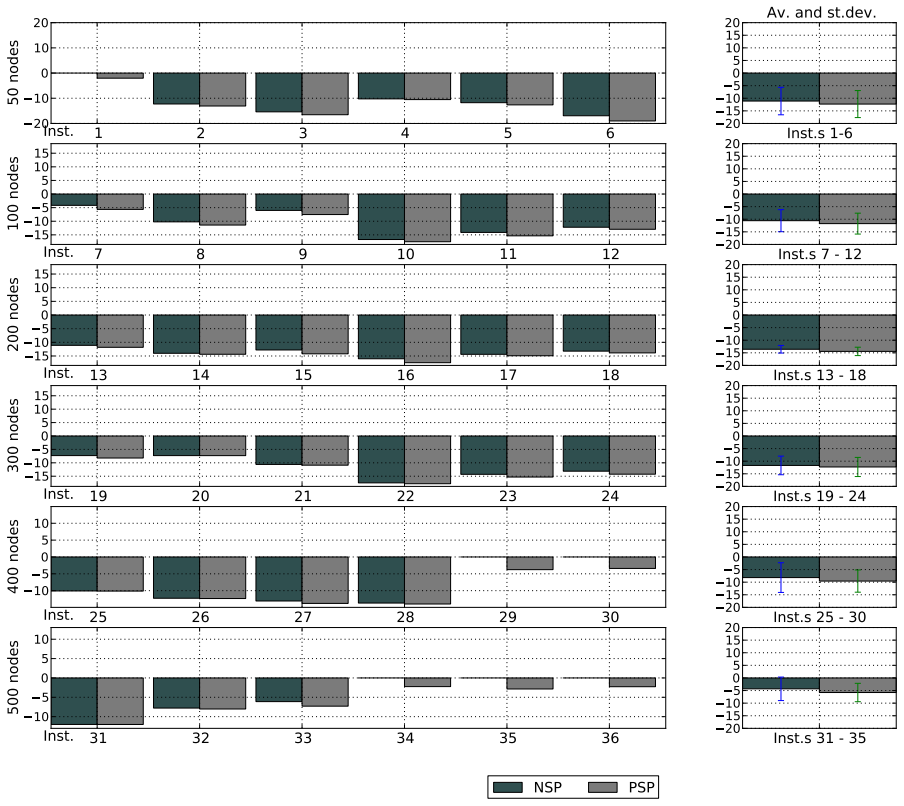

FIGURE 4.3: Comparison of the results obtained after NSP and after PSP with those obtained after CP.

suggested in [13] (Cplex Pol. 900s), i.e., solution *polishing* is applied, and the time limit is set to 900 seconds. The proposed method, Cplex 5h and Cplex Pol. 900s are initialized with the solution u^{CP} constructed as explained in Section 4.4.1.

In Table 4.1, we report the results obtained by Cplex 5h, which will be used as our benchmark for comparison. In particular, we report the data on the instances, the solution (u^{CP}) obtained at the end of the constructive phase and used for initializing each of the methods, the best lower bound (LB) and the best upper bound (UB) obtained by Cplex 5h, the duality gap (Gap%), the number of branch and bound nodes (BBn), and the computing time. As it can be seen from Table 4.1, the time limit is reached for all instances but the three smallest ones for which Cplex is able to prove optimality. For the remaining instances, the duality gaps are often quite large and for seven instances Cplex is not even able to improve the initial solution.

In order to measure the performance of the proposed method, we show in Figure 4.4 the comparison between the results we obtain in 900 seconds of time limit and the results obtained by Cplex 5h and by Cplex Pol. 900s. The detailed results are reported in Table 4.3 in the Appendix. In particular, we show one graphic for each class of instances (from 50 nodes to 500 nodes). In this graphic, the comparison is presented

					Cplex 5h				
Inst.	n	$t\%$	K	u^{CP}	LB	UB	Gap%	BBn	Time
1	50	25	5	22904	22438	22438	0.00	1977	6
2	50	50	5	60921	52947	52952	0.01	602223	2666
3	50	100	5	79487	66334	66340	0.01	968741	4634
4	50	25	10	52835	44129	47272	6.65	464679	18000
5	50	50	10	66781	55277	57861	4.47	374536	18000
6	50	100	10	88323	70085	71526	2.01	544735	18000
7	100	25	5	39255	36741	37031	0.78	960936	18000
8	100	50	5	89264	76498	78702	2.80	430334	18000
9	100	100	5	81126	74331	74822	0.66	675507	18000
10	100	25	10	86929	67148	71192	5.68	63827	18000
11	100	50	10	115437	92265	97246	5.12	44262	18000
12	100	100	10	132233	112062	114624	2.24	76558	18000
13	200	25	5	98497	77288	85676	9.79	67461	18000
14	200	50	5	142509	113158	122062	7.29	53440	18000
15	200	100	5	169962	139302	144508	3.60	75979	18000
16	200	25	10	134999	98358	114995	14.47	11630	18000
17	200	50	10	173335	133819	148087	9.63	9406	18000
18	200	100	10	219903	175660	184992	5.04	14684	18000
19	300	25	5	92259	73805	83302	11.40	28125	18000
20	300	50	5	139954	115164	128296	10.24	22212	18000
21	300	100	5	183689	150048	162860	7.87	22284	18000
22	300	25	10	148349	103953	148349	29.93	2927	18000
23	300	50	10	201301	151200	199456	24.19	2198	18000
24	300	100	10	271340	214577	268072	19.96	2603	18000
25	400	25	5	109241	87877	98297	10.60	14881	18000
26	400	50	5	217300	175286	190328	7.90	12671	18000
27	400	100	5	291469	234266	252987	7.40	18336	18000
28	400	25	10	158033	117143	158033	25.87	1191	18000
29	400	50	10	253648	191242	253648	24.60	1239	18000
30	400	100	10	325512	255769	325512	21.43	1278	18000
31	500	25	5	106191	75197	98778	23.87	7576	18000
32	500	50	5	189269	159465	177572	10.20	10186	18000
33	500	100	5	261922	216832	241684	10.28	8584	18000
34	500	25	10	214149	153247	214149	28.44	325	18000
35	500	50	10	262379	196930	262379	24.94	315	18000
36	500	100	10	323275	249201	323275	22.91	564	18000

TABLE 4.1: Results obtained with Cplex on the MIP flow formulation in five hours of time limit.

with respect to the solutions obtained by Cplex 5h and we show in black the percentage gap of the solutions obtained by Cplex Pol. 900s and in gray the percentage gap of the solutions obtained by RND Heur. On the right of the figure, we also show the average and standard deviation for each class of instances.

As it is evident from Figure 4.4, the three methods obtain comparable results for instances with up to 100 nodes. However, as the instances get larger, the proposed method becomes more effective than the other ones, and it is able to improve the results obtained by the other two methods. In particular, compared to Cplex Pol. 900s that has the same time limit, the proposed method always obtains better solutions for

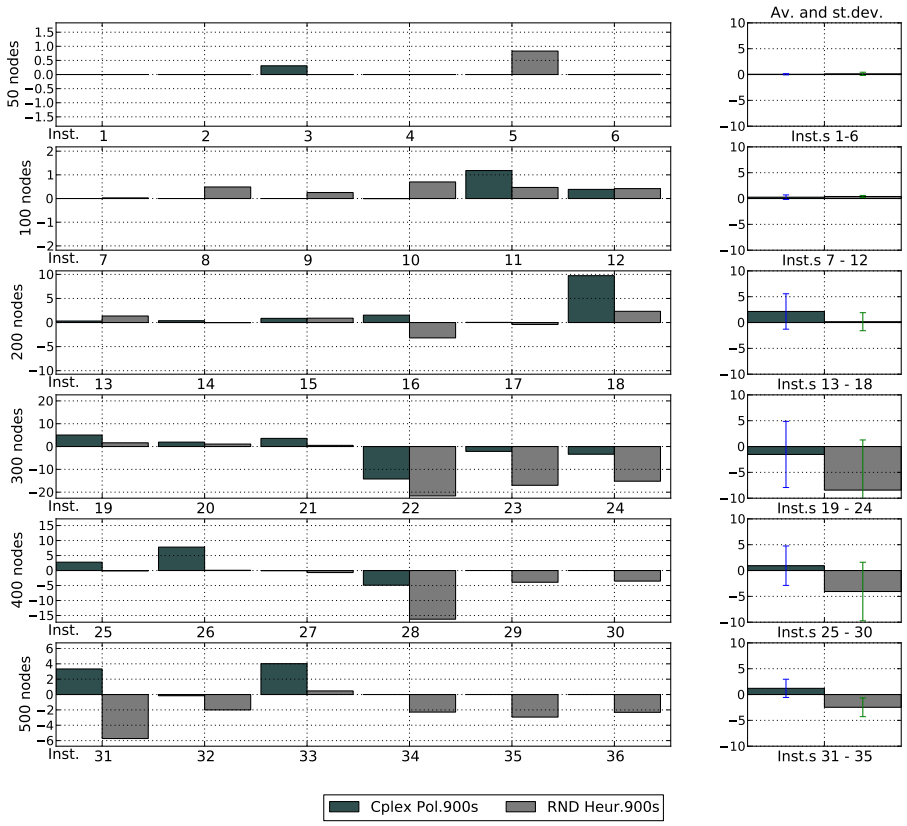

FIGURE 4.4: Comparison of the proposed heuristic RND (time limit of 15 minutes) with Cplex Pol. 900s and with Cplex 5h.

instances with at least 300 nodes. It obtains solutions with a cost less or equal than those obtained by Cplex Pol. 900s for 27 out of 36 instances, and is at most 1.05% worse for a single instance. The improvement is significant (between 3% and more than 14%) for 14 out of 36 instances. Even compared to Cplex run for five hours, the proposed method performs on average better on instances with at least 200 nodes, especially when we have 10 scenarios. It is able to obtain better or equal solutions for 20 out of 36 instances. The average percentage improvement with respect to Cplex 5h and Cplex Pol. 900s is 2.38% and 2.90%, respectively.

In order to further validate the results presented in Figure 4.4, we performed extensive computational experiments on instances with $n = 300$ and $n = 400$ nodes. In particular, we considered five instances for each sub-class, defined by selecting $t \in \{25\%, 50\%, 100\%\}$ and $k \in \{5, 10\}$. This gives a total testbed of 60 instances. In Figure 4.5, we show the comparison between the three methods. The comparison is presented with respect to the solutions obtained by Cplex in five hours. We show in black the percentage gap of the solutions obtained by Cplex Pol. 900s and in gray the percentage gap of the solutions obtained by RND Heur in 900s. Compared to Cplex Pol. 900s, that has the same time limit, the proposed method always obtains

better solutions, and, compared to Cplex 5h, performs better on all instances with 10 scenarios, confirming the effectiveness of the proposed approach.

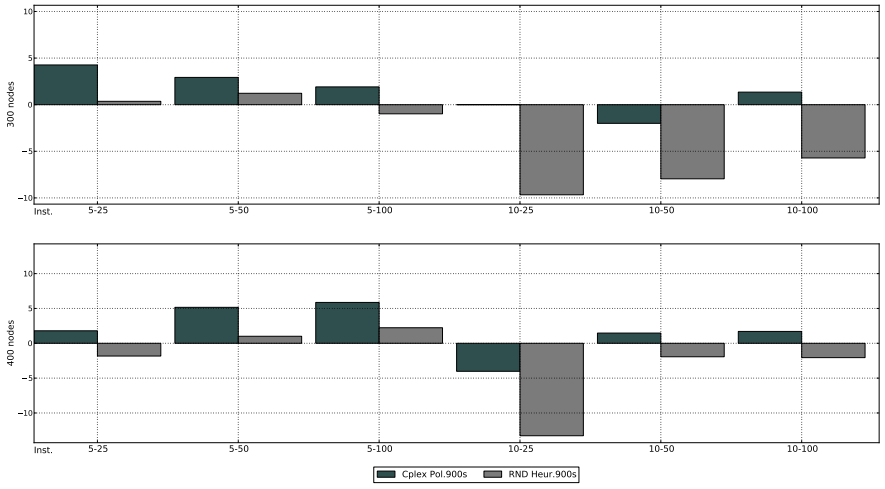

FIGURE 4.5: Comparison of the three methods on additional instances with 300 and 400 nodes.

4.6 Conclusions and Future Research

We have presented a single-commodity robust network design problem and we have shown complexity results for special classes of instances, including hypercubes. By the complexity analysis, we have shown that instances with random integer balances different from 1 and -1 are NP-hard, even if computationally easy ([9, 2]). In order to explain why, we have shown that instances defined on hypercubes with balances in $\{-r, 0, r\}$ (r integer, $r > 1$) are theoretically easy, while instances defined on hypercubes with balances in $\{-1, 0, 1\}$ are structurally hard. This has motivated us to study instances (defined on random graphs) with balances of $\{-1, 0, 1\}$. We have developed a heuristic algorithm composed of three phases.

The first one reduces the instance graph and constructs a feasible solution, the second one solves an MIP flow-formulation of the problem on the reduced graph for a given time limit, in order to improve the solution found, and the last phase applies a modified version of the recent technique of proximity search to further improve the solution. We have tested the proposed method on randomly generated instances with balances of $\{-1, 0, 1\}$, and we have compared the obtained results with those obtained by Cplex both in 5 hours (default version) or by using the *polishing* algorithm to enhance its heuristic behavior (for 900 seconds). The results show that our method is comparable with the other ones for instances with up to 100 nodes, but obtains better solutions for larger instances. Future research can be devoted to extend the proposed algorithm

to the multi-commodity case. In addition, the proposed method takes into account the balances of all the scenarios, but a less conservative approach could be considered, for example, by taking into account the probability of each scenario. Other extensions could be to tackle related variants of robust network design, such as Survivable Network Design: mostly the constructive phase needs to be modified, as long as a good MIP formulation exists. Additional parameter tuning might be necessary as well.

Appendix

Proof of Lemma 7

We now provide a full version of the proof of Lemma 7. Let us denote by V^d and E^d the set of nodes and edges of $\mathcal{H}_d$, respectively.

Proof of Lemma 7. If we define the set

$$\mathcal{S} := \left\{ S \subset V^d \mid S \text{ is connected and separates at least one } v_q \text{ from its partner } v_q^o \right\},$$

we can find an optimum fractional solution for $\mathcal{H}_d^r$ with the following linear program [2].

$$\begin{aligned} \min \sum_{e \in E^d} u_e & \\ \sum_{e \in \delta(S)} u_e \geq r & \quad \text{for all } S \in \mathcal{S} \\ u_e \geq 0 & \quad \text{for all } e \in E \end{aligned} \tag{CAP}$$

If $d = 2$, it holds that $|S| = d = 2$ for all $S \in \mathcal{S}$. Consequently, if we set $u_e = r/2$ for all $e \in E^d$, all primal constraints are satisfied with equality and the solution is optimal. If $d \geq 3$, we introduce dual variables ξ_S for all $S \in \mathcal{S}$ and obtain the following dual program:

$$\begin{aligned} \max \sum_{S \in \mathcal{S}} r \cdot \xi_S & \\ \sum_{\substack{S \in \mathcal{S}: \\ \{i,j\} \in S}} \xi_S \leq 1 & \quad \text{for all } \{i,j\} \in E^d \\ \xi_S \geq 0 & \end{aligned} \tag{CAP^*}$$

We consider the following pair of primal and dual solutions:

$$u_e := r/d \qquad \text{for all } e \in E^d \qquad\qquad \xi_S := \begin{cases} 0, & \text{if } |\delta(S)| > d \\ 1/2, & \text{if } |\delta(S)| = d \end{cases} \qquad \text{for all } S \in \mathcal{S}.$$

To prove our claim, we need to show that u and ξ are feasible and satisfy complementary slackness. Feasibility of u follows by the first part of Lemma 9: For all $S \in \mathcal{S}$, we have $|\delta(S)| \geq d$ and thus $\sum_{e\in\delta(S)} u_e = |\delta(S)|(r/d) \geq r$. Observe that by the second part of Lemma 9 equality holds if and only if $|\delta(S)| = d$. Thus, we have $(\sum_{e\in\delta(S)} u_e - r)\cdot\xi_S = 0$ for all $S \in \mathcal{S}$, yielding primal complementary slackness. To see why ξ is feasible for (CAP^*) we need to show that

$$\sum_{\substack{S\in\mathcal{S}:\\ \{i,j\}\in S}} \xi_S = \sum_{\substack{S\in\mathcal{S}:\\ |\delta(S)|=d\\ \{i,j\}\in S}} \xi_S \leq 1 \qquad \text{for all } \{i,j\} \in E^d.$$

By applying Lemma 9, we can rewrite this as

$$\sum_{\substack{S\in\mathcal{S}:\\ |\delta(S)|=d\\ \{i,j\}\in S}} \xi_S = \sum_{\substack{S\in\mathcal{S}:\\ |S|=1\\ \{i,j\}\in S}} \xi_S = \xi_{\{i\}} + \xi_{\{j\}} = 1 \qquad \text{for all } \{i,j\} \in E^d$$

which also yields that $(\sum_{S\in\mathcal{S}:e\in S} \xi_S - 1) \cdot u_e = 0$ for all $e \in E^d$, i.e., we have dual complementary slackness. Finally, both solutions yield the desired objective value of $\sum_{e\in E^d} r/d = d \cdot 2^{d-1} \cdot (r/d) = r \cdot 2^{d-1}$. □

The following lemma provides the missing piece for the above proof.

Lemma 9. *Let $d \geq 3$. Then in $\mathcal{H}_d$, $|\delta(S)| \geq d$ for all $\emptyset \subsetneq S \subsetneq V^d$. Moreover, equality is attained if and only if $|S| = 1$ or $|S| = |V^d| - 1$.*

Proof. The first part of the lemma is well-known: Saad and Schultz [26, Propositions 3.2 and 3.3] proved that for any two nodes i, j of a d-dimensional hypercube, there are at least d node disjoint paths between i and j. By Menger's Theorem [22], this implies that $|\delta(S)| \geq d$ for all $\emptyset \subsetneq S \subsetneq V^d$. Also, if S contains a single node i, then $|\delta(S)| = |\delta(i)| = d$. It remains to show that the inequality is strict if $2 \leq |S| \leq |V^d| - 2$. Without loss of generality, we can assume that $|S| \leq \frac{1}{2}|V^d|$ since $\delta(S) = \delta(V \setminus S)$.

Now, choose an arbitrary decomposition of $\mathcal{H}_d$ into two $(d-1)$-dimensional hypercubes $H_1 = (V_1, E_1), H_2 = (V_2, E_2)$ such that neither of $S_1 := S \cap V_1$ and $S_2 := S \cap V_2$ is empty. This is possible because S contains at least two and at most $|V|/2$ nodes. It also implies that neither $S_1 = V_1$ nor $S_2 = V_2$, as otherwise S_2 or S_1 would be empty, respectively.

For $i = 1, 2$, the node set S_i defines a cut $\delta_i(S_i)$ in H_i. Since $S_i \neq \emptyset$ and $S_i \neq V_i$, we know that $|\delta_i(S_i)| \geq d - 1$, since H_i is a $(d-1)$-dimensional hypercube. Also, $\delta_1(S_1), \delta_2(S_2) \subseteq \delta(S)$ and therefore $|\delta(S)| \geq 2 \cdot (d-1) > d$ for $d \geq 3$. □

Tables

In Table 4.2, we show the results obtained by the proposed method after each of the three phases described in Section 4.4. In particular, we show the instance name (Inst.), the number n of nodes in the graph G, the percentage $t\%$ of nodes that are terminals, the number K of considered scenarios, the solution (u^{CP}) obtained at the end of the constructive phase, the solution u^{NSP} obtained after the neighborhood search phase (and the corresponding percentage improvement $Impr_{u^{CP}}\%$ with respect to u^{CP}) and the final solution u^{PSP} provided by our method by applying proximity search (and the corresponding percentage improvement $Impr_{u^{NSP}}\%$ with respect to u^{CP}). We do not report the computing times, as the time limit of 900 seconds is reached for all instances.

In Table 4.3, we report the value of the best solution obtained by each method, and, for Cplex Pol. 900s and for RND Heur., we show the percentage gap $\text{Gap}_{C^{5h}}\%$ to the best upper bound computed by Cplex 5h. In the last column, we also show the percentage gap $\text{Gap}_{C^{900s}}\%$ between the solutions obtained by RND Heur. and Cplex Pol. 900s. Finally, in the last rows of the table, we show the average (Avg.), the median (Median) and the standard deviation (StDev.) of the percentage gaps, as well as the minimum (Min) and the maximum (Max) percentage gap.

Inst.	n	$t\%$	K	u^{CP}	u^{NSP}	$Impr_{u^{CP}}\%$	u^{PSP}	$Impr_{u^{NSP}}\%$
1	50	25	5	22904	22904	0.00	22438	-2.03
2	50	50	5	60921	53443	-12.27	52952	-13.08
3	50	100	5	79487	67250	-15.39	66340	-16.54
4	50	25	10	52835	47419	-10.25	47272	-10.53
5	50	50	10	66781	58928	-11.76	58346	-12.63
6	50	100	10	88323	73352	-16.95	71530	-19.01
7	100	25	5	39255	37624	-4.15	37041	-5.64
8	100	50	5	89264	80139	-10.22	79088	-11.40
9	100	100	5	81126	76247	-6.01	75012	-7.54
10	100	25	10	86929	72399	-16.71	71694	-17.53
11	100	50	10	115437	99155	-14.10	97703	-15.36
12	100	100	10	132233	116100	-12.20	115107	-12.95
13	200	25	5	98497	87562	-11.10	86855	-11.82
14	200	50	5	142509	122543	-14.01	122032	-14.37
15	200	100	5	169962	148214	-12.80	145826	-14.20
16	200	25	10	134999	113380	-16.01	111439	-17.45
17	200	50	10	173335	148397	-14.39	147487	-14.91
18	200	100	10	219903	190824	-13.22	189406	-13.87
19	300	25	5	92259	85518	-7.31	84681	-8.21
20	300	50	5	139954	129723	-7.31	129709	-7.32
21	300	100	5	183689	164206	-10.61	163699	-10.88
22	300	25	10	148349	122424	-17.48	121953	-17.79
23	300	50	10	201301	172539	-14.29	170487	-15.31
24	300	100	10	271340	235728	-13.12	232706	-14.24
25	400	25	5	109241	98219	-10.09	98176	-10.13
26	400	50	5	217300	190677	-12.25	190492	-12.34
27	400	100	5	291469	253378	-13.07	251291	-13.78
28	400	25	10	158033	136413	-13.68	135968	-13.96
29	400	50	10	253648	253648	0.00	244109	-3.76
30	400	100	10	325512	325512	0.00	314428	-3.41
31	500	25	5	106191	93433	-12.01	93425	-12.02
32	500	50	5	189269	174540	-7.78	174082	-8.02
33	500	100	5	261922	245907	-6.11	242828	-7.29
34	500	25	10	214149	214149	0.00	209360	-2.24
35	500	50	10	262379	262379	0.00	254891	-2.85
36	500	100	10	323275	323275	0.00	315955	-2.26
Avg.						-9.91		-11.02

TABLE 4.2: Results obtained by the proposed method within 900 seconds of time limit.

				Cplex 5h	Cplex Pol. 900s		RND Heur. 900s		
Inst.	n	$t\%$	K	UB	UB	$\text{Gap}_{C^{5h}}\%$	u^{PSP}	$\text{Gap}_{C^{5h}}\%$	$\text{Gap}_{C^{900s}}\%$
1	50	25	5	22438	22438	0.00	22438	0.00	0.00
2	50	50	5	52952	52952	0.00	52952	0.00	0.00
3	50	100	5	66340	66546	0.31	66340	0.00	-0.31
4	50	25	10	47272	47272	0.00	47272	0.00	0.00
5	50	50	10	57861	57861	0.00	58346	0.83	0.83
6	50	100	10	71526	71526	0.00	71530	0.01	0.01
7	100	25	5	37031	37031	0.00	37041	0.03	0.03
8	100	50	5	78702	78702	0.00	79088	0.49	0.49
9	100	100	5	74822	74822	0.00	75012	0.25	0.25
10	100	25	10	71192	71189	0.00	71694	0.70	0.70
11	100	50	10	97246	98409	1.18	97703	0.47	-0.72
12	100	100	10	114624	115068	0.39	115107	0.42	0.03
13	200	25	5	85676	85947	0.32	86855	1.36	1.05
14	200	50	5	122062	122522	0.38	122032	-0.02	-0.40
15	200	100	5	144508	145770	0.87	145826	0.90	0.04
16	200	25	10	114995	116786	1.53	111439	-3.19	-4.80
17	200	50	10	148087	148138	0.03	147487	-0.41	-0.44
18	200	100	10	184992	204936	9.73	189406	2.33	-8.20
19	300	25	5	83302	87723	5.04	84681	1.63	-3.59
20	300	50	5	128296	130825	1.93	129709	1.09	-0.86
21	300	100	5	162860	168882	3.57	163699	0.51	-3.17
22	300	25	10	148349	129877	-14.22	121953	-21.64	-6.50
23	300	50	10	199456	195300	-2.13	170487	-16.99	-14.55
24	300	100	10	268072	259317	-3.38	232706	-15.20	-11.44
25	400	25	5	98297	101115	2.79	98176	-0.12	-2.99
26	400	50	5	190328	206445	7.81	190492	0.09	-8.37
27	400	100	5	252987	252842	-0.06	251291	-0.67	-0.62
28	400	25	10	158033	150661	-4.89	135968	-16.23	-10.81
29	400	50	10	253648	253648	0.00	244109	-3.91	-3.91
30	400	100	10	325512	325512	0.00	314428	-3.53	-3.53
31	500	25	5	98778	102182	3.33	93425	-5.73	-9.37
32	500	50	5	177572	177292	-0.16	174082	-2.00	-1.84
33	500	100	5	241684	251807	4.02	242828	0.47	-3.70
34	500	25	10	214149	214149	0.00	209360	-2.29	-2.29
35	500	50	10	262379	262379	0.00	254891	-2.94	-2.94
36	500	100	10	323275	323275	0.00	315955	-2.32	-2.32
Avg.						0.51		-2.38	-2.90
Median						0.00		0.00	-1.35
StDev.						3.67		5.75	3.96
Min						-14.22		-21.64	-14.55
Max						9.73		2.33	1.05

TABLE 4.3: Comparison of the proposed heuristic (time limit of 15 minutes) with Cplex (time limit of 5 hours or 15 minutes).

Bibliography

[1] A. Altin, E. Amaldi, P. Belotti, and M.C. Pinar. Provisioning virtual private networks under traffic uncertainty. *Networks*, 49:100–115, 2007.

[2] E. Álvarez-Miranda, V. Cacchiani, T. Dorneth, M. Jünger, F. Liers, A. Lodi, T. Parriani, and D. Schmidt. Models and algorithms for robust network design with several traffic scenarios. In A. R. Mahjoub et al., editor, *ISCO 2012*, volume 7422 of *Lecture Notes in Computer Science*, pages 261–272. Springer, 2012.

[3] A. Atamtürk. On capacitated network design cut-set polyhedra. *Mathematical Programming*, 92:425–437, 2000.

[4] W. Ben-Ameur and H. Kerivin. Routing of uncertain traffic demands. *Optimization and Engineering*, 6:283–313, 2005.

[5] A. Ben-Tal and A. Nemirovski. Robust solutions of uncertain linear programs. *Operations Research Letters*, 25(1):1–13, 1999.

[6] A. Ben-Tal and A. Nemirovski. Robust solutions of linear programming problems contaminated with uncertain data. *Mathematical Programming*, 88:411–424, 2000.

[7] D. Bertsimas and M. Sim. The price of robustness. *Operations Research*, 52(1):35–53, 2004.

[8] D. Bienstock, S. Chopra, O. Günlük, and C.H. Tsai. Minimum cost capacity installation for multicommodity network flows. *Mathematical Programming*, 81(2):177–199, 1998.

[9] C. Buchheim, F. Liers, and L. Sanità. An exact algorithm for robust network design. In J. Pahl, T. Reiners, and S. Voß, editors, *INOC*, volume 6701 of *Lecture Notes in Computer Science*, pages 7–17. Springer, 2011.

[10] C. Chekuri. Routing and network design with robustness to changing or uncertain traffic demands. *SIGACT News*, 38:106–128, 2007.

[11] R. T. Chien. Synthesis of a communication net. *IBM Journal of Research and Development*, 4(3):311–320, 1960.

[12] 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 *Proceedings of the conference on Applications, technologies, architectures, and protocols for computer communication*, SIGCOMM '99, pages 95–108. ACM, 1999.

[13] M. Fischetti and M. Monaci. Proximity search for 0-1 mixed-integer convex programming. Technical report, DEI, University of Padova, Italy, 2013.

[14] A.V. Goldberg. An efficient implementation of a scaling minimum-cost flow algorithm. *Journal of Algorithms*, 22(1):1–29, 1997.

[15] R. Gomory and T. Hu. An application of generalized linear programming to network flows. *Journal of the Society for Industrial and Applied Mathematics*, 10(2):260–283, 1962.

[16] R.E Gomory and T.C. Hu. Multi-terminal network flow. *SIAM Journal on Applied Mathematics*, 9:551–570, 1961.

[17] L. R. Ford Jr. and D. R. Fulkerson. A simple algorithm for finding maximal network flows and an application to the hitchcock problem. *Canadian Journal of Mathematics*, 9:210–218, 1957.

[18] H. Kerivin and A.R. Mahjoub. Design of survivable networks: A survey. In *In Networks*, pages 1–21, 2005.

[19] A.M.C.A. Koster, M. Kutschka, and C. Raack. Robust network design: Formulations, valid inequalities, and computations. *Networks*, 61(2):128–149, 2013.

[20] M. Labbé, R. Séguin, P. Soriano, and C. Wynants. Network synthesis with non-simultaneous multicommodity flow requirements: Bounds and heuristics, 1999.

[21] S. Mattia. The robust network loading problem with dynamic routing. *Computational Optimization and Applications*, 54:619–643, 2013.

[22] Karl Menger. Zur allgemeinen Kurventheorie. *Fundamenta Mathematicae*, 10(1):96–115, 1927.

[23] M. Minoux. Optimum synthesis of a network with non-simultaneous multicommodity flow requirements. In *Annals of Discrete Mathematics (11) Studies on Graphs and Discrete Programming*, volume 59, pages 269 – 277. North-Holland, 1981.

[24] F. Ortega and L.A. Wolsey. A branch-and-cut algorithm for the single-commodity, uncapacitated, fixed-charge network flow problem. *Networks*, 41(3):143–158, 2003.

[25] E. Rothberg. An evolutionary algorithm for polishing mixed integer programming solutions. *INFORMS Journal on Computing*, 19:534–541, 2007.

[26] Youcef Saad and Martin H. Schultz. Topological properties of hypercubes. Technical Report YALEU/DCS/TR389, Yale University, 1985.

[27] L. Sanità. *Robust Network Design*. Ph.D. Thesis. Università La Sapienza, Roma, 2009.

[28] L. Sanità. Private communication. 2013.

[29] A. L. Soyster. Convex programming with set-inclusive constraints and applications to inexact linear programming. *Operations Research*, 21(5):1154–1157, 1973.

Chapter 5

A Decomposition Based Heuristical Approach for Solving Large Stochastic Unit Commitment Problems Arising in a California ISO Planning Model[1]

5.1 Introduction

The state of California has embarked on an aggressive plan to produce 33% of its electric energy from renewable resources by the year 2020 [3]. The increased penetration of intermittent renewable generation needed to meet this goal will substantially increase the variability and uncertainty in generation resources available to system operators. To assess the impact of these high renewable penetrations, the California Energy Commission funded a recently-completed study at Lawrence Livermore National Laboratory to couple atmospheric models capable of producing renewable generation trajectories with stochastic day-ahead unit commitment optimization models [9]. This stochastic day-ahead unit commitment model employs at its core a deterministic unit commitment planning model developed by the California Independent System Operator (ISO)

[1] **Accepted** - "An Efficient Approach for Solving Large Stochastic Unit Commitment Problems Arising in a California ISO Planning Model", T. Parriani, G. Cong, C. Meyers, D. Rajan, *2014 PES General Meeting Proceedings*; **Under review** - "*A framework for solving mixed-integer programming problems with decomposition and generic approaches*", G. Cong, T. Parriani, United States Patent and Trademark Office, Research Disclosure YOR8-2013-1124.

for a study of market impacts under the 33% renewable portfolio standard [2]. The deterministic model is based on a grid description and operational specifications provided by the California ISO, and is implemented using the PLEXOS power market software package [10], which generates a mixed-integer linear programming formulation suitable for determining day-ahead hourly commitment decisions.

The stochastic unit commitment model is formulated as a two-stage mixed-integer stochastic optimization extension of the deterministic model, where scenarios are defined by different renewable generation trajectories, unit commitment states for long-start generators are treated as first-stage decisions (common across all scenarios), and economic dispatch values and unit commitment states for short-start generators are treated as second-stage decisions (one for each scenario).

5.1.1 Model Description and Prior Computational Performance

The model representation of the Western Energy Coordinating Council (WECC) grid developed by the California ISO includes more than 2,400 generating units, over 42 zones in 11 states, with 120 transmission lines between zones (a zonal model). Wind and solar (PV) inputs are included at a zonal level, aggregated from numerous individual real and proposed wind and solar sites. The California ISO model calculates hourly day-ahead unit commitments for all 2,400 generating units, with integer commitments for all (several hundred) generation units in California, and fractional commitments elsewhere. As a result, the deterministic day-ahead mixed-integer program (MIP) is already fairly large, including roughly 400,000 constraints, 600,000 continuous variables, 10,000 general integer variables and 2,000 binary variables.

Variables and constraints in the stochastic version of the model are (roughly) linear multiples of the corresponding values in the deterministic model, according to the number of scenarios used. Thus, the computational burden associated with solving such problems becomes prohibitive for even very powerful systems. In the California Energy Commission study [9], it was found that for only 8 scenarios, each day-ahead stochastic unit commitment problem already takes an average of 5 hours to solve, and no solutions at all were found for 20 or more scenarios. It was for this reason the original study was downscaled to include only 5 scenarios [9].

5.1.2 Contributions

We propose in Section 5.3 a new method for solving two-stage stochastic optimization problems that significantly reduces the solution time of the aforementioned stochastic unit commitment problems. As detailed in Section 5.4, this method allows us to solve individual unit commitment problems roughly 6 times faster than previously

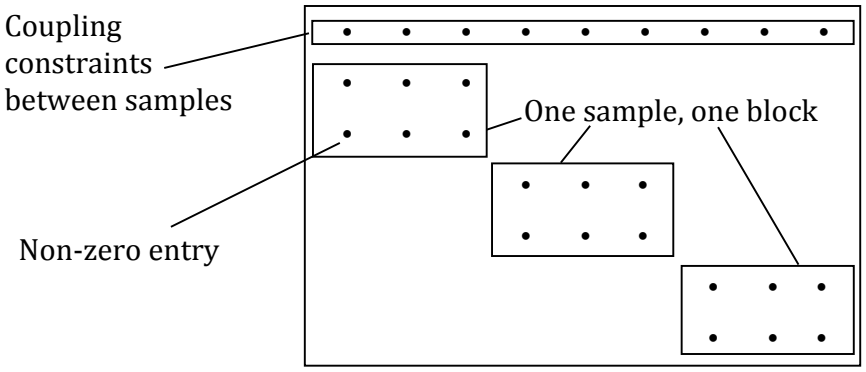

FIGURE 5.1: Pictorial Representation of non-zero entries in constraint matrix: Coupling constraints (5.2) have first-stage variables from all scenarios. Constraints (5.3) and (5.4) can be divided into blocks, one for each scenario s.

achievable, while a parallel implementation of this method (many of its stages are parallelizable) solves the problems more than 20 times faster.

5.2 Dual Decomposable Schemes for Stochastic Optimization Problems

We begin by reviewing stochastic optimization problems, and briefly describe schemes for solving such problems. We then consider the Progressive Hedging (PH) algorithm in detail, highlighting many of its strengths and some weaknesses.

5.2.1 Stochastic Optimization: An overview

A stochastic optimization problem is often formulated as a two-stage optimization problem. Given a set of scenarios $\omega_s, s \in [1, S]$ with corresponding probabilities p_s, the sample-based MIP is formulated as:

$$\min \quad \sum_{s \in S} p_s \left(c^T x_s + q_s^T y_s \right) \tag{5.1}$$

$s.t.$

$$x_1 = x_2 = \cdots = x_s \tag{5.2}$$

$$Ax_s \leq b \tag{5.3}$$

$$T_s x_s + W_s y_s = h_s \tag{5.4}$$

$$l_1 \quad \leq x_s \leq u_1 \tag{5.5}$$

$$l_2 \quad \leq y_s \leq u_2 \tag{5.6}$$

$$x_s^j \in \mathbb{Z}, \forall j \in D_1 \subseteq \{1, \ldots, n_1\} \tag{5.7}$$

$$y_s^j \in \mathbb{Z}, \forall j \in D_2 \subseteq \{1, \ldots, n_1\} \tag{5.8}$$

where x and y are first- and second-stage decision variables, respectively.

This formulation is often referred to as dual decomposable, since by eliminating the coupling constraints (1a), the problem can be decomposed into a separate subproblem for each scenario; see Figure 5.1. Many dual decomposition approaches have been applied to such problems, including Lagrangian [4], and augmented Lagrangian methods [1, 22], branch and price approaches based on Dantzig-Wolfe decomposition [16], and the Progressive Hedging (PH) algorithm proposed in [19]. The Dantzig-Wolfe decomposition requires the iterative resolution of an integer reduced master problem. Conversely, the augmented lagrangian method is not naturally decomposable and requires the use of some "tricks" (see e.g., [5, 21]). An alternate deterministic equivalent formulation (also expanded by the number of scenarios) uses coupling first-stage variables that are common for all the scenario. In this primal decomposable formulation, once the coupling first-stage variable are eliminated, the problem decomposes into separate subproblems corresponding to scenarios. Using primal-decomposition schemes such as Benders' (see, e.g., [29]), stochastic optimization problems can be effectively solved when all the second-stage variables are continuous. When the problem has general integer and binary second-stage variables, as the problem addressed here, it is not possible to directly use the Benders' scheme. Extensions have been proposed to primal decomposition schemes for such cases, (see e.g., [30, 23, 24, 11]).

Progressive Hedging (PH) was designed specifically for stochastic programming problems and combines the idea of augmented Lagrangian methods with a scenario based decomposition. Unlike branch and price and other approaches based on column generation, the subproblems in PH are updated without needing to iteratively solve a master problem.

5.2.2 The Progressive Hedging Algorithm

Studies concerning the structural proprieties of PH can be found in [26] while in [25] a first analysis of different parallelization strategies is presented. For convex optimization problems, such as stochastic **linear** programs, PH is guaranteed to converge to the optimal solution ([13, 17]), even if the subproblems are solved approximately ([28, 19]). In our case, both since both first and second stage decisions contain integer variables, theoretical convergence is lost. Nevertheless, PH can be applied to non-convex stochastic integer programs (such as ours) to obtain heuristic solutions ([7, 12, 15, 14, 27]).

In PH, for iteration i, a subproblem is defined for each scenario s as:

$$(SP_s^i) \qquad \min p_s(c^T x_s + q_s^T y_s) + \hat{f}_s'(x_s, i) \tag{5.2a}$$

$$Ax_s + = b \tag{5.2b}$$

$$T_s x_s + W_s y_s = h_s \tag{5.2c}$$

$$x \in X, y \in Y \qquad . \tag{5.2d}$$

The *penalty function* $\hat{f}_s(x_s, i)$ is then defined as:

$$\hat{f}_s'(x_s, i) = \lambda_s^i x_s + \frac{1}{2}\rho^i (x_s - \overline{x}^i)^2 \quad \forall i > 0 \tag{5.3}$$

$$\hat{f}_s(x_s, i) = 0 \quad i = 0 \tag{5.4}$$

where $\rho^i = \rho > 0 \ \forall i$ is a parameter we refer to as "*penalty factor*"; $\overline{x}$ is a vector of the averages of subproblem solutions, x_s^{*i-1}, defined as $\overline{x}^i = \sum_{s \in S} p_s x_s^{*i-1}$ and λ_s^i is defined as

$$\lambda_s^i = \lambda_s^{i-1} + \rho^{i-1}(x_s^{*i-1} - \overline{x}^i) \quad \forall i > 0 \tag{5.5}$$

$$\lambda_s^i = 0 \quad i = 0 \quad . \tag{5.6}$$

At every iteration of the algorithm, SP_s^i is solved for every scenario and solutions are used to update the penalty function. The purpose is to define a penalty function so that all first-stage variables assume the same value in all subproblem solutions, producing a feasible solution (optimal in the convex case).

A natural stopping criterion for PH consists in halting execution when convergence is reached for all the first-stage variables. In a commonly used termination criterion [8], PH terminates when the norm

$$\delta = \{||\overline{x}^i - \overline{x}^{i-1}||^2 + \sum_{s \in S} p_s ||x^{*i,s} - \overline{x}^i||^2\}^{\frac{1}{2}}$$

drops below a certain parametric threshold. In this case δ is a measure of the "distance from convergence". As we discuss later, we also consider termination criteria based on the gap between the best-known upper bound (from feasible solutions) and the best-known lower bound (obtained by combining the solutions of the subproblems for the first iteration). In practice, many problems also consider a global execution time limit.

The main drawbacks of PH are that the commonly used penalty factor results in a quadratic integer program for each subproblem, which can present computational challenges; furthermore, the PH scheme does not provide feasible solutions to the stochastic problem until it converges (or at least all the integer-first stage variables converge). Even then, there is no guarantee that a feasible solution exists for the original problem with fixed first-stage integer variables. We address both these shortcomings in

mPH, our modified PH-based algorithm for solving two-stage stochastic optimization problems.

5.3 mPH: A PH-based Heuristic for Two-stage Stochastic Problems

We motivate our introduction of mPH, a modified version of PH effective in solving large real-world stochastic problems, by detailing our approach in overcoming the drawbacks of PH.

5.3.1 Linear subproblems

The penalty function defined in (5.3)-(5.4) leads to quadratic subproblems. (If all the second-stage variables are defined as binaries there is an equivalent linear formulation for SP_s^i [7], but this does not apply in our unit commitment problems.) When the size of the problem increases, the quadratic subproblems quickly become difficult to solve if not intractable. From a theoretical point of view, the presence of the quadratic term ensures convergence to an optimal solution for the convex case (as in stochastic linear programs). From our perspective, losing the convergence property by modifying the penalty function is acceptable since the proof of convergence does not extend to the non-convex stochastic integer program case.
As illustrated in Figure 5.2(a), the quadratic term in $\hat{f}'_s(x_s, i)$ pushes the first-stage variables to assume values that are near the average $\overline{x}^i$. In a linear penalty function obtained by removing the quadratic term, the distance from $\overline{x}^i$ is not penalized, causing oscillatory behavior of the variables and impacting, convergence of the algorithm. Therefore, instead of removing the quadratic term, we approximate the quadratic distance from $\overline{x}^i$ with the absolute distance. To the best of our knowledge, this is a new technique in the context of PH.
For a generic scenario s, we define the following penalty function:

$$\hat{f}_s(x_s, i) = \lambda_s^i x_s + \frac{1}{2}\rho^i |x_s - \overline{x}^i| \tag{5.7}$$

where the penalty factor ρ^i is a vector as defined below. The intent behind the definition of $\hat{f}_s(x_s, i)$ is to mitigate the oscillatory behaviour and, at the same time, allow the solution of linear subproblems. Note that with the linear penalty the overall cost of a given first-stage variable $x_{s,j}$ has a minimum at $\overline{x}_j^i$, if $\rho_j^i \geq 2(c_j + \lambda_{s,j}^i)$, or at the minimum defined by c and λ_s^i, otherwise (see Figure 5.2(b) and Fig.5.2(c)).
Choosing the correct penalty factor for the original quadratic penalty function is critical and is often data dependent ([27, 18]). When $\hat{f}_s(x_s, i)$ is used instead of $\hat{f}'_s(x_s, i)$, the smoothness of the quadratic penalization is lost and choosing the correct penalty factor

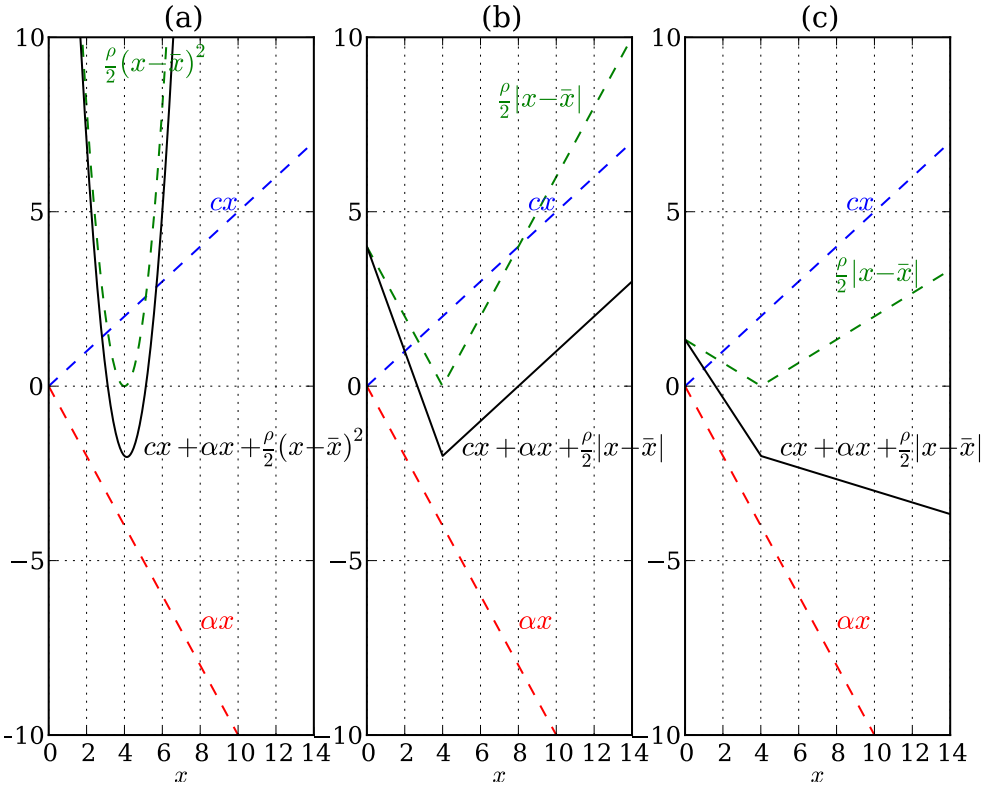

FIGURE 5.2: Examples of penalty functions: for a given first-stage variable with original cost $c = 0.5$, in (a) the penalty function $\hat{f}'_s(x_s, i)$ is represented supposing $\lambda^i_s = -1, \rho^i = 2$; in (b) and (c), the penalty function $\hat{f}_s(x_s, i)$ is represented with same λ^i_s and,respectively, $\rho^i = 2$ and $\rho^i = \frac{2}{3}$.

is as critical as in the quadratic case.

Penalty factor: We extend the element-specific penalty factor first proposed in [27]. The authors define a penalty factor, for each iteration i and for each first-stage variable j, as follows:

$$\tilde{\rho}^i_j = \frac{|c_j|}{(\max_s x^{*i,s}_j - \min_s x^{*i,s}_j)} \quad .$$

When the absolute distances replaces the quadratic term, using the same definition results in a weak penalty factor and, consequently, slow convergence of the algorithm. Instead, we use the vector $\rho^i = (\rho^i_1, ..., \rho^i_n)$, where $\rho^i_j = (\tilde{\rho}^i_j)^2$ and n is the number of first-stage variables.

In $\hat{f}'_s(x_s, i)$, any $\rho > 0$ guarantees that the subproblems are bounded. This is not the case when absolute distance is considered. To overcome this issue, for every unbounded first-stage variable j, we adjust $\tilde{\rho}^i_j$ so that the overall cost defined by c,λ and $\tilde{\rho}^i_j$ is greater then zero for any $x_{s,j} \geq \overline{x}^i_j$. In particular if we observe for a positive unbounded variable j that:

$$\tilde{\rho}^i_j \leq -2 \min_{s:(c^s_j+\lambda^{s,v}_j)<0} (c^s_j + \lambda^{s,v}_j)$$

we impose:

$$\tilde{\rho}_j^i = -2 \min_{s:(c_j^s+\lambda_j^{s,v})<0} (c_j^s + \lambda_j^{s,v}) + \gamma$$

where $\gamma > 0$. In our experiments, we set $\gamma = 1$.

5.3.2 Guided MIP Solves for feasible solutions

PH does not directly provide feasible solutions until complete convergence. For difficult to solve instances, it is highly likely that convergence will not complete within a given time limit. Existing strategies typically stop the algorithm before convergence is reached for all the first stage variables and then search for global feasible solutions by fixing some variables. Our new approach searches for feasible solutions while the algorithm iterates. Our ability to find good feasible solutions even after the first iteration is particularly useful for instances such as ours with large and difficult subproblems, in which a single iteration of mPH requires a considerable computational effort. Moreover, different iterations of the algorithm may generate different feasible solutions by exploring different parts of the feasibility region.

Guided solves: We refer to first-stage variables that have the same value in the solutions of all subproblems for a given iteration as "*agreed*" variables. We obtain feasible solutions by optimizing SUC with agreed variables fixed to the common value. We call this problem SUC' and its solution as *"guided solves"*. Since we may have different sets of agreed variables in different iterations, we may derive a different SUC' formulation at every iteration, producing a different solution.

For continuous first-stage variables, we define a tolerance in which two first-stage variables can be considered equal and therefore, agreed. For any $\epsilon > 0$, we denote by "$\epsilon-$*agreed*" all the variables for which the variance from a common value is less than ϵ. All the $\epsilon-$agreed variables can then be fixed at their average value $\overline{x}^i$. If ϵ is smaller than the feasibility tolerance of the MIP solver used for the guided solves, then the definition of agreed and ϵ-agreed variables coincide.

SUC' **feasibility:** Observe that Problem SUC' is obtained by adding constraints to the original SUC formulation; thus the feasibility region of SUC' is contained in the region of SUC. On the other hand, there is no guarantee that SUC' has feasible solutions even if SUC does. In fact, it is not always possible to satisfy the non-anticipativity constraints for all non-fixed first-stage variables for all agreed variables. However, if a combination of first-stage variables is known to be feasible for all scenarios if it is feasible for one scenario, then is possible to state a-priori that SUC' is always feasible if SUC is feasible.

ϵ-strategies: The parametric definition of ϵ-agreed variables suggests certain natural strategies. When the number of agreed first-stage variables is low, the guided solve requires a computational effort similar to the original SUC formulation. In this case, considering integer variables that have similar (unequal) values as ϵ-agreed may help.

	gap % mPH			gap % mPH	
Instance	It 1	It 2	Instance	It. 1	It. 2
D2020-03-02	0.1	0.11	D2020-12-29	0.01	0.02
D2020-01-04	0.13	0.11	D2020-10-10	0.01	0.01
D2020-05-14	0.09	0.09	D2020-07-16	0.87	-
D2020-08-07	-	0.13	D2020-06-30	0.23	0.12
D2020-09-27	0.03	0.02	D2020-04-15	0.02	0.02
D2020-11-09	0.05	0.06	D2020-02-12	0.00	0.00

TABLE 5.1: Comparison of solution quality for 5-scenarios instances

This causes an increased number of fixed variables in SUC', with consequent reduction of required computation and, potentially, solution quality.

Conversely, one may want to reduce the number of fixed first-stage variables, when easy to solve SUC' problems return poor or infeasible solutions. One strategy is to fix variables that converged only in the previous iteration. This idea arises from the fact that some generation resources are used only when uncertainty is considered. For example, the subproblems solved separately may not use such resources in the initial iterations, even if they are used in the optimal solution. In this case, fixing such resources to zero will cut off the optimum.

5.4 Computational Results

Instances and Experimental Setup: We present numerical results evaluating our algorithm mPH on a test bed containing twelve 5-scenario instances and one 20-scenario instance. Each instance corresponds to a daily hour-time-step unit commitment problem, simulated for year 2020, as described in Section 5.1.1. We ran our algorithm mPH on a machine with a Power7 processor, which has 8 cores running at 3.61GHz, with each core capable of four-way simultaneous multithreading. Power7 executes instructions out-of-order. There are 12 execution units (including 2 fixed-point units and 2 load/store units) per core shared by the 4 hardware threads.

5.4.1 Results

We compare mPH with "*direct*" solutions of the SUC formulation using CPLEX 12.4 with standard parameters except for "relative MIP gap tolerance which is set to 0.05%. We use the same version of CPLEX and the same parameter settings to solve SUC' in mPH, but set the relative MIP gap tolerance to 0.5% for the subproblems SP_s^i.

To reduce the computational requirements for solving SUC', we fixed some second-stage variables in addition to the "$\epsilon-$*agreed*" first-stage variables. If we observed that the second-stage variables for a given variable assume the same value in the solutions

of all the subproblems, then in SUC' we fix all these second-stage variables to this agreed value.

Five scenarios instance: In Table 5.1 we summarize the quality of the results achieved for the 5-scenarios instances. In column "It. 1" and "It. 2" we report the gap, in percentage, between the optimal solutions obtained using the direct approach and the feasible solutions returned by the guided solves after the first two iterations. In our experiments, we stopped mPH after just two iterations since we obtain solutions comparable to the optimum in two iterations. We did obtain infeasible SUC' problems in two cases, but the overall performance was not compromised since for every instance it was possible to reach a feasible solution in either the first or second iteration.

Table 5.2 compares the solution times (in seconds) for the 5-scenario problems. We achieve a 6x speedup in average serial solution time. A parallel implementation solves all the subproblems in parallel; the guided solve subsequent to iteration i is also carried out in parallel with the subproblems of $i + 1$. In this implementation, we find a 22x average speedup.

	Times [s]		
Instance	Direct 0.05-Opt	mPH Serial	mPH Parallel
D2020-03-02	9923.30	1642	463.3
D2020-01-04	5318.20	2503.1	643.9
D2020-05-14	13708.30	1668.3	513.1
D2020-08-07	8097.70	1269.5	412.6
D2020-09-27	11842.70	1381.4	491.7
D2020-11-09	5484.30	1543.6	380.6
D2020-12-29	9192.00	2699	691.7
D2020-10-10	35951.50	3923.7	1020.7
D2020-07-16	2965.80	1221.5	301
D2020-06-30	5988.80	1110.9	321.1
D2020-04-15	36022.50	3650.4	1151.3
D2020-02-12	7642.10	1748.2	483.8
AVERAGES	12678.1	2030.13	572.90

TABLE 5.2: Comparison of execution times for 5-scenario instances

In our experiments, we observed a decrease in subproblem solution times for the second iteration, which is partially due to information from the first iteration and partially due to a different (penalizing) objective function. We also observe a reduction in solution time for the guided solves, averaging a 40% improvement. In our experiments, the guided solve re-optimized from scratch at every iteration, with no information used from the previous iteration, so the reduced solution times can only be attributed to the different fixed variables in SUC'.

Twenty scenarios instance: Table 5.3 lists mPH execution times and solution quality for the 20-scenario instance. Time columns are the same as the 5-scenario case, and the rows here are the iteration number. The gap between the solutions from the

guided solves the best known lower bound is reported in column "Gap % SUC'_i-LB". It is clear from the results that the solutions provided by the guided solves are extremely close to optimal. The feasible solutions returned by mPH are the only valid upper bound in this case as the 20-scenarios instance is intractable when solved with the direct approach (no feasible solution was found in 24 hours).

Convergence: While mPH provided near optimal solutions after 2 iterations, the convergence of the algorithm depends on the effectiveness of the linear penalty function $\hat{f}_s(x_s, i)$. To gauge this, we ran the algorithm for ten iterations for the 5-scenarios instances and seven iterations for the 20-scenario instance; there were no cases of complete convergence.

	Times [s]			Gap %
It.	SP^i_s Sum	SP^i_s Max	SUC'_i	SUC'_i-LB
1	7172.88	466.37	8781.04	0.02
2	2600.46	186.71	2419.81	0.02
3	3012.23	214.35	2072.5	0.01
4	2543.42	202.39	2277.18	0.01
5	2786.57	260.27	1732.35	0.01
6	3614.93	439.67	2120.29	0.01
7	4033.06	382.32	1218.11	0.02

TABLE 5.3: Results for the 20-scenarios instance

In Figure 5.3 we report the measure δ. After the first iteration the δ in the 5-scenarios instances reaches very small values and remains there. In the twenty scenarios instance, we observed a slight but constant reduction of δ in the following iterations.

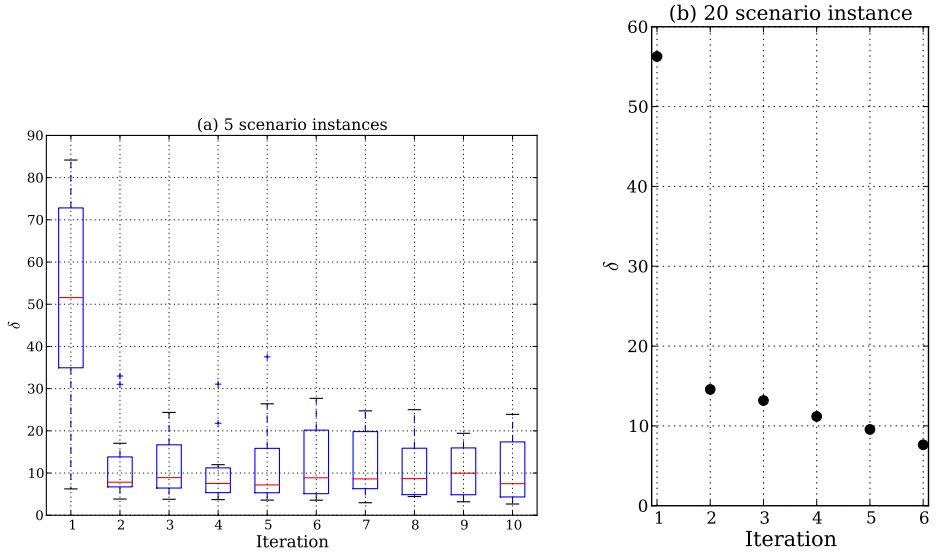

FIGURE 5.3: Evolution of δ.

To achieve complete algorithmic convergence, one could fix the agreed variables both in SUC' and also in the subproblems of following iterations. Alternatively, one could use a definition of ϵ^i that increases when the iterations of i increases.

5.5 Conclusions

We have observed that the mPH algorithm is able to produce near-optimal solutions for our stochastic unit commitment problems, with roughly a 4 times improvement in solution time over the standard approach. This allows us to solve our stochastic unit commitment problems using many more renewable generation scenarios than was previously reasonable computationally, including the solution of formerly intractable problems such as the 20-scenario case. Given the recent interest within the power industry for stochastic unit commitment models as a way of managing uncertainty [6, 20], these results represent another tool that can be used to mitigate the computational burden of such problems.

We note that the linearization introduced in the penalty function allows the resolution of very large mixed integer linear subproblems with the direct use of generic purpose solvers. Nonetheless the computation needed to solve the subproblems can be considerable, and convergence of the algorithm within the reduced number of iterations computable within a given time limit can be a challenge. Thus it is crucial to also allow the parallel solver to search for feasible solutions before convergence is reached.

The combinatorial methods proposed here suggest several areas for further analysis. First, as mentioned in the ϵ-strategies paragraph, there are different possible definitions of the agreed variables. Second, when feasible solutions are available there is the concrete possibility (not exploited in this work) to use these solutions to help convergence of PH or to speed up the resolution process in general. As an example, if for a given iteration SUC' returns a particularly good feasible solution, it may be advantageous to fix some of the agreed variables in the subproblems. This will help produce easier subproblems, with a consequent reduction of computation, and may also help convergence of the algorithm by progressively fixing more variables.

Bibliography

[1] Alexandre Belloni, AL Diniz Souto Lima, ME Piñeiro Maceira, and Claudia A Sagastizábal. Bundle relaxation and primal recovery in unit commitment problems. the Brazilian case. *Annals of Ops Research*, 120:21–44, 2003.

[2] California Independent System Operator. Integration of renewable resources: Technical appendices for California ISO renewable integration studies. Technical report, California ISO, 2010.

[3] California State Senate. Bill Number 2, April 12 2011.

[4] Claus C Carøe and Rüdiger Schultz. Dual decomposition in stochastic integer programming. *Ops Research Letters*, 24:37–45, 1999.

[5] Pierre Carpentier, G Gohen, J-C Culioli, and Arnaud Renaud. Stochastic optimization of unit commitment: a new decomposition framework. *Power Systems, IEEE Transactions on*, 11(2):1067–1073, 1996.

[6] K. Cheung, D. Gade, C. Monroy, S. Ryan, J. Watson, R. Wets, and D. Woodruff. Toward scalable stochastic unit commitment - part 2: assessing solver performance. *IEEE Transactions on Power Systems*, 2013. Submitted.

[7] Teodor Gabriel Crainic, Xiaorui Fu, Michel Gendreau, Walter Rei, and Stein W Wallace. Progressive hedging-based metaheuristics for stochastic network design. *Networks*, 58:114–124, 2011.

[8] Amal De Silva and David Abramson. Computational experience with the parallel progressive hedging algorithm for stochastic linear programs. In *Proceedings of 1993 Parallel Computing and Transputers Conference Brisbane*, pages 164–174. Citeseer, 1993.

[9] T. Edmunds, A. Lamont, V. Bulaevskaya, C. Meyers, J. Mirocha, A. Schmidt, M. Simpson, S. Smith, P. Sotorrio, P. Top, and Y. Yao. The value of storage and demand response for renewable integration. Technical Report under contract CEC-500-10-051, California Energy Commission, 2013. Awaiting approval for public release.

[10] Energy Exemplar. PLEXOS Integrated Energy Modeling Software. Product of Energy Exemplar, LLC, 2013. http://www.energyexemplar.com/.

[11] Dinakar Gade, Simge Küçükyavuz, and Suvrajeet Sen. Decomposition algorithms with parametric gomory cuts for two-stage stochastic integer programs. *Mathematical Programming*, pages 1–26, 2012.

[12] Kjetil K Haugen, Arne Løkketangen, and David L Woodruff. Progressive hedging as a meta-heuristic applied to stochastic lot-sizing. *European Journal of Operational Research*, 132(1):116–122, 2001.

[13] Thorkell Helgason and Stein W Wallace. Approximate scenario solutions in the progressive hedging algorithm. *Annals of Ops Research*, 31:425–444, 1991.

[14] Arne Lokketangen and Fred Glover. Solving zero-one mixed integer programming problems using tabu search. *European Journal of Operational Research*, 106(2):624–658, 1998.

[15] Arne Løkketangen and David L Woodruff. Progressive hedging and tabu search applied to mixed integer (0, 1) multistage stochastic programming. *Journal of Heuristics*, 2:111–128, 1996.

[16] Guglielmo Lulli and Suvrajeet Sen. A branch-and-price algorithm for multistage stochastic integer programming with application to stochastic batch-sizing problems. *Management Science*, 50:786–796, 2004.

[17] John M Mulvey and Hercules Vladimirou. Applying the progressive hedging algorithm to stochastic generalized networks. *Annals of Operations Research*, 31(1):399–424, 1991.

[18] John M Mulvey and Hercules Vladimirou. Solving multistage stochastic networks: An application of scenario aggregation. *Networks*, 21:619–643, 1991.

[19] R Tyrrell Rockafellar and Roger J-B Wets. Scenarios and policy aggregation in optimization under uncertainty. *Mathematics of operations research*, 16:119–147, 1991.

[20] P. Ruiz, C. Philbrick, E. Zak, K. Cheung, and P. Sauer. Uncertainty management in the unit commitment problem. *IEEE Transactions on Power Systems*, 24:642–651, 2009.

[21] Andrzej Ruszczyński. On convergence of an augmented lagrangian decomposition method for sparse convex optimization. *Mathematics of Operations Research*, 20(3):634–656, 1995.

[22] Andrzej Ruszczyński. Decomposition methods in stochastic programming. *Mathematical programming*, 79(1-3):333–353, 1997.

[23] Hanif D Sherali and Barbara MP Fraticelli. A modification of benders' decomposition algorithm for discrete subproblems: An approach for stochastic programs with integer recourse. *Journal of Global Optimization*, 22(1-4):319–342, 2002.

[24] Hanif D Sherali and Xiaomei Zhu. On solving discrete two-stage stochastic programs having mixed-integer first-and second-stage variables. *Mathematical Programming*, 108(2-3):597–616, 2006.

[25] Michael Somervell. *Progressive hedging in parallel*. PhD thesis, Engineering Science)–University of Auckland, 1998.

[26] Stein W Wallace and Thorkell Helgason. Structural properties of the progressive hedging algorithm. *Annals of Operations Research*, 31(1):445–455, 1991.

[27] Jean-Paul Watson, David L Woodruff, and David R Strip. Progressive hedging innovations for a stochastic spare parts support enterprise problem. *Naval Research Logistics*, 2007.

[28] Roger JB Wets. The aggregation principle in scenario analysis and stochastic optimization. In *Algorithms and model formulations in mathematical programming*, pages 91–113. Springer, 1989.

[29] Lei Wu and Mohammad Shahidehpour. Accelerating the Benders decomposition for network-constrained unit commitment problems. *Energy Systems*, 1:339–376, 2010.

[30] Qipeng P Zheng, Jianhui Wang, Panos M Pardalos, and Yongpei Guan. A decomposition approach to the two-stage stochastic unit commitment problem. *Annals of Operations Research*, pages 1–24, 2012.