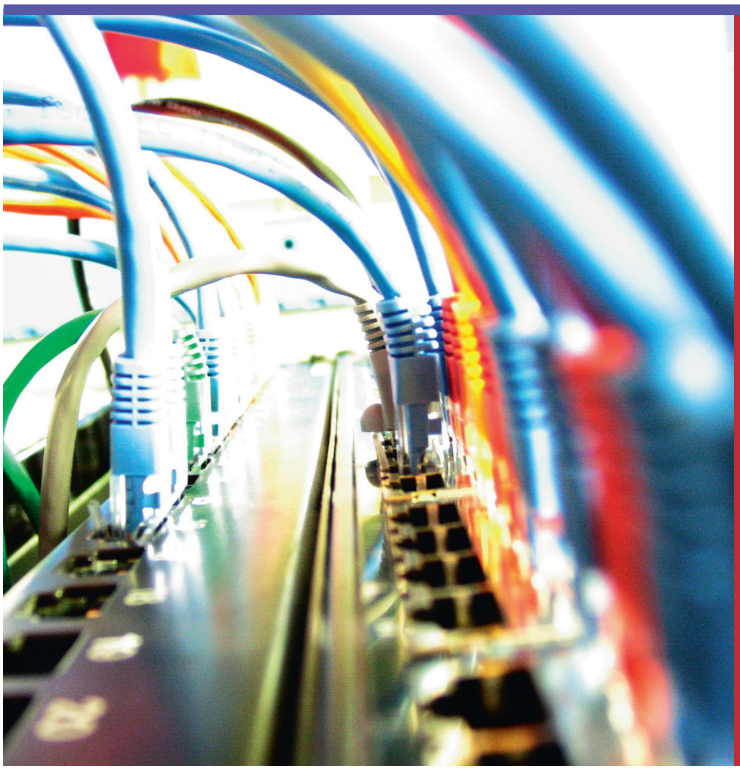
Scheduling Network Traffic for Grid Purposes

Gamst, Mette; Pisinger, David

Publication date:
2010

Document Version
Publisher's PDF, also known as Version of record

Scheduling Network Traffic for Grid Purposes



PhD thesis 10.2010

DTU Management Engineering

Mette Gamst
August 2010

DTU Management Engineering
Department of Management Engineering

Scheduling Network Traffic for Grid Purposes

Mette Gamst

Preface

This PhD study took place under the ”Industrial PhD programme”. The purpose of the programme is to perform research which is interesting not only academically but also to the industry. The PhD candidate is employed at a private company, but the study must be performed with strong connections to a university, where the candidate is expected to be present at least half of the study time.

The PhD study was performed at the Department of Computer Science, University of Copenhagen (DIKU) from June 2007 to February 2009 and at DTU Management Engineering, Technical University of Denmark from February 2009 to March 2010. Furthermore, the work was partly conducted at GlobalConnect A/S, a telecommunications company located in the outskirts of Copenhagen, Denmark. Professor David Pisinger supervised the study, Professor Brian Vinter acted as second supervisor at DIKU and CSO Niels Raun supervised at GlobalConnect A/S.

This PhD thesis consists of four introductions (Chapter 1, 2, 3 and 8), six research papers (Chapter 4, 5, 6, 7, 9 and 10) and a conclusion (Chapter 11). The research papers have been written in collaboration with co-authors, who are mentioned at the beginning of each paper. The research papers are relatively self-contained, however, the bibliography of each research paper is left out and instead one bibliography for all chapters is included at the end of the thesis.

The PhD thesis contains four parts. The first part is an introduction split into two chapters. The next part concerns the scheduling problem in grid computing and is split into five chapters. The first is an introduction to the problem and the remaining four are research papers. The third part of this thesis deals with the multi-commodity k-splittable flow problem, where each commodity may use at most k paths to route its flow. This problem has relevance in the telecommunications sector when gathering

several data packets under the same label. The third part consists of an introduction and two research papers. The final part of this thesis contains concluding remarks and suggestions for future work.

Acknowledgements

First of all I would like to thank David Pisinger for his splendid supervision, for numerous interesting discussions and for his support throughout this PhD study. I would also like to thank GlobalConnect A/S for their support and contribution to this PhD study. Especially a big thank you to Niels Raun for many interesting discussions and for sharing his extensive knowledge. Furthermore, I would like to thank Brian Vinter at Copenhagen University for many fruitful discussions and good advices, Josva Kleist at Nordic DataGrid Facility, who I visited and worked with on several occasions, and Dorit Hochbaum, who I visited at UC Berkeley. Also, I would like to thank my colleagues at DTU Management Engineering, DIKU, UC Berkeley and GlobalConnect A/S for insightful discussions and for providing a pleasant work environment, especially Bjørn Petersen for good co-operation, Laurent Muller for good co-operation and company at UC Berkeley, Simon Spoorendonk for useful advices and fruitful discussions and Tommy Clausen and Line Blander Reinhardt for proof reading parts of this thesis. Also many thanks to Gerd Behrmann and Michael Grønager from Nordic DataGrid Facility and Lars Fischer from NORDUnet for good co-operation. Finally, a big thank you goes out to my family and friends. Thanks!

Mette Gamst, May 2010

Contents

Part I

Introduction

CHAPTER 1

Introduction

1.1 Motivation

This thesis is part of a general study of grid computing performed by DTU Management Engineering at the Technical University of Denmark, the Department of Computer Science at Copenhagen University, GlobalConnect A/S and the Nordic DataGrid Facility. The thesis is a contribution to strengthen the utilization of grid computing by improving the current routing and scheduling scheme.

Grid computing is a service which provides applications, storage and computational power. The idea is that users can access the grid by plugging their computer into the wall to access the grid; just like one gets electricity. Grid computing is hence named after the power grid. The home computer of a user only has to have a good internet connection and to display graphics, thus the user can save money on buying a new computer, new software etc. every few years. The user is also freed from software and - to some extent - hardware maintenance as this is handled on the grid by grid administrators.

The full vision of grid computing has not been implemented at this point of time. Instead grid computing has become a tool for scientists to obtain computational power. A grid can therefore be viewed as a number of computer resources from (different) administrative domains working together for solving large problems. The problem size is

generally measured in the amount of required CPU cycles or required data. Researchers use grid computing to solve problems requiring more resources than available at each research group, see e.g. Grønager [96] and Shiers [176]. The methods presented in this thesis will allow researchers to solve problems of larger size and scope than what is possible today. This is highly relevant because several sciences such as biology, chemistry and physics are currently producing data at an exponential rate, see e.g. Bergeron [37], Marcotte [143] and Ricker [167].

Many different grid implementations are in use today. They all hold a scheduler which to a certain degree decides the activity in the grid, i.e., which resource computes what job and when. The schedulers do, however, generally not take network traffic into account. When a job runs on a resource its input files must be present. The data must be sent to the resource from storage. If time spent on data transmission is not taken into account when scheduling computation, the resulting plan may be infeasible; some jobs may not be able to execute on time because they are still waiting for data. In the case of an infeasible scheduling plan, the grid may have to reject execution of some jobs. Hence the grid may become an unstable computation source for the users, which may lead to a decrease in the desire for using grid computing in general. Taking data transmission into account will result in feasible scheduling solutions and in an increase in the stability of the grid. The need for considering data traffic in grid computing is illustrated in Figure 1.1. The upper figure shows regular network traffic, which never exceeds 400 Mbps. But when logging on to a grid, the network load explodes as illustrated in the bottom figure; the amount of ingoing traffic increases to 1 Gbps. The figure stems from the Nordic DataGrid Facility and is representative for grid usage. Looking at the bottom figure, it seems that all network capacity is used at certain times. That is, the network constitutes a bottleneck in some periods and will thus delay the computation of some jobs.

1.2 Solution approaches

Operations research is a discipline in applied mathematics and computer science and is widely used for solving planning problems. Operations research provides a number of tools useful for computing precise and detailed plans. These tools include mathematical modeling and programming. The interest in applying operations research to real-life problems has increased as more computer resources have become available. A research area which utilizes operations research for solving planning problems is telecommunications: re-occurring telecommunications problems which can be solved using operations research include routing data through networks, designing networks, and distributing job executions among several CPUs or computers. Operations research can thus be applied to the scheduling problem in grid computing, which is a combination of routing data and of distributing job executions.

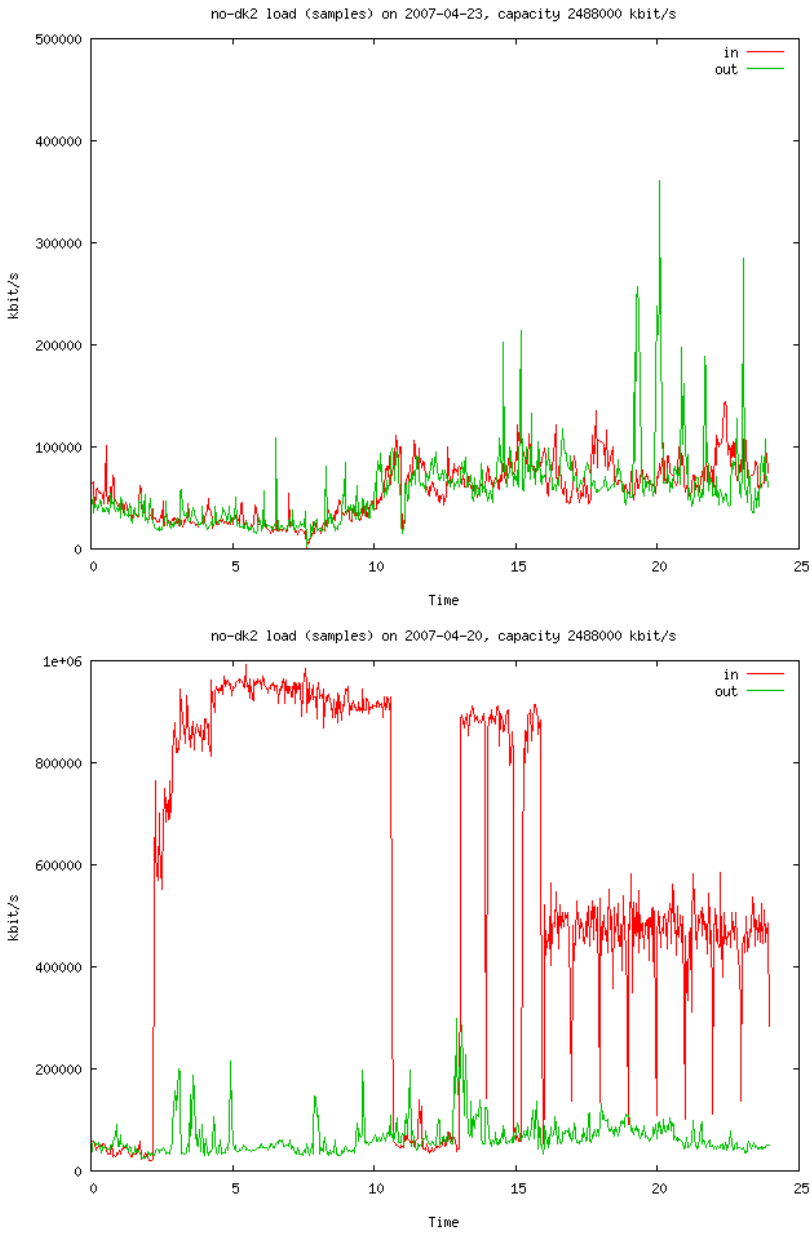


Figure 1.1: Network traffic at the Nordic DataGrid Facility: the upper figure illustrates regular network load and the lower figure illustrates network load when logged on to the grid.

A mathematical model can generally be said to contain a set of variables, which represent decisions in the corresponding problem. To ensure that only feasible decisions are made, subsets of variables are gathered in mathematical equations. These are also called constraints and they formulate properties of the corresponding problem. Finally, a model contains an objective function which defines the overall goal of solving the corresponding problem. For instance in telecommunications, we want to establish as many data connections (decision variables) as possible through a network (objective function) such that network data does not exceed the bandwidth (constraints). That is, a decision variable corresponds to establishing a data transmission, the constraints correspond to setting upper bounds on the amount of data traveling on the network links, and the objective is the total number of established data transmissions.

A mathematical model transforms possibly weak or abstract requirements into a number of precise and detailed mathematical equations such that the corresponding real-life problem is well-represented. The model is global, because it consists of mathematical equations and thus only can be read in one way. Also, a good model should be kept as simple as possible and it may be beneficial to reformulate it into more appropriate representations. This thesis concerns both modeling and reformulation of models. In Part II, the grid scheduling problem with network constraints is formalized in a mathematical model and reformulations are applied to reach more tractable representations.

Many of the problems considered in this thesis are very difficult to solve, because the set of solutions is so large that enumerating and investigating each solution is simply not feasible. These problems are said to be $\mathcal{NP}$-hard and careful considerations must be given on selecting an appropriate solution method. Three types of methods are typically employed to solve $\mathcal{NP}$-hard problems:

- Heuristic solution methods.
  This approach uses rules of thumb when finding a solution. Heuristics often choose the decision, which currently seems more appropriate without knowing exactly how the decision affects the overall solution. That is, heuristics give no guarantee of the solution quality, but they are capable of finding a solution quickly. Heuristics are often used in a real-life framework where time usage constitutes a bottleneck. See e.g. Rothlauf [169] for a study on heuristic methods.
  A special class of heuristics is meta-heuristics, which consists of general frameworks of heuristics to be applied to many classes of problems. Meta-heuristics often require more time but may also give better solutions. See e.g. Glover and Lagunda [92], Goldberg [93], and van Laarhoven and Aarts [193] for studies on meta-heuristics.
  This thesis considers heuristics for solving the grid scheduling problem. The heuristic methods are presented in Chapter 2.

- Approximation algorithms.
  The approximate solution approach gives a guarantee on the solution quality. Approximation algorithms, however, may require more time than the heuristic approach. Approximation algorithms are not part of this thesis and are thus not explained further. For more information see e.g. Vazirani [197].

- Exact solution methods.
  Solving a problem exactly or to optimality guarantees that the best possible solution is found. This approach can be very time consuming as all solutions in worst case must be explored. A number of different exact approaches exists; these include decomposition methods, dynamic programming methods, matrix manipulation methods, etc., see e.g. Nemhauser and Wolsey [152]. Some real-life problems may be tractable in practice despite being $\mathcal{NP}$-hard, for instance because of properties of the problem instances or because of properties of the problem type. In these cases, exact methods may be applied. If not, then exact methods may be used for benchmarking the performance of heuristics. Exact methods are also a useful tool for analyzing problem bottlenecks and behaviour. The grid scheduling problem and data transmission problems in telecommunications are solved to optimality in this thesis using the Dantzig-Wolfe decomposition method [54] and using standard solvers performing matrix manipulations. These exact solution methods are presented in Chapter 2.

1.3 Goals

The focus of this thesis is on the grid scheduling problem with respect to network constraints. The main goals are summarized as:

- Formalize the scheduling problem in grid computing where network limitations are taken into account.
- Use experiments to investigate the practical complexity of the problem.
- Investigate the consequences of different network topologies, specifically when using a standard packet switched network (e.g. the internet) and when using an optical network (which is circuit switched).
- Investigate the underlying network problem, when using Multi-Protocol Label Switching (MPLS) for routing data. The problem is denoted the Multi-commodity k-splittable Flow Problem (MCkFP) in operations research context.

The main problem of job scheduling in grid computing is investigated with respect to practically relevant constraints on the underlying network topology. Work in the

literature has mainly focused on solving the scheduling problem without network constraints. First a standard network topology is applied and then a more sophisticated optical network topology is considered. The goal is to investigate the impact of network constraints in the grid scheduling problem.

By implementing a number of algorithms the goal is to present several options for solving the scheduling problem according to requirements on time usage and solution quality. The methods also give an impression on how difficult the problem is to solve practically.

Operations research can be applied to several problems arising in telecommunications. The scheduling problem in grid computing is an example of this. Another is the problem of determining routing tables when using Multi-Protocol Label Switching (MPLS). In MPLS several data packets are gathered under the same label to reduce routing tables and to increase quality of service. However, the cost of sending data depends on the number of *Label Switch Paths*, thus the number of paths should be limited for each label. This corresponds to the the Multi-commodity k-splittable Flow Problem. A goal in this thesis is to investigate this problem and to improve current solution techniques from the literature.

1.4 Contributions

The main contributions of this thesis are summarized below:

- The offline scheduling problem in grid computing with respect to network constraints is formalized and proved to be $\mathcal{NP}$-hard. Grid components are assumed to be connected through a packet switched network. Experiences with heuristic approaches are discussed and exact solution methods are proposed.
- Comparing the proposed exact methods with heuristic experiments for the offline grid scheduling problem using a packet switched network shows that the problem can be solved to optimality for all tested benchmark instances with up to 1000 jobs and resources.
- The offline scheduling problem in grid computing where components are connected through an optical network is formalized and proved to be $\mathcal{NP}$-hard. Exact and heuristic solution methods are proposed.
- Comparing exact and heuristic solution methods for the formalized offline scheduling problem in grid computing using an optical network, shows that the heuristics perform better. The exact solution method times out for many instances,

while the heuristics have very small running times and finds solution with an average solution gap of only 3%.

- The real-life grid computing network of the Nordic DataGrid Facility is formalized into a mathematical formulation. Solving the formulation to optimality reduces network usage significantly. Practical, relevant changes to the grid and network functionality are added to the formulation and the effects are analyzed.
- New exact algorithms for the Multi-commodity k-splittable Flow Problem are proposed; one for the minimum cost problem and one for the maximum flow problem. The algorithms are developed with the intention of reducing symmetry in the solution space.
- The exact algorithms for the Multi-commodity k-splittable Flow Problem outperform exact algorithms from the literature. The algorithms eliminate a significant amount of symmetry in the solution space and even though this complicates branching, the algorithms perform very well.

The contributions are introduced further in the following section and are discussed in detail in Part II for the grid scheduling problem and in Part III for the Multi-commodity k-splittable Flow Problem.

1.5 Overview of PhD thesis

The thesis consists of the following parts:

Part I: Introduction. This part consists of two chapters: the current chapter which contains a motivation and an introduction to the thesis. The second chapter describes the overall solution methods in the thesis. The methods are heuristics and exact methods, the latter including branch-and-bound with column generation, cutting planes, and stabilized column generation. The second chapter is meant as an introduction to the solution methods and may be skipped by the advanced reader.

Part II: Scheduling in grid computing. This part considers the scheduling problem in grid computing. The part contains four papers:

- **Integrated job scheduling and network routing.** This paper considers the integrated job scheduling and network routing problem, which has application in grid computing. The problem is considered to be offline, i.e., it computes a job execution and data transfer plan in advance. The paper suggests three algorithms for solving the problem to optimality. The first is a

straight-forward branch-and-price algorithm, which runs into memory and time problems rather quickly. Thus the algorithm is extended to a branch-and-cut-and-price algorithm, where only violated constraints are included in the master problem. This reduces memory and time usage significantly. However, the algorithm still has room for improvement which is done by adding stabilized column generation. This reduces the number of iterations in each branch-and-bound node considerably and hence dramatically improves memory usage and running times. Instances with up to 1000 jobs and 1000 resources are solved to optimality. The work has been presented as follows:

  - A paper co-authored with David Pisinger is in submission [86].
  - Poster presentation at the Foundations for Innovative Research-based Software Technologies (FIRST) Retreat, Denmark 2008 (presenter: Mette Gamst).
  - An extended abstract is in Proceedings of Forskningsnet Konferencen, Denmark 2007 [77] and the work was also presented at this conference (presenter: Mette Gamst).

  Furthermore, the work was presented at the ALGO seminars at Department of Computer Science, University of Copenhagen, 2008 (presenter: Mette Gamst).

- **A survey of the routing and wavelength assignment problem.** The Routing and Wavelength Assignment problem (RWA) arises when routing data through an optical network. In an optical network each data connection travels on a given path at a given wavelength. Each wavelength on a fiber can be used by at most one data connection because of hardware limitations. RWA is the problem of finding routes and wavelengths for a number of data connections. The survey presents the most common solution methods from the literature, proposes theoretical running times for the methods and discusses their computational evaluations. Furthermore, suggestions for future directions are given. The work has been presented as follows:

  - A Technical Report is published at DTU Management Engineering, Technical University of Denmark [78].

- **On the integrated job scheduling and constrained network routing problem.** This paper considers the offline job scheduling and data transfer problem in grid computing where the underlying network is optical. The problem is considered as a combination of the offline grid scheduling problem described in the first paper of this part, and the RWA problem described in the second paper. A branch-and-price algorithm is presented and implemented. Test results show that although the algorithm generally performs better than CPLEX, it still has memory and time problems. Thus a number of heuristics are proposed, based on merging grid heuristics with heuristics for the RWA problem. Test results show that the grid scheduling heuristics

have the larger impact on performance. The best heuristic setting performs well with an average solution gap of 3% and solves all instances within seconds. The work has been presented as follows:

- A paper is in submission [81].
- A short paper is in the Proceedings of the International Symposium on Combinatorial Optimization 2010 (ISCO'10) [80] and was also presented at this conference (presenter: Mette Gamst).
- Presentation at the Department of Industrial Engineering, Operations Research, UC Berkeley, 2009 (presenter: Mette Gamst).

Furthermore, the work was presented at the ORSEM seminars at DTU Management Engineering, Technical University of Denmark, 2010 (presenter: Mette Gamst). Preliminary work was presented at the ALGO seminars at the Department of Computer Science, University of Copenhagen, 2008 (presenter: Mette Gamst).

- **Analysis of internal network requirements for the distributed Nordic Tier-1.** This paper concerns the real-life grid computing system from the Nordic DataGrid Facility (NDGF). The paper describes and formalizes the system. The mathematical formulation is optimized using CPLEX and the resulting system is analyzed. The paper shows how operations research can help utilizing real-life grid computing systems, which results in more stable and efficient grid systems. The work has been presented as follows.
  - A paper written with Gerd Behrmann, Lars Fischer, Michael Grønager and Josva Kleist is currently being finished.
  - Early work is in Proceedings of Computing in High Energy and Nuclear Physics (CHEP) 2009, where it was also presented (presenter: Josva Kleist) [33].

Part III: The Multi-commodity k-splittable Flow Problem. This part considers the Multi-commodity k-splittable flow problem and contains two papers:

- **Two- and three-index formulations for the multi-commodity k-splittable flow problem.** This paper considers the problem of sending a number of commodities through a network subject to edge capacities and such that each commodity uses at most k paths. The objective is to minimize the total transmission cost. We present a mathematical formulation, which is simpler than that used in the literature. A corresponding new branch-and-price algorithm is proposed and is compared with the work from the literature. The simpler formulation eliminates much symmetry in the solution space but also complicates branching slightly. The new algorithm outperforms exact algorithms from the literature, both with respect to the number of solved instances and with respect to time usage. The work has been presented as follows:

  - A paper co-authored with Peter N. Jensen, David Pisinger and Christian Plum is published in the European Journal of Operations Research, 2010 [83].
  - A short paper co-authored with Peter N. Jensen, David Pisinger and Christian Plum is in Proceedings of the International Network Optimization 2009 (INOC'09) [82] and was also presented at this conference (presenter: Mette Gamst).

  Furthermore the work was presented at the ORSEM seminars at DTU Management Engineering, Technical University of Denmark, 2009 (presenter: Mette Gamst).

- **Comparing branch-and-price algorithms for the multi-commodity k-splittable flow problem.** The final paper of this part also considers the problem of sending a number of commodities through a network subject to edge capacities and such that each commodity uses at most k paths. The objective is to maximize the total amount of transmitted flow. The simpler model and corresponding branch-and-price algorithm from the minimum cost version are applied to the maximum flow problem, but test results are not as promising when comparing to a branch-and-price algorithm from the literature. The reason for this is that the simpler model combined with its branching strategy causes a large branch-and-bound tree, when the objective function is to maximize the total amount of transmitted flow. Hence a new branch-and-price algorithm is proposed, where column generation remains unchanged and where a new branching strategy adds cuts to the master problem. The new branch-and-price algorithm performs very well and outperforms both the former algorithm and the algorithm from the literature by solving more instances and spending less time. The work has been presented as follows:

  - A paper is co-authored with Bjørn Petersen and is in submission [84].
  - An extended abstract is in the Proceedings of the International Symposium of Mathematical Programming 2009 (ISMP'09) [85] and was also presented at the conference (presenter: Mette Gamst).

Part IV: Conclusion. The final part of the thesis contains a summary, some concluding remarks and suggestions for future work.

Each paper is discussed further in the introductions for each part. That is, the four papers in Part II are evaluated in Chapter 3 and the two papers in Part III are discussed in Chapter 8.

CHAPTER 2

Introduction to solution methods

This chapter describes the solution approaches which are used in this thesis. Both exact solution methods and greedy heuristics are considered. First comes an introduction to the exact solution methods, which are based on Dantzig-Wolfe decomposition and branch-and-bound. Afterwards follows an overview of greedy heuristics given.

2.1 Exact solution method

The exact solution approaches in this thesis are based on the Dantzig-Wolfe decomposition technique used in a branch-and-bound context. Dantzig-Wolfe decomposition transforms the original mathematical problem into a *master problem*, where the number of columns may be large but the number of rows is reduced. To make the new model more tractable, columns are generated iteratively in the hopes of only having to include a subset of the columns in the model. This is denoted column generation and consists of solving a *pricing problem* in each iteration. When the lower bound to the problem is found using Dantzig-Wolfe decomposition and column generation in a branch-and-bound context, the resulting method is denoted a *branch-and-price algorithm*. It may seem straight-forward to add the branch-and-bound search method to the column generation procedure, however, several issues must be taken into account. One

is that branching may change the structure of the pricing problem and hence make it significantly more difficult to solve. Another is the complexity of the solution method for solving the pricing problem. The pricing problem is solved a potentially large number of times, thus the corresponding solution approach should have good performance.

In this thesis branch-and-price is used for solving the $\mathcal{NP}$-hard scheduling problem in grid computing context to optimality. The reason for this is that the grid scheduling problem takes on a form, which is suitable for Dantzig-Wolfe decomposition and that Dantzig-Wolfe decomposition combined with branch-and-bound shows very good results for a wide variety of problems in the literature, see e.g. Barnhart et al. [27], de Aragão and Uchoa [55] Desaulniers et al. [57], and Lübbecke and Desrosiers [136]. Many of these problems share similarities with the grid scheduling problem considered, e.g., a variety of multi-commodity network flow problems, see Alvelos [7], Alvelos and de Carvalho [8], Barnhart et al. [26], and Truffot and Duhamel [190]. Applying other exact solution approaches such as Benders decomposition, see Benders [36], or dynamic programming, see Bellman [35], would be less straightforward. These and many other exact solution approaches exist, but they will not be discussed any further in this chapter. Instead an overview can be seen in e.g. Chvatal [50], Lübbecke and Desrosiers [136], Martin [146], Nemhauser and Wolsey [152], Schrijver [174], and Wolsey [204].

This chapter introduces the used exact solution methods and is not meant to be an in-depth survey but more a guide for understanding the basics of the approaches. For details on the methods and examples of applications, see e.g. Desaulniers et al. [56], Lübbecke and Desrosiers [136], and Nemhauser and Wolsey [152]. The chapter is organized as follows. First Dantzig-Wolfe decomposition is introduced in Section 2.1.1, which is followed by column generation in Section 2.1.2. Adding cuts to strengthen the mathematical formulation is discussed in Section 2.1.3. Branching is described in Section 2.1.4 and overall solution methods are discussed in Section 2.1.5. Finally, methods for stabilizing dual variables are presented in Section 2.1.6.

2.1.1 Dantzig-Wolfe decomposition

Dantzig-Wolfe decomposition was introduced by Dantzig and Wolfe [54] and consists of reformulating a problem into a master problem and a pricing problem for improving the tractability of large-scale problems. The master problem typically has fewer constraints than the original problem, but the number of columns may be very large. The pricing problem generates columns, which have the potential to improve the current solution.

In order to Dantzig-Wolfe decompose a problem, the constraint matrix should take on

a certain structure and consist of a number of *independent* constraints and a number of *connecting* constraints. The constraint matrix is block-angular, i.e., the matrix can be divided into blocks with non-zero coefficients. These blocks constitute the independent constraints. Connecting constraints binds the columns together. Consider the problem:

$$\min \quad \sum_{k \in K} c^k x^k \tag{2.1}$$

$$\text{s. t.} \quad \sum_{k \in K} A^k x^k \leq b \tag{2.2}$$

$$D^k x^k \leq d^k \qquad \forall k \in K \tag{2.3}$$

$$x^k \in \mathbb{Z}_+^{nk} \qquad \forall k \in K \tag{2.4}$$

where K is the set of blocks and A^k and D^k constitute the constraint matrix. Constraints A^k are the connecting block, and D^k the independent block.

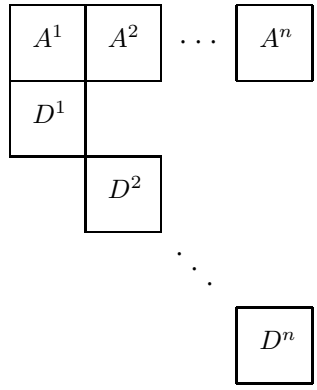


Figure 2.1: The desired matrix structure for Dantzig-Wolfe decomposition. The blocks $A^1,\ A^2, \ldots, A^n$ are connecting constraints and the blocks $D^1,\ D^2, \ldots, D^n$ are independent constraints.

Figure 2.1 illustrates this matrix consisting of connecting and independent constraints as blocks A^k and D^k, respectively. Now, we define the domains X^k as $X^k = \{x^k \in \mathbb{Z}_+^{nk}, D^k x^k \leq d^k\}$ and we can rewrite our problem into:

$$\min \quad \sum_{k \in K} c^k x^k \tag{2.5}$$

$$\text{s. t.} \quad \sum_{k \in K} A^k x^k \leq b \tag{2.6}$$

$$x^k \in X^k \qquad \forall k \in K \tag{2.7}$$

Note that this problem only contains the connecting constraints. The variables x^k must satisfy the independent constraints, which thus are left out. According to the theorems

of Minkowski and Weyl [173], each $k \in K$ can be written as a convex combination of the extreme points $\{x^{kp}\}_{p \in P^k}$ and of the extreme rays $\{x^{kr}\}_{r \in R^k}$:

$$x^k = \sum_{p \in P^k} x^{kp} \lambda_{kp} + \sum_{r \in R^k} x^{kr} \lambda_{kr}$$
$$\sum_{p \in P^k} \lambda_{kp} = 1$$
$$\lambda_{kp} \in \{0, 1\}$$
$$\lambda_{kr} \in \mathbb{Z}_+$$

This leads to a reformulation of the problem (2.5) - (2.7) and is named the master problem:

$$\min \quad \sum_{k \in K} c^k \left(\sum_{p \in P^k} x^{kp} \lambda_{kp} + \sum_{r \in R^k} x^{kr} \lambda_{kr} \right) \tag{2.8}$$

$$\text{s. t.} \quad \sum_{k \in K} A^k \left(\sum_{p \in P^k} x^{kp} \lambda_{kp} + \sum_{r \in R^k} x^{kr} \lambda_{kr} \right) \leq b \tag{2.9}$$

$$\sum_{p \in P^k} \lambda_{kp} = 1 \qquad \forall k \in K \tag{2.10}$$

$$\lambda_{kp} \in \{0, 1\} \qquad \forall p \in P^k, \forall k \in K \tag{2.11}$$

$$\lambda_{kr} \in Z_+ \qquad \forall r \in R^k, \forall k \in K \tag{2.12}$$

This model holds fewer constraints than the original formulation, but the number of columns may be very large. How to deal with the large number of variables is discussed in the next section.

EXAMPLE: Consider the Minimum Cost Multi-Commodity unsplittable Flow Problem (MCMCuFP), which consists of sending a number of commodities through a capacitated network such that the total routing cost is minimized and such that each commodity uses exactly one path.

The network is represented as a graph with nodes and edges $G = (V, E)$. Commodities are represented by the set L and each commodity $l \in L$ consists of a source node, a target node, and a quantity q^l to route. Let $c_{ij} \geq 0$ be the cost of routing one unit of flow on edge $(ij) \in E$ and let d_{ij} be the capacity of edge $(ij) \in E$. Finally, let $x^l_{ij} \in \{0, 1\}$ be a binary variable indicating whether or not

commodity $l \in L$ visits edge $(ij) \in E$. Now MCMCuFP can be formulated as:

$$\min \quad \sum_{l \in L} \sum_{(ij) \in E} c_{ij} q^l x_{ij}^l \tag{2.13}$$

$$\text{s. t.} \quad \sum_{l \in L} q^l x_{ij}^l \leq d_{ij} \qquad \forall (ij) \in E \tag{2.14}$$

$$\sum_{(ij) \in E} x_{ij}^l - \sum_{(ji) \in E} x_{ji}^l = b_i^l \quad \forall i \in V, \forall l \in L \tag{2.15}$$

$$x_{ij}^l \in \{0, 1\} \qquad \forall (ij) \in E, \forall l \in L \tag{2.16}$$

The objective (2.13) minimizes the total cost of routing all commodities. The first constraint (2.14) ensures that edge capacities are not violated. In constraint (2.15) let $b_i^l = 1$ if i is the source node of commodity l, let $b_i^l = -1$ if i is the target node of commodity l, and let $b_i^l = 0$ otherwise. Constraint (2.15) ensures that each commodity is routed from its source node to its target node. Finally the bound (2.16) makes sure that variables take on binary values.

Barnhart et al. [26] Dantzig-Wolfe decomposed MCMCuFP such that the pricing problem generates a path for each commodity and the master problem merges the paths into an overall feasible solution. Let P be the set of paths and let the cost c_p of each path be defined as the sum of visited edge $\sum_{(ij) \in p} c_{ij}$. The binary variable $x_p^l \in \{0, 1\}$ indicates whether or not commodity $l \in L$ uses path $p \in P$. Also, let δ_{ij}^p be a constant denoting whether or not path p visits edge $(ij) \in E$. The master problem is:

$$\min \quad \sum_{l \in L} \sum_{p \in P} c_p q^l x_p^l \tag{2.17}$$

$$\text{s. t.} \quad \sum_{l \in L} q^l \delta_{ij}^p x_p^l \leq d_{ij} \quad \forall (ij) \in E \tag{2.18}$$

$$\sum_{p \in P} x_p^l = 1 \qquad \forall l \in L \tag{2.19}$$

$$x_p^l \in \{0, 1\} \qquad \forall p \in P, \forall l \in L \tag{2.20}$$

The objective (2.17) still minimizes the total cost of routing the commodities and the first constraint (2.18) makes sure that edge capacities are satisfied. Constraint (2.19) says that each commodity can use exactly one path and the bound (2.20) ensures that variables take on feasible values.

2.1.2 Delayed column generation

When applying LP relaxation to the master problem, it can be used to calculate lower bounds for the original problem. In the relaxed formulation, the variables λ_{kp} and λ_{kr}

are continuous. The number of columns may be very large, thus an idea is to only include a subset of the columns. In this case we denote the relaxed version of (2.8)-(2.12) the *restricted master problem*, because only a subset of columns are included. Columns are generated iteratively by solving the pricing problem. Only columns, which have the potential to improve the current solution to the restricted master problem, are added. This procedure is denoted *delayed column generation*, or simply *column generation*.

To decide whether or not a column has potential to improve the current solution to the restricted master problem, the dual variables of the current solution are considered. Consider the restricted master problem:

$$\begin{aligned} \min \quad & \sum_{j \in J} c_j x_j \\ \text{s. t.} \quad & \sum_{j \in J} a_j x_j \geq b \\ & x_j \in X \end{aligned} \tag{2.21}$$

The *reduced cost* for a column $j \in J$ is defined as $c_j - ya_j$ where y is the dual cost vector. In minimization problems, a generated column has potential to improve the current solution to the restricted master problem if its reduced cost is negative; in maximization problems positive reduced costs are sought. Now, the objective of the pricing problem is the reduced cost and the constraints are the independent constraints of the original problem:

$$\begin{aligned} \min \quad & (c_j - ya_j)x_j \\ \text{s. t.} \quad & Dx_j \leq d \\ & x_j \in \mathbb{Z}_+^n \end{aligned} \tag{2.22}$$

A pricing problem is generated for each block $k \in K$ of the original problem. The pricing problems for different blocks may thus differ. Columns generated by the pricing problem, are not necessarily part of the solution in the following iteration even though they had negative reduced costs. If one generated column becomes part of the next solution then the remaining generated columns may become uninteresting. Also, even if a column is part of the solution in the iteration just after its generation, the column is not necessarily part of an optimal solution.

The overall column generation procedure can now be stated as:

1. Solve the restricted master problem (2.21)
2. Generate columns with the most negative reduced cost by solving the corresponding pricing problems (2.22)
3. If new columns are generated go to step 1, otherwise stop

Often it is only slightly more expensive to generate several columns at a time. Hence this may be beneficial, for instance when the pricing problem is difficult to solve, e.g. $\mathcal{NP}$-hard. In this case, the pricing problem can also be solved heuristically. However, when the heuristic cannot generate a column with negative reduced cost, then the pricing problem must be solved to optimality to ensure that the column generation procedure eventually gives an optimal solution.

EXAMPLE (CONT). Consider the Minimum Cost Multi-Commodity unsplittable Flow Problem from the previous example and how the problem was Dantzig-Wolfe decomposed. This example shows how to generate columns for the master problem according to Barnhart et al. [26]. The restricted master problem became:

$$\min \quad \sum_{l \in L} \sum_{p \in P} c_p q^l x_p^l \tag{2.23}$$

$$\text{s. t.} \quad \sum_{l \in L} q^l \delta_{ij}^p x_p^l \leq d_{ij} \quad \forall (ij) \in E \tag{2.24}$$

$$\sum_{p \in P} x_p^l = 1 \qquad \forall l \in L \tag{2.25}$$

$$x_p^l \in \{0, 1\} \qquad \forall p \in P, \forall l \in L \tag{2.26}$$

Let $\pi_{ij} \leq 0$ be the dual of constraint (2.24) and $\sigma^l \in \mathbb{R}$ be the dual of constraint (2.25). The reduced cost for a column p for a commodity l is:

$$\bar{c}_p^l = \sum_{(ij) \in E} q^l (c_{ij} - \pi_{ij}) - \sigma^l$$

The pricing problem for each column p and commodity l seeks to find columns with negative reduced cost. Now, σ^l is known for each commodity and the reduced cost can be rewritten as:

$$\sum_{(ij) \in E} q^l (c_{ij} - \pi_{ij}) < \sigma^l$$

Let the cost of each edge $(ij) \in E$ in the graph be replaced by $(c_{ij} - \pi_{ij})$, which is non-negative because $c_{ij} \geq 0$ and $\pi_{ij} \leq 0$. The pricing problem consists of finding the shortest path from the source node to the target node of the commodity, such that the total (reduced) cost is minimized. Because edge weights are non-negative, the pricing problem is polynomially solvable. If the pricing problem finds a path with total cost less than σ^l then the corresponding column is priced into the master problem.

2.1.3 Cutting planes

Solution methods adding cuts to the master formulation are proposed in this thesis: the added cuts are violated original constraints in the master problem, which were initially left out. For this reason, this section only gives a very brief introduction to cutting planes and does not go into details about specific cuts.

A cut is a valid inequality cutting off parts of the relaxed solution space which is infeasible to the original problem. The cut is derived from either the master problem or the original problem formulation. It is not beneficial to add valid inequalities, which do not cut off parts of the solution space, as the inequalities would only increase the size of the mathematical model. Cuts can be used instead of or together with Dantzig-Wolfe decomposition to tighten the LP-relaxation of some constraints $Dx \geq d$ in the original problem. If cuts are added in a column generation context, then the pricing problem must handle the extra dual variables stemming from the cuts. Care must be taken to avoid adding cuts, which complicate the structure of the pricing problem too much.

Cuts are derived by solving a *separation algorithm*, which finds some x for which $Dx \geq d$ is violated in the current LP-solution. If $Dx \geq d$ is from the convex hull of the integer problem, then we can add all such cuts until the convex hull has been fully found. This cutting planes algorithm was proposed by Gomory [94, 95]. However, deriving all cuts is as difficult as column generation with respect to complexity, i.e., if the pricing problem is $\mathcal{NP}$-hard then so is the separation routine, see Grötschel et al. [97]. For more details on cuts, separation routines, addition of cuts etc., see Desaulniers et al. [57], Martin [146], and Wolsey [204].

EXAMPLE (CONT). Recall the Minimum Cost Multi-Commodity unsplittable Flow Problem (MCMCuFP) from the previous examples. This example shows how to add cuts to the LP-relaxed MCMCuFP according to Barnhart et al. [26]. The constraint:

$$\sum_{l \in L} q^l x_{ij}^l \leq d_{ij} \quad \forall (ij) \in E$$

ensures that edge capacities are never violated. In a fractional solution, however, we may have a subset $C \subseteq L$ of commodities visiting edge $(ij) \in E$, where:

$$\sum_{l \in C} q^l > d_{ij}$$

Denote C a *cover*. To potentially strengthen the LP-relaxed mathematical formulation, we add the *cover inequality*:

$$\sum_{l \in C} x_{ij}^l \leq |C| - 1$$

2.1.4 Branching

The branch-and-bound algorithm was first presented by Land and Doig [133] and can be illustrated as a branch-and-bound tree as shown in Figure 2.2. An LP-relaxed problem is solved in the root note. If the solution is not feasible for the original (not LP-relaxed) problem, then some branching constraints are added. The resulting new problems are solved in the children nodes in the branch-and-bound tree. The branch-and-bound procedure is repeated in each *branching child*. The procedure consists of three parts:

Bounding. The problem in the current branch-and-bound node is solved.

Branching. Branching constraints are added to the current solution. More details are given below.

Pruning. A global upper bound for minimization problems (lower bound for maximization problems) is maintained throughout the branch-and-bound tree. If the solution in the current branch-and-bound node is greater than the upper bound for a minimization problem (smaller than the lower bound for a maximization problem) then the branch-and-bound node is discarded, because the problem in the node can never hold an optimal solution to the original problem.

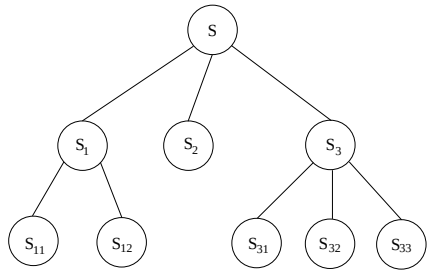


Figure 2.2: An example of a branch-and-bound tree. The original problem is LP-relaxed into S and is solved in the root node. In each child node a slightly modified problem S_i is solved.

The purpose of branching is to systematically search the solution space such that an optimal solution is eventually found. Branching cuts off parts of the solution space in each branching child. The branching strategy must ensure the finiteness of the solution approach and that all optimal solutions remain intact in the branch-and-bound tree. A

simple branching strategy is to find a variable with a fractional value and then create two branching children, where the variable is upper bounded by the floored fractional value and lower bounded by the ceiled fractional value, respectively. This may, though, not be a very good strategy if the number of variables is very large.

A branching strategy is to add cuts on single variables or on sums of variables. Adding a cut on a single variable corresponds to changing the bound of the variable. Cuts can be imposed on sums of variables from the master problem or on variables from the original formulation, see Desaulniers et al. [57].

For historical overviews, examples of branching strategies from the literature and detailed discussions on branching schemes, see for instance Ryan and Foster [171], Vanderbeck [194] and Villeneuve et al. [200].

EXAMPLE (CONT). Recall the Minimum Cost Multi-Commodity unsplittable Flow Problem (MCMCuFP) from the previous examples. The restricted master problem is LP-relaxed into:

$$\begin{aligned}
\min \quad & \sum_{l \in L} \sum_{p \in P} c_p q^l x_p^l \\
\text{s. t.} \quad & \sum_{l \in L} q^l \delta_{ij}^p x_p^l \leq d_{ij} && \forall (ij) \in E \\
& \sum_{p \in P} x_p^l = 1 && \forall l \in L \\
& 0 \leq x_p^l \leq 1 && \forall p \in P, \forall l \in L
\end{aligned}$$

An optimal solution to the LP-relaxed restricted master problem may be fractional and thus infeasible for the original problem. In a fractional solution, some commodities use more than one path to send their flow through the network. Barnhart et al. [26] suggest a branching strategy which eventually ensures an integer solution and which does not destroy the structure of the pricing problem.

Let the *divergence node* d_l be the first node which has one incoming and several outgoing paths for commodity l. The outgoing visited edges are divided into two balanced subsets $A(d_l, a1)$ and $A(d_l, a2)$. Two branching children are generated. In each branching child we forbid usage of the edges in the corresponding subset:

$$\sum_{p \in P} \sum_{e \in A(d_l, a1)} \delta_e^p x_p^l = 0 \quad \text{vs.} \quad \sum_{p \in P} \sum_{e \in A(d_l, a2)} \delta_e^p x_p^l = 0$$

where δ_e^p is a constant indicating whether or not path p visits edge e. The pricing problem for a commodity l can easily be modified into fitting the branching strategy: forbidden edges for commodity l are simply removed from the graph.

2.1.5 Overall exact solution approaches

Incorporating column generation in a branch-and-bound context gives a *branch-and-price* algorithm. In each branching node, bounding is done by column generation. It is important to consider which impact the branching strategy has on the pricing problem in the branching children. Adding branching cuts, for instance, affects the pricing problem. The reduced costs must consider the new dual variables from the branching cuts. Thus, the branching strategy should seek to limit the impact on the pricing problem.

Using cutting planes for bounding the problem gives a *branch-and-cut* algorithm. Cuts are added throughout the branch-and-bound tree. As stated previously in Section 2.1.3 it can be very time consuming to derive all cuts such that an optimal integer solution is reached for the LP-relaxed problem. The branch-and-cut algorithm seeks a compromise between reaching good bounds in each branch-and-bound node and calculating the bounds quickly. Successful applications of branch-and-cut include the Traveling Salesman Problem (TSP), see Applegate et al. [15] and the Capacitated Vehicle Routing Problem (CVRP), see Lysgaard et al. [141]

Using both column generation and cutting planes gives a *branch-and-cut-and-price* algorithm. Which cuts to add when using this algorithm name can be discussed: is the algorithm a branch-and-*cut*-and-price algorithm if only branching cuts are added? Or if the added cuts are constraints from the master problem, which are only added when violated? In this thesis I denote the latter approach branch-and-cut-and-price. In the branch-and-cut-and-price algorithm it is important to consider how the cuts affect the pricing problem. As argued for the branch-and-price algorithm, added cuts should not complicate the structure of the pricing problem too much.

2.1.6 Stabilization of dual variables

In column generation we use the values of dual variables of the current solution for calculating the reduced costs. The dual variables may, however, take on unfortunate values.

EXAMPLE: Recall the Minimum Cost Multi-Commodity unsplittable Flow Problem (MCMCuFP) from the previous examples. The pricing problem for a commodity $l \in L$ is a shortest path problem with edge weights $(c_{ij} - \pi_{ij}) \geq 0$ and the goal is to find a path with reduced cost:

$$\sum_{(ij)\in E} q^l(c_{ij} - \pi_{ij}) - \sigma^l < 0$$

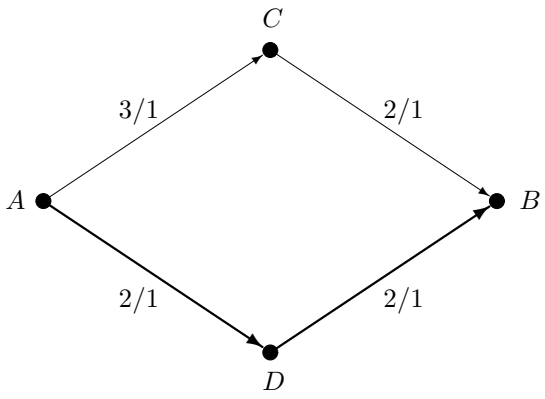


Figure 2.3: A network consisting of four nodes and edges. The cost and capacity of each edge is shown.

Consider the network in Figure 2.3. Given is a commodity l with source node A, target node B and one unit of flow to send through the network $q^l = 1$. Edge cost and capacity are displayed at each edge. The thick lines ($A \rightarrow D \rightarrow B$) represent the currently chosen path for commodity l. Let $\sigma^l = -8$ and let π_{ij} be defined as:

Edge:	(A, C)	(C, B)	(A, D)	(D, B)
π_{ij} :	-1	-1	-3	-3

The next column to be generated contains the path $A \rightarrow C \rightarrow B$ with negative reduced cost $3 + 2 - (-1 - 1) - 8 = -1$. The path, however, will never be part of an optimal solution as the current path is cheaper. In stabilized column generation we seek to find better values for dual variables such that the number of iterations and added columns is reduced. In this example, the values of all π_e could be set to -2, which would prevent the generation of the uninteresting path $A \rightarrow C \rightarrow B$.

2.1.6.1 Stabilizing methods in the literature

Several methods for stabilizing dual variables exist in the literature; this section presents the most common of these. The motivation for using stabilization is that dual variables may not necessarily converge nicely toward their respective optimal values but instead may take on fluctuating values; this is illustrated in Figure 2.4, which is

taken from Lübbecke and Desrosiers [135]. A reason for the poor convergence of the dual variables may be that many LP-solvers return an extreme point in the dual solution space and especially in the beginning of column generation where the master problem holds few columns, the values of the dual variables often fluctuate, see Sigurd [177]. This is especially a problem for degenerated problems, which have an infinite number of dual solutions, see Rousseau et al. [170]. Stabilizing the dual variables may thus reduce the number of iterations and the number of generated columns, which again may reduce the solution time and memory usage.

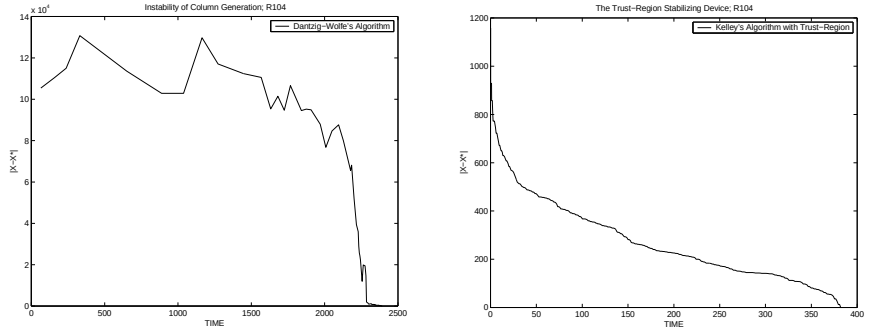


Figure 2.4: The left figure illustrates the convergence of dual variables over time. Using stabilization of dual variables results in the convergence of dual variables illustrated in the right figure [135].

Stabilization methods in column generation try to prevent the dual variables from taking on values significantly different from the values in the last iteration. A stabilization approach is to define a box covering the last values of dual variables and modifying the master problem to ensure that future dual variables take on values lying in that box, see Marsten et al. [145]. Another method is to modify the master problem such that differences in the values of dual variables are punished linearly, see Kim et al. [120]. A combination of the two approaches is also possible: a box is defined and if dual variables take on values outside the box, a penalty is added to the objective function, see duMerle et al. [60]. For more stabilization methods, we refer to the overview and work of Neame [151] and Lübbecke and Desrosiers [135].

2.1.6.2 Interior point stabilization of dual variables

The stabilization method used in this thesis was presented by Rousseau et al. [170]. The idea behind interior point stabilization is to identify a number of extreme points in the dual solution space and then to calculate a point lying within the dual solution space (an *interior* point) as a convex combination of the extreme points. The interior

point constitutes the dual variables and is used as base for calculating reduced costs in the column generation procedure. Figure 2.5 presents an example of the stabilization method, where the interior point is calculated as the median of the extreme points.

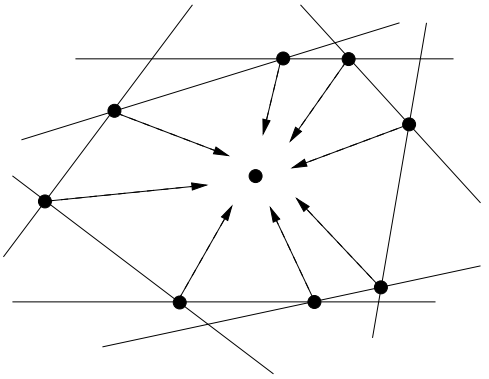


Figure 2.5: An example of several extreme points in the dual solution space and an interior point calculated as the median of the extreme points.

To properly illustrate the interior point stabilization method we introduce the *Set Partitioning*-problem as an example:

$$\min \quad \sum_{r \subseteq R} c_r x_r \tag{2.27}$$

$$\text{s. t.} \quad \sum_{r \subseteq R} a_{ir} x_r \geq 1 \quad \forall i \in \{1, \ldots, N\} \tag{2.28}$$

$$x_r \in \{0, 1\} \quad \forall r \subseteq R \tag{2.29}$$

The problem is $\mathcal{NP}$-hard and consists of finding the cheapest way of choosing sets $r \in R$ such that all elements $i \in \{1, \ldots, N\}$ are covered, see Cormen et al. [52]. The variables are LP-relaxed into $x_r \geq 0$ and the LP-relaxed problem is denoted the Master Problem (M). The dual problem (D) is:

$$\max \quad \sum_{i \in \{1,\ldots,N\}} \lambda_i \tag{2.30}$$

$$\text{s. t} \quad \sum_{i \in \{1,\ldots,N\}} \lambda_i a_{ir} \leq c_r \quad \forall r \subseteq R \tag{2.31}$$

$$\lambda_i \geq 0 \quad \forall r \subseteq R \tag{2.32}$$

The dual variable for constraint i in (M) is denoted λ_i. When the primal problem is solved, the set R^* contains the sets r in the current solution, i.e., with $x_r > 0$. The set S contains the elements for which the constraints (2.28) are not tight. Using the definition

on complimentary slackness condition [204], the dual solution space containing all optimal values for λ is defined as:

$$\begin{aligned}
\sum_{i\in\{1,\ldots,N\}} \lambda_i a_{ir} &\leq c_r \quad && \forall r \subseteq R\backslash R^* \\
\sum_{i\in\{1,\ldots,N\}} \lambda_i a_{ir} &= c_r \quad && \forall r \subseteq R^* \\
\lambda_i &= 0 && \forall i \in S \\
\lambda_i &\geq 0 && \forall i \in \{1,\ldots,N\}\backslash S
\end{aligned}$$

This also defines the constraints in the stabilized dual problem (SD). The objective function in the (SD) is manipulated into giving different extreme points in the dual solution space. Let u be a vector containing random real numbers in the interval $[0, 1]$. (SD) is defined as:

$$\begin{aligned}
\max \quad & \sum_{i\in\{1,\ldots,N\}} u_i\lambda_i \\
\text{s. t} \quad & \sum_{i\in\{1,\ldots,N\}} \lambda_i a_{ir} \leq c_r \quad && \forall r \subseteq R\backslash R^* \\
& \sum_{i\in\{1,\ldots,N\}} \lambda_i a_{ir} = c_r \quad && \forall r \subseteq R^* \\
& \lambda_i = 0 && \forall i \in S \\
& \lambda_y \geq 0 && \forall i \in \{1,\ldots,N\}\backslash S
\end{aligned}$$

Solving (SD) for different vectors u gives extreme points in the dual solution space. For each vector u, (SD) can also be solved for the corresponding $-u$ to reach extreme points far from each other. The dual solution space is convex; hence any convex combination of extreme points yields an interior point lying within the dual solution space. The number of extreme points to generate varies from problem to problem; however, Rousseau et al. [170] argued that 20 points suffice.

Instead of solving (SD) for each extreme point, the dual of (SD) can be generated. Let (PD) denote the dual problem of (SD). (PD) is very similar to the master problem (M) and (PD) can easily be formulated:

$$
\begin{aligned}
\min \quad & \sum_{r \subseteq R} c_r x_r & & \\
\text{s. t.} \quad & \sum_{r \subseteq R} a_{ir} x_r \geq u_i & & \forall i \in \{1, \ldots, N\} \backslash S \\
& \sum_{r \subseteq R} a_{ir} x_r \geq -\infty & & \forall i \in S \\
& x_r \geq 0 & & \forall r \subseteq R \backslash R^* \\
& x_r \in \mathbb{R} & & \forall r \subseteq R^*
\end{aligned}
$$

It may not be necessary to modify all right hand sides. Hence (PD) may be solved faster by re-optimizing the master problem (M) with modified right hand sides.

2.2 Greedy heuristics

Heuristics are often based on ”rules of thumb” which lead to a solution that hopefully is close to the optimum. Heuristics generally have small running times but give no guarantee on the solution quality. See e.g. Judea [113] or Michalewica and Fogel [149] for details on heuristic methods.

Heuristics are applied to the grid scheduling problem to investigate if they provide a satisfying alternative to exact solution approaches. Only greedy heuristics are considered, because preliminary work pointed towards mediocre results when applying more sophisticated meta-heuristics to the grid scheduling problem [79].

Greedy heuristics choose the next step from what appears to be best right now: the heuristics make a locally optimal choice in hopes of reaching an overall good or possibly optimal solution. Generally, greedy heuristics work as follows:

- A candidate set of feasible choices is found
- A candidate is selected greedily
- A new solution is generated
- The solution value is calculated

These steps can be repeated according to some criteria. Greedy heuristics do generally not work exhaustively on all combinations of selections and thus do not necessarily

find optimal solutions. Early decisions may prevent finding the overall best solution later.

EXAMPLE: Consider a problem consisting of scheduling jobs on machines such that the overall profit of scheduled jobs is maximized. A greedy heuristic assigns the next job on the ”best” available machine. The greedy steps are:

- Find all available machines, the job can be assigned to
- The available machine with smallest possible time slot is chosen
- The job is assigned to the chosen machine
- Add the job profit to the solution value

These steps are repeated until we have tried to assign all jobs for execution.

Greedy heuristics are typically applied to $\mathcal{NP}$-hard problems or to polynomial problems, which are difficult to solve, e.g., because of the problem instance size or because of modeling issues. Some polynomial problems may be solved to optimality using greedy choices. These problems typically have in common that a locally optimal choice is also globally optimal (*greedy-choice property*) and that an optimal solution contains optimal solutions to subproblems (*optimal substructure*):

- Greedy-choice property emphasizes why the greedy approach differs from dynamic programming. A greedy choice may be based on steps up until now but cannot depend on future choices or future subproblems. Dynamic programming consists of solving subproblems in a bottom-up fashion until the overall problem has been solved, where the greedy approach can be viewed as solving the overall problem in a top-down fashion.
- Optimal substructure is an important property both in the context of greedy algorithms and of dynamic programming. When a problem has optimal substructure, it can be split into subproblems. In dynamic programming, optimal solutions for the subproblems are eventually gathered into an overall optimal solution. A greedy approach iteratively extends a (sub) solution by greedily solving the next subproblem.

Proving that a greedy approach solves certain problems to optimality is not necessarily trivial; Cormen et al. [52] propose to use theory on independent matroids as a proof for several problems. Examples are Prim’s and Kruskal’s algorithms for minimum spanning trees.

2.3 Summary

This chapter described the solution methods used in this thesis. The exact solution methods are relevant in chapters 4, 6, 9 and 10 and the greedy heuristics in chapters 3, 6, 9 and 10.

The exact solution methods were all based on the branch-and-bound algorithm and were extended with Dantzig-Wolfe decomposition, column generation, cutting planes and/or stabilized column generation. Dantzig-Wolfe decomposition and column generation were presented along with examples of how to apply both on mathematical formulations. Cutting planes were briefly introduced along with a discussion on benefits and drawbacks of adding cuts to a mathematical formulation. The necessity of branching was described along with a discussion on what a good branching strategy is. Stabilized column generation was presented as a method to stabilize the values of dual variables. A dual variable was said to be stabilized when its value did not fluctuate from iteration to iteration in a branch-and-price scheme. The presented stabilization method was based on finding several extreme points in the dual solution space and then taking the average of these extreme points. The resulting interior point defined the values of the dual variables.

Greedy heuristics were shown to generally consisting of four steps: identifying feasible candidate set, selecting a candidate, generating a solution, and calculating the solution value. Also the expected complexity and applications of greedy heuristics were discussed.

Part II

Scheduling in grid computing

CHAPTER 3

Introduction to the scheduling problem in grid computing

Grid Computing is the name of a service which provides applications, storage and computational power. The idea behind grid computing was that users could access the grid by plugging their computer into a grid plug in the wall of their house; similar to the way we get electricity by plugging an electric device to the power grid. Grid computing is hence named after the power grid. When using the grid, requirements for the home computer were to be low: software applications, storage and computational power were received from the grid. The home computer only needed to support a high quality internet connection and the display of graphics.

The full vision of grid computing has not been implemented at this point of time. Most grid computing systems currently work as computational power for researchers who wish to run data and computationally heavy jobs. A grid can hence be viewed as a number of computer resources from different administrative domains working together for solving large problems. Here the size of a problem is typically measured in the number of needed CPU cycles or in the amount of needed data.

This chapter is organized as follows. First, a short introduction to grid computing is given in Section 3.1. Then in Section 3.2 the problem of scheduling jobs on re-

sources in grid computing is presented. The scheduling problem is considered to be either offline or online. Network topology is important when considering a distributed system like grid computing. Section 3.3 discusses the topologies considered in this thesis. Next, the motivation for considering scheduling in grid computing is discussed in Section 3.4. Then the contribution on grid scheduling of this thesis is presented in Section 3.5. Finally, the chapter ends in Section 3.6 with suggestions for future work on the scheduling problem in grid computing.

3.1 An introduction to grid computing

A grid system consists of a number of computer resources and grid servers which are connected through a network, e.g., the internet. The grid hence differs from super computers, because the latter consists of a number of CPUs sharing a local computer bus. The grid also differs from a cluster of computers, because a cluster is connected through a local area network. Finally, grid computing differs from Cloud Computing; the architecture of a grid system is defined to be resources connected to grid servers through a network, where the architecture of a cloud is more far-stretched and can be a grid, a cluster, a supercomputer etc. A general illustration of a grid system is seen in Figure 3.1; though the figure illustrates a specific grid system - the Minimum intrusion Grid (MiG) - it applies to the general grid system because it consists of resources, grid servers (illustrated as a grid cloud), and a number of users of the grid. A resource may consist of several computers, which are administrated locally; this is denoted a Virtual Organization. Many grids actually consist of VOs, which work together and share their competences and resources. Similarly, many grids require their users to be VOs. We, however, view a grid as consisting of users, grid servers and resources unless otherwise stated.

The software of a grid system is typically divided into two classes; the software or *middleware* on grid servers and resources and the software on the user side. The middleware enables sharing of resources, scheduling of jobs, transmission of data, storage of data, and all other activity in the grid. The software on the user side enables the user to log on to the grid. Countless middleware implementations exist; some of the larger projects include `Globus`, `gLite`, and `ARC`. The software on the user side matches the middleware of the corresponding grid and typically supports a secure connection to the grid, upload of job requests and data, and reception of result files. For more details on grid computing in general, including technical descriptions and an overview of existing grid projects, we refer to the survey paper of Baker et al. [20]. In the following we briefly introduce a few grid computing systems: the Worldwide Large Hadron Collider Grid, SETI@HOME, Nordic DataGrid Facility and the Minimum intrusion Grid.

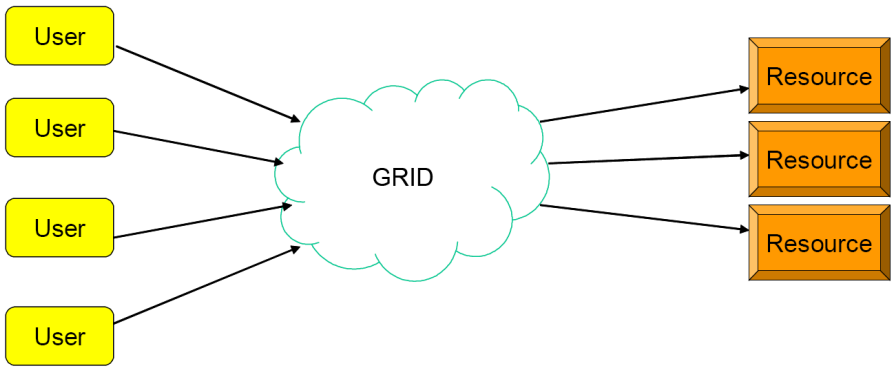


Figure 3.1: An abstract model of the Minimum intrusion Grid taken from Andersen and Vinter [10].

The World Wide Large Hadron Collider Grid (WLCG) is a project which aims to handle the massive amount of data generated by the Large Hadron Collider (LHC). The European Organization for Nuclear Research (CERN) currently works on testing different predictions of high-energy physics, including the hypothesized Higgs boson. The project is expected to generate 15 petabytes of data annually, thus grid computing is used for not only distributing the scientific work on the data but also for distributing storage and back-up of produced data. The WLCG consists of hundreds of VOs all over the world and uses gLite as middleware. For more information on the WLCG, we refer to Shiers [176] or the project homepage [41].

The SETI@home (seti-at-home) project consists of a ”Search for Extraterrestrial Intelligence (SETI)” by analyzing radio telescope data. The available computational power limits the frequency range and the sensitivity of the search, because the search results in very large amounts of data. Hence interested parts can download a program, which uses network bandwidth and computer CPU and disk to analyze radio telescope data. The user may control the amount of bandwidth, CPU and disk to be used on the project and when the calculations may be performed. By enabling outside parts to help with analyzing data, larger searches can be performed. The SETI@home project comes from the Space Science Laboratory at the University of California, Berkeley, and was launched in 1999. The middleware is BOINC (Berkeley Open Infrastructure Network Computing, see e.g. Anderson [11]), which is also used by later projects like Folding@home (protein folding and other disease research problems, see Beberg et al. [32]), ABC@home (mathematical computations, see the project homepage [2]), FightAIDS@home (HIV/AIDS research, see Chang et al. [44]), etc. [1] For more information

[1] It is argued that projects using the BOINC middleware are not grids but instead Public Resource Computing (PRC) systems. In PRC the idea is that anybody with an internet connection donates CPU cycles on

on the SETI@home project, we refer to Anderson et al. [12] and the material of the project's homepage [175].

The Nordic DataGrid Facility (NDGF) is a grid in the Nordic countries based on collaboration between Denmark, Norway, Sweden and Finland. NDGF is used in research context; currently the main purpose of the grid is to assist in computations for the Large Hadron Collider (LHC) project by CERN. The NDGF is closely related to the NorduGrid project but is more operational orientated, where NorduGrid has focus on development. The current NDGF topology is illustrated in Figure 3.2. The links between the countries are hosted by NORDUnet. NDGF mainly uses NorduGrid's ARC (Advanced Resource Connector) middleware. Only Virtual Organizations (VOs) can

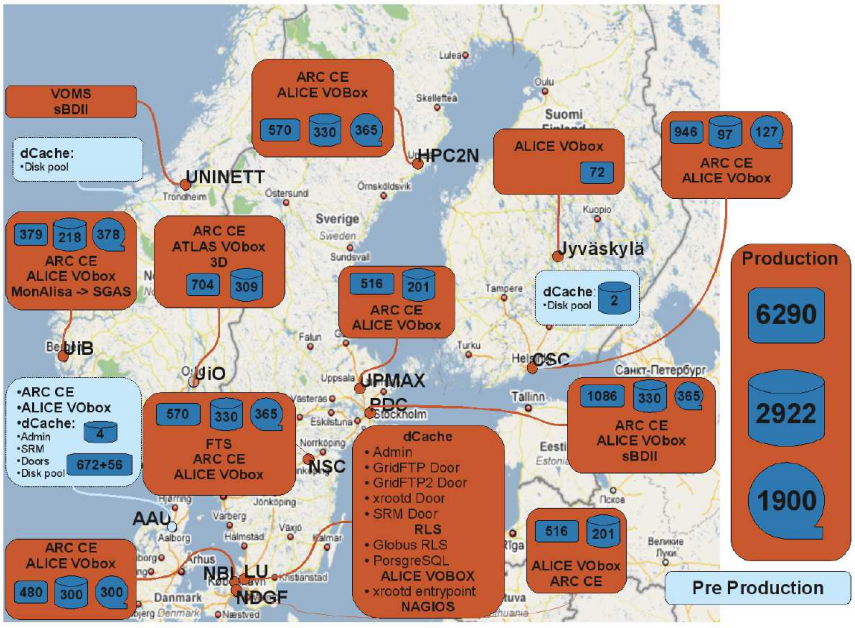


Figure 3.2: A model of the Nordic DataGrid Facility displaying CPU, Disk and Tape availability at each site.

gain access to NDGF. The ARC middleware provides resource discovery for each VO, which then - and not the automated scheduler - places its jobs on appropriate resources. For more information on NDGF, we refer to their homepage [154], the presentation of Grønager [96] and the work of Fischer et al. [70].

their computer for a larger project. PRC systems are said to be much more unreliable and unstable than grids because of the uncertainty of the resources, see Neves et al. [153]. Though we recognize this difference, we mention SETI@home project in this section because it is a well-known project very closely related to grid computing.

The Minimum intrusion Grid (MiG) also provides researchers with computational power. MiG is illustrated in Figure 3.1 and its main purpose is to support complex computations for researchers in Denmark. The idea behind MiG is to minimize software and middleware requirements on the user and resource sides. A user logs on to the grid through a secure web interface and identifies herself with a small certificate file. Resources need to be registered at the grid and log on to the grid via a secure shell (SSH) tunnel using a small certificate for security reasons. Grid middleware such as Globus [75] and NorduGrid ARC [63] require users and resources to install large amount of software to use the overlying grid. MiG tries to avoid that by requiring as little as possible from users and resources; hence the name *Minimum intrusion* Grid. The functionality of the MiG can roughly be described as:

1. A user sends a job request to the MiG server, which puts the job on queue
2. A resource requests a new job to execute
3. The grid server creates a job script from a job on queue and sends the job script to the resource
4. The resource starts the job script
5. The resource requests the needed input files
6. The job is being executed once input files have arrived
7. The resource sends output files to the grid server

When the grid server creates a job script, it has decided which job to send to the resource. This decision is currently based on a greedy *first come first serve* approach, see Sørensen [181]. The job assignment method of MiG is an online algorithm, which schedules job execution every time a resource signals its availability. It does not take time spent on data transmission into account. For more information on the Minimum intrusion Grid, we refer to Andersen and Vinter [10] and Vinter [201].

3.2 Scheduling in grid computing

Most middleware either supports job scheduling or can easily integrate job scheduling into its functionality. In the remainder of this chapter we thus assume that the middleware supports scheduling.

Scheduling in grid computing consists of assigning jobs to resources such that all job demand arrives before job execution begins and such that network constraints are satisfied. Most grids hold three schedulers:

- A global grid scheduler, which assigns job requests to resources.
- A local queue scheduler at a resource, which assigns job requests locally on the resource. This is especially relevant, when the resource consists of several CPU's, e.g., the resource is a cluster or a super computer.
- The user, who submits jobs.

The local scheduler is locally administrated and out of scope for the grid system. For this reason, we do not consider local schedulers. User behaviour may delay grid performance, e.g., when users submit erroneous job requests or jobs, i.e., where the request does not correspond to the actual job. We do not take this into account but instead assume correct user behaviour. Some global grid schedulers does not assign job requests to resources, but suggests a number of available resources to the user, who then makes the decision on which resource to assign the job to. In this case the user also acts as a scheduler. We assume that the global grid scheduler assigns jobs to resources instead of only suggesting available resources to the user. Hence the user is left out of scheduling decisions.

In this thesis a number of assumptions are made in the scheduling algorithms. First, only one grid server is assumed. Redefining how grid servers communicate is out of scope, hence we assume that all resources are connected to one grid server. The assumption may introduce some inaccuracy to the schedules, because latency times on copying job requests and exchanging information on available resources between grid servers are not taken into account by the solution algorithms. We, however, try to compensate for this by including extra time buffers between job executions in the schedules. Job data can be stored on resources and on grid servers. To simplify the offline problem we consider the grid server to be a resource, which can never execute jobs. A last assumption is that we assume that only job data is sent between resources. Job requests and job result files are generally small, thus it is fair to ignore them.

Users submit job requests to the grid server, where job requests are queued. Each job request must include information on the submission time and the latest execution time - together this forms the time window of the job. Each job request must also hold information of the needed input files, i.e., a list of required job files, their size and their position. Finally, each job request must hold an estimate of required CPU time for execution.

When a resource signals its availability, it provides information on how much CPU time is available and on available bandwidth to and from the resource. Job execution cannot take place before all data files are present at the executing resource. All data transfers must satisfy bandwidth limitations.

We only consider job requests, data transmissions between resources, and job executions on resources. Hence users are left out of the scheduling problem.

Data is copied to a resource before job execution begins. Two different data storage approaches can be considered:

- Staging: copied data is deleted when job execution finishes
- Data replication: copied data is saved even after job execution

Replicating data may lead to more jobs being executed and to a smaller network load, but job starvation may eventually occur. Jobs using the same data may end up being executed before jobs requiring rarely used data. Also replicating data may require more storage than a resource is capable of providing. In this thesis we assume that data is deleted after job execution, i.e., staging.

The grid scheduling problem can be divided into two categories: offline and online. In the former full knowledge on future activity in the grid is assumed and a plan for job execution is calculated prior to the start of any activity. In the latter there is no knowledge on future activity in the grid, hence job execution is determined at job arrival time or when a resource becomes available. In this section we describe both scheduling scenarios and give an overview of relevant work in the literature.

3.2.1 Offline scheduling in grid computing

An offline scheduling algorithm in grid computing determines all grid activity in advance. Hence the algorithm assumes full knowledge on resource and job availability, deadlines for job execution, bandwidth limitations etc.

It is interesting to investigate offline grid scheduling because of its many applications. An offline scheduling algorithm can be used to empty a queue of jobs; a procedure which grid administrators may deem necessary from time to time to improve overall grid performance and to avoid job starvation. Another application is advance reservation, where grid resources are reserved in advance for a number of planned jobs. Finally an offline algorithm provides an excellent strategic planning tool, where grid administrators can analyze the effects of changes to the system. If the grid administrators for instance wish to upgrade the grid, then an offline algorithm can help them decide whether the grid needs more CPU power, better network connections, more storage etc.

The offline grid scheduling problem has not been given much attention in the literature. This is probably due to its more analytic nature, where the online algorithm can be

directly applied in a day-to-day use in grid computing. Marchal et al. [142] considered the problem of sharing bandwidths in the context of grid computing. A given set of data transmissions with time windows are to be routed through a network. Marchal et al. proved that the problem is $\mathcal{NP}$-hard and proposed a number of greedy heuristics.

Agarwal et al. [3] proposed an offline scheduler, where jobs first are scheduled to resources such that the total penalty of delayed job executions is minimized. Then data availability and data transfer is considered to decide the final schedule.

A tabu-search algorithm was proposed by Elghirani et al. [64]. The algorithm searches through solutions by moving an executed job from one resource to another. If a move is repeated often, then it is penalized to avoid cycles. If the solution has not been improved in a given time interval, then the tabu list of penalized moves is cleared, a random solution is found, and the algorithm starts over.

Varvaigos et al. [196] worked on Advance Reservations in grid computing, but only proposed an algorithm for reserving network resources for a job. For a given data transmission, the authors found all optimal paths and then selected the ”best” path according to a multi-cost objective and to available network resources. Because only one job is considered at a time, it can be argued that their algorithm is not particular offline.

The offline scheduling problem assumes full knowledge of the system in advance. A problem instance thus includes the job requests queued on the grid server. Each job request holds a time window, bounded by the submission and latest execution times, and an estimated execution time. Each job request also holds a list of needed files, their size and their position. A problem instance must also hold information on when each resource is available and on bandwidth availabilities. Bandwidths may vary over time.

The offline scheduling problem is proved $\mathcal{NP}$-hard by reduction from the $\mathcal{NP}$-hard *knapsack problem*. In the latter, a knapsack and a set of items with profits are given. The goal is to pack items into the knapsack such that the total profit of packed items is maximized. For a survey of the knapsack problem and corresponding solution methods, see Kellerer et al. [118] and Pisinger [160]. Packing items in a knapsack is equivalent to assigning jobs to a resource such that the total profit of executed jobs is maximized. Hence, a solution to the offline scheduling problem, where no data files are to be transferred, is applicable to the knapsack problem. The offline scheduling problem is thus $\mathcal{NP}$-hard.

3.2.2 Online scheduling in Grid Computing

An online scheduler in grid computing decides job execution when a job request is submitted or when a resource signals its availability. The former is typically the case when at least one resource is available and only few jobs are on queue. The latter is the case when several jobs are on queue and when resources are busy. The online scheduling problem only knows the state of the grid when scheduling takes place. It has no knowledge on future activity or on future bandwidth availabilities.

Online scheduling is interesting because it typically constitutes the main functionality of grid systems. Almost every grid middleware has its own scheduler. The schedulers vary from having full control over which jobs are assigned to which resources, to only proposing a number of candidate resources to the user, who then decides where to submit the job. This thesis does not consider online scheduling in grid computing, but because online scheduling is a large research field and an important functionality in most grid systems, this section is dedicated to giving an overview of important results from the literature.

The online scheduling of jobs on several resources is a well-studied problem in the literature. In the following we only consider work performed on the scheduling problem, where data transmission is taken into account. Much work has been performed on online job scheduling and data replication in grid computing, where scheduling and data replication are treated as two separate problems, see e.g. Avellino et al. [17], Bjerke [39], Chrabakh and Wolski [49], Foster and Kesselman [76], Jiang and Yang [112], Meyer et al. [148], Schintke and Reinefeld [172], and Weng et al. [203]. More recent work, however, also focuses on performing online job scheduling where data replication is taken into account when scheduling.

Thain et al. [186] presented the idea of binding execution and storage sites together into I/O communities reflecting physical reality. Computations should be performed mainly within a community, where job requirements are present. A simulation study where communities are built by hand is performed. The study shows that the communities improve the throughput of the grid system.

Ranganathan and Foster [165] considered scheduling jobs with respect to data replication and data availability. Using a number of known scheduling and data replication strategies, they investigate the performance when taking data location and network limitations into account. Their results show that assigning jobs to resources according to job data location and replicating often used data increase grid utilization.

Chakrabarti et al. [42] proposed an integrated data replication and job scheduling strategy. The scheduler takes the number of missing data files at each site into account. Data

replication is performed at a given interval of time. Their results show that best performance is reached when taking data replication costs into account when scheduling job execution.

Cameron et al. [40] performed simulation studies where both job scheduling and data replication are considered. Data replication is performed whenever a job is to be scheduled. Their results show that the scheduling strategy taking both data access and CPU costs into account has best performance.

Baranovski et al. [24] presented a scheduler for the SAMGrid which uses the Condor-G technology. Previously, users had to assign their jobs to resources with no help from the Grid. The proposed scheduler decides where to execute jobs based on a matchmaking framework which takes CPU workload and job data files into account. Network capacities and data replication are not considered. Initial simulation results show that the proposed scheduler mainly assigns job to resources with most data cached.

Thysebart et al. [188] considered simultaneous data transmission and job execution where network capacities are taken into account. Grid sites are connected through VPNs. Three scheduling strategies are considered: *non-network aware*, *network aware* and *prefer local*. All considers CPU workload and storage capacities, the network aware also takes bandwidth limitation into account, and the prefer local strategy furthermore tries to assign jobs to sites, where most or all job data is stored. Simulation results show that the network aware and prefer local strategies by far have best performance.

Related work by Volckaert et al. [202] proposed more scheduling approaches. The first is the `minimum hop count` which schedules a job to the site where the number of hop counts used to transmit data is minimized. Jobs tend to get assigned to sites with most or all data stored. The second is the `service differentiation approach` which analyzes each job, classifies each job as either data or computational heavy and assigns the job according to its classification. Results once again show that the network aware scheduling strategies have best performance.

Huang et al. [100] proposed job and data co-scheduling algorithms. Their approach is based on a number of steps: first data replication takes place, then jobs are assigned to resources and finally job data files not present at the executing resource are transmitted. Data transmission is allowed to take place while the corresponding or any other job is being computed. Computational results show promising behaviour and reveal that job features and data sizes influence the performance of the grid.

Tang et al. [184] suggested a number of data replication strategies and compare their performance when using three different job scheduling methods. The replication strategies build on historic data usage and replication is performed at a given time interval. The centralized data replication strategy counts the number of times each file has

been accessed, replicates the most accessed files and places replica at the server with enough storage, least CPU workload and best bandwidth. The distributed data replication strategy replicates most accessed data files and places replica at local servers with sufficient storage capacity. The grid scheduling strategies are based on `shortest turnaround time` where queuing, data transfer and execution times are taken into account, `least relative load` where CPU workload is taken into account, and `data present` where job data location is considered. Computational evaluations show that dynamic data replication improves performance and that especially `shortest turnaround time` with centralized data replication performs well. Related work was presented by Tang et al. [185] where the data replication strategies differ slightly: data is replicated to being as close as possible to the requesting jobs. Again test results show that dynamic data replication improves performance.

Veenugopal [198] proposed network aware scheduling where data presence and resource usage cost among others are taken into account. Taking these two factors into account improved performance compared to traditional greedy scheduling approaches. Also, a scheduler which either tries to minimize execution or data transfer time is presented. The scheduler is economy based and provided promising results.

McClatchey et al. [147] proposed a scheduler, DIANA, which takes CPU workloads and network limitations into account. The scheduler calculates costs for job execution and data transmission and schedules a job based on the sum of these. Allocation of weights on the costs is possible in the scheduler. It may be beneficial to add different weights to different parts of the grid in order to reach more appropriate performance. E.g., if a job is very data heavy then weights can assure that it is assigned to the grid resource storing most job data. Computational results show that DIANA performs very well with respect to execution times and network load. For a thorough description, analysis and discussion of DIANA we refer to the PhD thesis of Anjum [13]. The thesis also discusses scheduling in grid computing in general.

Chang et al. [45] proposed a scheduler for a cluster grid where data transmission and network capacities are taken into account. The scheduler defines a cost function at each cluster based on CPU workload, data availability, network capacities within the cluster, and network capacities on connections to the cluster. A job is assigned to the cluster with smallest cost. Once a job is assigned to a cluster, data replication is performed. The proposed scheduler is tested on the OptorSim simulator. Test results show that performance is improved when using the proposed scheduler. However, it tends to assign the same type of jobs to the same cluster, which may lead to load balancing problems. Finally, Chang et al. provides a detailed overview of work performed on job scheduling and data replication.

Dang and Lim [53] proposed a data replication method based on job placement. It calculates the number of times a data file is requested and then replicates the most requested files. For each replica and each site, the distances between the requests and

the sites are calculated. The data replica is placed at the site with smallest total distance. Initial simulation using OptorSim shows promising results; the replication strategy performs better than Random, Least Recently Used (LRU) and Least Frequently Used (LFU) replication strategies.

Baraglia et al. [23] proposed a heuristic which tries to fulfill the CPU requirements and tries to exploit the parts of the grid network with high bandwidths. Costs are added to the needed CPU time and to data transfer for each request on queue. The heuristic considers several job requests at a time and groups requests together according to how many data files they share. Resources are grouped according to network availability. Finally, each group of requests is scheduled on a suitable group of resources. Simulation studies show that the heuristic performs well and is a viable scheduling approach.

Ferrandiz and Marangozova [68] analyzed existing scheduling and replica policies in the Large Hadron Collider (LHC) grid. When a job arrives, the resource broker assigns the job to the grid site with shortest job queue. The grid site requests job files from the storage optimizer which fetches each file and uploads a replica to the grid site unless the file is already available. Ferrandiz and Marangozova implemented a simulator LCGSim to simulate LHC activity. Three different replication strategies are implemented: no advance replication, LRU and LFU. They compared with simulations obtained through OptorSim. Their results do not show any clear pattern for which replication strategy is more beneficial.

Kokkinos et al. [124] propose algorithms for determining how to perform data replication given a job schedule. Their algorithms take base in random procedures, transmission costs, job execution costs or a combination of the two latter. Simulation results show that the best results with respect to network load and job delay are reached when taking transmission costs into account.

Abawajy [1] considers a grid system, where each grid resource is a cluster. The proposed scheduling algorithm focuses on assigning a group of jobs sharing the same data files to the same cluster or clusters. Simulation results show that it is beneficial to consider the I/O requirements of jobs when scheduling. Data replication strategies and data distribution were not considered.

3.3 Network topology

Grid computing is a distributed system, hence it is important to consider the underlying network topology. So far we have assumed that the grid components are connected through a packet switched network, where we have no influence on the specific path used between two components. An example of this is the internet. The considered

bandwidth constraints cover upload, download and connection speed. Hence all grid components can be viewed as being directly connected and the data transmission problem then becomes to find a time to transmit data, such that the considered bandwidth constraints are satisfied.

Another topology to consider stems from optical networks. An optical fiber carries data at a certain wavelength. Using the wavelength-division multiplexing technology, a fiber can transmit data at several wavelengths. However, two data connections cannot share the same wavelength on a fiber due to network switch limitations. Transmitting data through an optical network corresponds to the $\mathcal{NP}$-hard Routing and Wavelength Assignment Problem (RWA). In this section, we discuss the benefits of using an optical network, the RWA problem in general and why it is relevant in grid computing context. Note that an overview of work performed on the RWA in the literature is left out because the second contributed paper in this part is a survey on the RWA.

3.3.1 Optical networks

In telecommunication an increasing part of the network infrastructure consists of optical fibers. An optical fiber is predominantly made of glass and carries light along its length at high speed and with little loss. Benefits of using optical fibers instead of the traditional copper telephone lines in a Wide Area Network (WAN) include:

- Optical fibers support much larger bandwidth speeds than copper lines.
- For high frequency transmissions, light signals can travel further than copper lines before being amplified.
- Optical fibers and copper lines have the same cost.

Several wavelengths on a single fiber can be used to transmit light signals, i.e., data, by using the wavelength-division multiplexing (WDM) technology. Furthermore, an optical network is a *circuit switched* network, i.e., the route of a data connection is established before data can be sent, see Halsall [98], Thiele and Nebeling [187] or the thesis of Jue [114] for more details.

In this thesis we consider an abstract model of the optical network and thus omit technical details. The optical network is considered to be a graph consisting of nodes and edges. Edges represent fibers and each fiber is capable of holding several wavelengths. A node represents any active equipment with at least one ingoing and/or outgoing edge. This could be a switch, a hub, an amplifier etc.

3.3.2 The Routing and Wavelength Assignment problem

RWA is the problem of finding a good way of establishing data connections and of assigning wavelengths to the connections. A request for a data connection consists of a source and a target. A route and wavelength(s) are to be found between the source and the target nodes. In RWA, paths of different data connections are to be generated such that no two paths share the same wavelength on an edge. That is, two paths using the same wavelength must be edge disjoint.

Variants of the RWA include limitations on wavelength conversion and on the lifetime of connections. Wavelengths can be convertible in all nodes, in a subset of nodes and in no nodes at all. The first version is denoted the wavelength convertible RWA as each data connection can use different wavelengths on all its edges, see Ramamurthy and Mukherjee [163]. The second is denoted sparse wavelength convertible RWA, see Iness and Mukherjee [104]. For both versions further constraints can be set on the range of possible conversions, i.e. a wavelength l_i can be converted into wavelengths $l_{i-k}, \ldots, l_i, \ldots, l_{i+k}$ for some nonnegative integer k. For more information on the limited-range wavelength converters, see the work of Iness and Mukherjee [104] or of Yates et al. [205]. Being able to convert wavelengths, though, does not necessarily increase the number of established data connections, see Jaumard et al. [108].

The lifetime of data connections is either permanent or temporary. The *static* RWA reserves routes and wavelengths for all future data connections and thus assumes full knowledge on all future activity in the optical network. Data connections are assumed to last "forever". Chlamtac et al. [47] shows that the static RWA is $\mathcal{NP}$-hard. The *dynamic* RWA only reserves routes and wavelengths for data connections when needed. The route and wavelength is released once the data connection is no longer needed. Thus only knowledge of the current state of the optical network is needed. Because no knowledge exists on future data connection requests, solutions to the dynamic RWA are local optimums. For an overview of both the static and the dynamic RWA, see Zang et al. [206].

It may not always be possible to establish all data connections in the static RWA. If a connection cannot be established, it is said to be *blocked*. The objective of the static RWA is typically to maximize the number of established data connections or the profit of established data connections.

Data connections may also be blocked in the dynamic RWA, thus the objective can also be to maximize the number of established data connections or the profit of established data connections. Because data connections do not last forever, wavelengths may be reused. The objective can thus also be to minimize the number of used wavelengths.

3.3.3 Optical networks in practice

Both the dynamic and the static RWA are applicable in telecommunications as explained. The dynamic RWA can be used to establish data connections on the fly. If an optical network is to be used as an infrastructure in e.g. parts of the internet, then the network utilization can be improved by determining paths and wavelengths when needed, rather than having to choose from a set of predetermined paths with fixed wavelengths. The reason for this is that the data load and the data connection requests may differ significantly from time to time.

The static RWA does not necessarily leave room for future connections, so it is mainly applicable when the current data connection requests are either the only requests or have much higher priority than any other requests. If the telecommunication company introduces costs on network usage, then the goal of the static RWA becomes to minimize the total cost. Network costs could depend on the number of used wavelengths, the number of hops used by the connections, the number of available wavelengths on used fibers, the capacity of each wavelength etc. Using one of these objectives, a solution to the static RWA does not only suggest how to route data connections; it can also be used to determine the price customers must pay to get their data connections established.

RWA is also relevant in grid computing context. A project like the Large Hadron Collider (LHC) Physics Program by the European Organization for Nuclear Research (CERN) is very data heavy and thus utilizes grid computing to not only distribute the scientific calculations, but also to store data. The grid computing system of LHC is denoted the Worldwide LHC Computing Grid (WLCG). It is estimated that the LHC experiments generate 15 petabytes of data annually [41], thus the network connections to and from WLCG must support large bandwidths, e.g., be optical.

The main subject of this thesis is to schedule network traffic for grid purposes. As the WLCG example illustrates, the infrastructure in grid computing systems may consist of optical fibers, hence the scheduling algorithms must take network constraints into consideration. Furthermore, it is fair to assume that fibers in large grid computing systems are bought or rented in advance such that network capacities are dedicated to the project.

3.4 Motivation

Integrated scheduling and network routing in grid computing is a relevant and complex problem. As argued in the beginning of this thesis, namely in Chapter 1, data transmis-

sion may constitute a bottleneck, especially when working with data heavy jobs. As grid computing is often used to handle data heavy jobs, transmission time should be taken into account when assigning jobs to resources. If not then jobs may be unnecessarily rejected and the grid may seem unstable to users. Also, grid resources with poor network connections are not utilized properly, if they are assigned data heavy jobs.

Offline scheduling can be applied to job queue emptying. Whenever the job queue on the grid server reaches a certain size, the offline algorithm can be used to compute a plan such that the job queue is emptied. In this way, job starvation is avoided to a great extent and ensuring that practically all jobs are executed increases user satisfiability. The offline algorithm can also be used to give a more homogeneous load in time periods, where the grid is used extensively: jobs with late deadlines can be scheduled for execution at a later time, hence making room for executing jobs, which currently are urgent. The offline algorithm also introduces the ability to plan jobs, i.e., the ability to reserve resources in advance for executing a set of planned jobs. Resource reservation is a powerful tool for researchers to meet deadlines. Also, when using grid computing commercially, resource reservation is used to guarantee customers that their jobs will finish within a certain time period. Finally, the offline algorithm can be used as a tool to analyze grid performance. The offline algorithm is capable of answering questions such as how many jobs can the grid handle within a given time period, what happens to grid performance if a number of extra resources are connected to the grid, how will substituting dedicated high-quality data connections for medium-quality internet connections affect grid performance, etc.

Scheduling of network traffic in grid computing is the main topic of this thesis. We have interpreted this problem as being the integration of job scheduling and data transmission, i.e., the grid scheduling problem. Hence this part constitutes the main contribution of the thesis.

3.5 Contribution

The topics covered in this part are divided into the following papers:

1. Integrated job scheduling and network routing.
2. A survey of the routing and wavelength assignment problem.
3. On the integrated job scheduling and constrained network routing problem.

4. Analysis of internal network requirements for the distributed Nordic Tier-1.

In the following each paper is summarized and discussed. It is noted that work has also been performed on heuristics for the offline scheduling problem in grid computing using a packet switched network topology. The data transmission problem was solved using a greedy multi-commodity network flow heuristic. This was integrated into job placement strategies resulting in four greedy heuristics. Furthermore, a local search meta-heuristic and an adaptive large neighbourhood search meta-heuristic were considered. Unfortunately, the approaches only resulted in mediocre results: solution values were not impressive and running times were at times quite long. Hence the work is not included here but is published as a technical report [79].

The first paper solves the offline scheduling problem in grid computing using a packet switched network to optimality. A branch-and-price algorithm is proposed, where the pricing algorithm assigns a given job on a given resource and makes sure that job data arrives in time. The master problem merges these ”sub schedules” into an overall solution. The branch-and-price algorithm initially has poor performance, because the discrete time representation makes the size of the master problem explode. Hence only violated constraints are included. This significantly improves performance, but the number of generated columns is still quite large. To reduce this, stabilized column generation is applied and the resulting performance is dramatically better. The improved algorithm is capable of solving all benchmark instances in very short time: the largest instance with 1000 jobs and resources is solved within 3 minutes and the far majority of instances are solved in seconds.

The second paper surveys work from the literature on solving the RWA, which mainly consists of heuristics and meta-heuristics. Recently, more work is performed on exact methods for the RWA; however, there is generally a strong emphasis on practical applications hence the former methods are dominating. The survey extends previous surveys in the literature by being updated, by providing theoretical running times on the heuristics, and by discussing test instances and results in much more detail.

The third paper focuses on solving the offline scheduling problem in grid computing where the underlying network topology is optical. As previously explained, this is relevant as an increasingly larger part of networks consists of optical fibers. Especially for larger grid projects, it is very probable that the grid owners rent or buy fibers or wavelengths and thus have an optical network dedicated to the grid project alone. Incorporating optical network constraints into the scheduling problem may increase performance of the grid, both with respect to the amount of transmitted data and scheduled jobs but also with respect to reaching feasible and hence more robust solutions.

The offline scheduling problem in grid computing is $\mathcal{NP}$-hard and not trivial to solve. Adding an optical network topology only complicates the problem further. The paper presents an exact branch-and-price algorithm, which performs better than using CPLEX to solve the mathematical formulation. Because of the complicating network topology, however, the computational results show that the exact branch-and-price algorithm has its limitations. It is capable of solving several of the larger instances, but generally it should only be applied to smaller problem instances because the time usage explodes. Hence work is also performed on solving the problem heuristically by combining heuristics for the offline scheduling problem in grid computing with heuristics for the RWA. Test results show that all heuristics have very good practical running times. The best heuristic setting gives an average solution value gap of less than 3% and solves all instances within minutes. An in-depth analysis of results and problem instance types, however, indicates that a black-box solution method may not always be appropriate. Instead the grid administrator should use knowledge on network and CPU load to choose the best heuristic setting.

The fourth and final paper in this part concerns distribution of network traffic for the Nordic DataGrid Facility (NDGF). In this paper, we apply operations research to a real-life grid. The network topology of NDGF is formalized into a mathematical formulation. Then different scenarios are investigated with respect to the maximal link load. The scenarios are formalized and incorporated in the mathematical formulation, which then is solved to optimality using CPLEX. The main goal of the project is thus not to propose new solution techniques, but to translate requests from NDGF into mathematical representations and to find the best way of distributing network traffic more evenly across the network. Results show that just by re-arranging job, the largest link load is reduced with 900 Mbps - from 4.4 Gbps to 3.5 Gbps. Introducing caches at each grid resource reduces the largest link load by another 500 Mbps. The results are used to change current job placements, to decide how to expand the grid in the future and to give a good overview of where bottlenecks occur in the grid.

3.6 Future directions

We believe that future work on the grid scheduling problem should consider network traffic as an integrated part. In this way the stability of the grid is increased, because more reliable time plans are calculated.

We show that the offline grid scheduling problem using a packet switched network is solvable in little time; hence it is worth considering if the exact algorithm could be integrated in the day-to-day use of the grid - both for supporting advance reservation but also for calculating time plans for execution of jobs on queue. It is even possible

to use offline grid scheduling in a real-time environment: given a queue of jobs and a number of resources, the best execution plan is calculated. Whenever a new job arrives or resource availability changes, the execution plan is re-optimized. In this scenario, the problem instance would probably never become too large for the proposed exact branch-and-price algorithm. Still, we believe that substituting an offline scheduler for the online approach may not happen for a long time to come, because online schedulers are far more easy to implement and maintain, because online algorithms generally have much shorter running time and because grid developers probably are more comfortable with a well-known scheduling approach.

The exact branch-and-price algorithms could be improved by introducing cutting planes. Recent research shows that cuts can improve the performance significantly, even though the overall problem formulation may be complicated, see Desaulniers et al. [57]. The proposed exact algorithm for the offline grid scheduling using a packet switched network performs well, but it is still interesting to investigate heuristic approaches - especially when the problem instance size increases. Early work on heuristics gave large solution gaps, thus future work should consider more sophisticated approaches. For instance, the greedy heuristics could focus more on data placement and schedule jobs to resources close to job data.

We believe that future work on the RWA problem may include incorporating RWA with network design in order to maximize the number of established data connections. Especially in the case of grid computing where the optical network is dedicated for the project, we believe that network design can really boost grid performance. It may not yet be that beneficial to incorporate RWA in a grid computing scheduling algorithm, because the scheduling problem becomes much harder to solve. As the support for WDM grows, however, the stability of future grids may depend greatly on having feasible routing schemes.

For the RWA problem, future work could include approximation algorithms to benefit from a guaranteed solution value reached in not too long time. Also, more work on exact algorithms could focus on taking advantage of the similarities between the RWA and the Multi-commodity unsplittable Flow Problem.

The proposed papers in this part assume that grid activity is controlled globally and exclusively by a grid scheduler. In real-life, however, this is not necessarily the case. As described in the introduction of this chapter, the scheduler in some grids only suggests available resources to the user, who then decides where to execute a job. We believe that the user should be left out of scheduling decisions. Resource utilization could more easily be increased and fairness ensured, when scheduling is performed automatically. Also, it would make the grid more user-friendly, because the user would not have to have an insight in benefits and drawbacks of the grid resources.

CHAPTER 4

Integrated job scheduling and network routing

Mette Gamst
DTU Management Engineering, Technical University of Denmark

David Pisinger
DTU Management Engineering, Technical University of Denmark

We consider an integrated job scheduling and network routing problem which appears in grid computing and production planning. The problem is to schedule a number of jobs on a finite set of machines, such that the overall profit of the executed jobs is maximized. Each job demands a number of resources which have to be sent to the executing machine through a network with limited capacity. A job cannot start before all of its resources have arrived at the machine.

The scheduling problem is formulated as a MIP problem and is proved to be $\mathcal{NP}$-hard. An exact solution approach using Dantzig-Wolfe decomposition is presented. The pricing problem is the linear multicommodity flow problem defined on a time-space network. Branching strategies are presented for the branch-and-price algorithm

In submission 2009

and three heuristics and an exact solution method are implemented for finding a feasible start solution. Finally, interior point stabilization is used to decrease the number of columns generated in the branch-and-price algorithm.

The algorithm is experimentally evaluated on job scheduling instances for a grid network. The Dantzig-Wolfe algorithm with stabilization is clearly superior, being able to solve large instances with 1000 jobs and 1000 machines covering 24 hours of scheduling activity on a grid network. The promising results indicate that it can be used as an actual scheduling algorithm in the grid or as a tool for analyzing grid performance when adding extra machines or jobs.

Key words: Scheduling; Computations Grid; Production Planning; Multicommodity Flow; Dantzig-Wolfe Decomposition;

4.1 Introduction

An exact solution approach for integrated scheduling of jobs and resources in a network is presented. The objective is to schedule a number of jobs on a set of machines, such that the overall profit of the executed jobs is maximized. It is assumed that all jobs are known in advance and that each job demands a set of resources, which has to be sent through a network with limited edge capacities. A job cannot start before all of its resources have arrived at the machine and it must not finish after its due date.

The problem has applications in distributed production systems, where a set of jobs can be carried out at various plants. Each job demands that a set of resources are available. In cases where the resources are bulky and the transportation paths are limited, it is necessary to consider both problems simultaneously. Typical applications are in the steel industry, where the production can be placed at various sites, but the transportation of iron ore and coal constitute a substantial logistic problem.

The problem also has applications in grid computing, where the jobs are programs to be executed at various grid resources and the demands are data needed for running the programs. All components are connected through a Wide Area Network (WAN) and may thus be geographically distributed. A job request consists of a list of required input files and their location: this is denoted the job data. A job request also holds information on how long it approximately takes to compute the job and a deadline for when the job execution must be finished. In grid computing, a grid server maintains a queue of job requests and decides which job to send to a grid resource. In the Minimum intrusion Grid (MiG) [10, 201] the grid server decides where and when to send jobs using a greedy *first come first serve* online algorithm [181], which does not take the

time spent on data transmission into account. Thus, the grid server may allocate a very data heavy job to a grid resource with very poor network connection, which may lead to delays. Moreover, it may result in heavy traffic during the day hours where many jobs are submitted. Scheduling all jobs in advance, i.e. performing offline scheduling, may be more beneficial. Such an offline algorithm must consider all system constraints, i.e., grid resource and job availability, deadlines for job execution, bandwidth limitations etc.

To simplify the problem, we assume that for each connection the bottleneck in the network capacity is determined by the capacity at the end nodes. This is the case in grid computing, where two grid resources establish a VPN connection when transmitting data. In road transportation, the bottleneck is frequently found at the access roads to the highways, and in maritime transportation the ports constitute the bottleneck.

The main contribution of this paper is to model and solve the integrated job scheduling and network routing problem. The model is able to handle time-dependent capacities, e.g., that the network at night may provide larger amounts of available capacity than during the day hours. We suggest an exact solution method based on Dantzig-Wolfe decomposition, where the pricing problem is to assign a single job to a single resource, and where the master problem finds an overall feasible solution for executing jobs. The described pricing problem is a linear multicommodity flow problem. Furthermore, we present a heuristic to reach early feasibility and a branching strategy based on three different constraints. We propose to extend the branch-and-price algorithm by only adding violated constraints in the master problem, which reduces the size of the master problem. Finally, stabilized column generation is added to the branch-and-cut-and-price algorithm to reduce the number of iterations. Computational evaluations show that this makes it possible to solve problems of a larger order of magnitude than previously.

Not much literature exists on the integrated scheduling problem with respect to bandwidth limitations in grid computing. A complexity proof and greedy heuristics for sharing bandwidths in grid computing context are presented by Marchal e.a. [142]. Agarwal e.a. [3] suggest an offline scheduler, where both job execution and data transmission is taken into account. The solution method consists of two steps: first, jobs are scheduled to grid resources such that the total penalty of delayed job executions is minimized. Data availability and transfer costs are taken into account. Then, the overall starting and end times of job schedules are determined. Elghirani e.a. [64] propose a tabu search algorithm, which schedules a queue of jobs. A solution is defined as a set of jobs assigned to a set of grid resources. The neighbourhood of a solution consists of moving a scheduled job to a different, available grid resource. Often used moves are penalized to avoid move cycles. When no improvement has been reached in a certain time interval, the tabu list is cleared, a new random solution is found, and the tabu procedure starts all over. Varvaigos e.a. [196] consider job routing and scheduling to support Advance Reservations in the context of grid computing or Optical Burst

Switching. Advance reservations consist of scheduling data transmissions in a network and the task is to reserve the appropriate network resources. Varvaigos e.a. consider one data transmission request at a time, for which they find all optimal paths and then select the ”best” path according to a multicost objective and to available network resources. The work of Varvaigos e.a. is related to the integrated scheduling of jobs and resources, however, because they only consider one job at a time, their algorithm can be viewed as being an online algorithm.

The integrated job scheduling algorithm can be applied on job queue emptying in a grid network. Whenever the job queue reaches a certain size, the offline algorithm can be used to compute a plan to empty the job queue. In this way, job starvation is avoided. Moreover ensuring that practically all jobs are executed increases user satisfiability. The offline algorithm can also be used to give a more homogeneous load in time periods, where the grid is used extensively: jobs with late deadlines can be scheduled for execution at a later time and in this way make room for executing jobs, which currently are urgent. The offline algorithm also introduces the ability to plan jobs, i.e., the ability to reserve resources in advance for executing a set of planned jobs. Resource reservation is a powerful tool for researchers to meet deadlines. Also, when using grid computing commercially, resource reservation is used to guarantee customers that their jobs will finish within a certain time period. Finally, the offline algorithm can be used as a tool to analyze grid performance. The offline algorithm is capable of answering questions such as how many jobs the grid can handle within a given time period, what happens to grid performance if a number of extra grid resources are added to the grid, how will grid performance be affected when substituting dedicated high-speed data connections for medium-speed internet connections, etc.

The proposed solution algorithms are computationally evaluated on instances reflecting activity in the grid over 24 hours, with up to 1000 jobs and 1000 machines, and time granularity as small as 30 or 15 minutes. The branch-and-cut-and-price algorithm clearly outperforms the original formulation with respect to the size of solved instances. The evaluation reveals that when working with a discrete time representation, small time intervals increase the time spent on finding an optimal solution, but also improve the solution quality. Overall, the branch-and-cut-and-price algorithm can be used as an actual scheduling algorithm for planned jobs or job queue emptying in grid computing and as a tool for analyzing grid performance.

This paper is organized as follows. First, in Section 4.2, the integrated scheduling problem is described in detail, a mathematical formulation is presented and the scheduling problem is proved to be $\mathcal{NP}$-hard. In Section 4.3, Dantzig-Wolfe decomposition is applied and the corresponding branch-and-price algorithm is shown. The algorithm includes methods for solving the pricing problem, several branching strategies and three heuristics and an exact solution method for finding a feasible start solution. Section 4.4 describes how stabilization is used to achieve faster column generation convergence and discusses a number of additional improvements. All the considered algorithms are

computationally evaluated in Section 4.5, and Section 4.6 contains final remarks on the performance of the algorithms and on possible applications of the solution approach.

4.2 Problem description

We use grid terminology to describe the integrated scheduling problem and network routing problem. Furthermore, in the following a grid resource is simply denoted a resource.

The set of jobs is denoted J, the set of resources is R, and the set of connections (edges) is E. We use a discretized time horizon, with time stamps $t \in T$ being a part of the problem instance input. Each job $j \in J$ has a time window $[a_j, b_j]$, the total size of the job data files S_j, the estimated computation time Q_j, the amount of data p_j^r placed on each resource $r \in R$, and a profit $c_j \in \mathbb{R}^+$.

At each time period $t \in T$ each resource $r \in R$ is assigned an availability start time a_r, an end time b_r, and an upper bound on in- and outgoing bandwidth. The upper bound at time $t \in T$ is denoted $d_{r^-}^t$ for ingoing network traffic and $d_{r^+}^t$ for outgoing network traffic.

An edge $(i, k) \in E$ going from resource i to resource k has bandwidth d_{ik}^t at time $t \in T$. It is assumed that all resources are connected.

To simplify notation the time window $[a_{ik}, b_{ik}]$ is introduced, where $a_{ik} = \min\{a_i, a_k\}$ and $b_{ik} = \min\{b_i, b_k\}$, for each $i, k \in R \cup J$. Finally, two sets J_t and R_t are introduced. The set J_t consists of jobs j with $a_j \leq t \leq b_j$. Similarly, the set R_t consists of resources r with $a_r \leq t \leq b_r$.

It is assumed that the only data to be sent between resources is job data. This abstraction is fair, because bandwidth consumption of job requests and job result files is insignificant and can thus be ignored. As bandwidth consumption of job requests is ignored, users submitting job requests are left out of the formal description. Instead, we focus on where and when jobs are to be executed, and whereto and when input data is to be sent. Job execution cannot take place before all input data of the job is copied to the executing resource. The objective of the problem is to maximize the profit of executed jobs.

Now, the model includes two types of variables $x_j^{tr} \in \{0, 1\}$ and $f_{ir}^{tj} \in \mathbb{R}_0^+$. If $x_j^{tr} = 1$ then job $j \in J$ is executed on resource $r \in R$ with execution beginning at time $t \in T$. If $x_j^{tr} = 0$ then the job is not executed on the resource with this beginning time. The

non-negative variable f_{ir}^{tj} denotes the amount of data on edge $(i,r) \in E,\ i,r \in R$ for job $j \in J$ at time $t \in T$. The integrated scheduling problem can now be formulated using an edge-based model (EDGE):

$$\max \quad \sum_{r\in R}\sum_{j\in J}\sum_{t\in[a_{rj},b_{rj}-Q_j]} c_j x_j^{rt} \tag{4.1}$$

$$\text{s.t.} \quad \sum_{r\in R}\sum_{t\in[a_{rj},b_{rj}-Q_j]} x_j^{rt} \le 1 \qquad \forall j \in J \tag{4.2}$$

$$\sum_{j\in J_t}\sum_{i\in R_t\setminus\{r\}} f_{ri}^{tj} \le d_{r+}^t \qquad \forall r \in R,\ \forall t \in [a_r,b_r] \tag{4.3}$$

$$\sum_{j\in J_t}\sum_{i\in R_t\setminus\{r\}} f_{ir}^{tj} \le d_{r-}^t \qquad \forall r \in R,\ \forall t \in [a_r,b_r] \tag{4.4}$$

$$\sum_{j\in J_t} f_{ri}^{tj} \le d_{ri}^t \qquad \forall r,i \in R,\ \forall t \in [a_{ri},b_{ri}] \tag{4.5}$$

$$x_j^{rt} = 1 \Rightarrow \sum_{t'=a_{ij}}^{\min\{b_i,t-1\}} f_{ir}^{t'j} = p_j^i \qquad \begin{array}{l}\forall j \in J, \forall i,r \in R,\\ \forall t \in [a_{rj},b_{rj}-Q_j]\end{array} \tag{4.6}$$

$$x_j^{rt} = 1 \Rightarrow \sum_{\substack{j'\in J\\ j'\neq j}} \sum_{t'=t}^{\min\{t+Q_j,b_{j'r}-Q_{j'}\}} x_{j'}^{rt'} = 0 \qquad \begin{array}{l}\forall j \in J, \forall r \in R,\\ \forall t \in [a_{rj},b_{rj}-Q_j]\end{array} \tag{4.7}$$

$$x_j^{rt} \in \{0,1\} \qquad \forall t \in [a_{rj},b_{rj}-Q_j],\ \forall j \in J,\ \forall r \in R$$

$$f_{ir}^{tj} \ge 0 \qquad \begin{array}{l}\forall j \in J,\ \forall r,i \in R : p_j^i > 0,\\ \forall t \in [a_{irj}, \min(b_i, b_{rj}-Q_j-1)]\end{array}$$

The objective function (4.1) maximizes the profit of the executed jobs. The first constraint (4.2) says that each job can be executed at most once. Constraints (4.3) and (4.4) make sure that in- and outgoing bandwidth limitations are obeyed and constraint (4.5) ensures that connection capacities are obeyed. All job data must be received before execution time (4.6). Constraint (4.7) says that a resource can execute at most one job at a time. The two bounds ensure that variables take on appropriate values. Note, that constraints (4.6) and (4.7) are not linear, but can be rewritten as:

$$\sum_{t'=a_{irj}}^{\min(t-1,b_i)} f_{ir}^{t'j} - p_j^i x_j^{rt} \ge 0, \qquad \forall j \in J,\ \forall t \in [a_{rj},b_{rj}-Q_j], \forall r,i \in R : p_j^i > 0 \tag{4.8}$$

$$\sum_{\substack{j'\in J\\ j'\neq j}} \sum_{t'=t}^{\min\{t+Q_j,b_{j'r}-Q'_j\}} x_{j'}^{rt'} + Q_j x_j^{rt} \le Q_j, \qquad \forall j \in J,\ r \in R,\ t \in [a_{jr},b_{jr}-Q_j] \tag{4.9}$$

The problem is $\mathcal{NP}$-hard by reduction from the *knapsack problem* [118]. Let c' be the capacity of the knapsack and p'_j, w'_j the profit and weight of item j. Construct an

instance of the scheduling problem by setting $c_j := p'_j$ and $Q_j := w'_j$. We have only one resource r with time window $[a_r, b_r] := [0, c']$. All edges have unlimited capacity d, so the task is now to pack all jobs into a limited time horizon c' such that the overall profit is maximized.

4.3 A branch-and-price solution approach

The formulation (EDGE) can be Dantzig-Wolfe decomposed [54], such that the resulting pricing problem assigns a given job to a given resource. The latter is denoted a subschedule. Each subschedule contains information on which job is assigned to which resource, when and where-from data is sent, and when job execution begins.

The master problem (MASTER) computes an optimal solution by merging subschedules. The model contains the decision variables y_p, $p \in P$, where P denotes the set of subschedules:

$$\max \quad \sum_{j \in J} c_j \sum_{p \in P} \delta_p^j y_p \tag{4.10}$$

$$\text{s.t.} \quad \sum_{p \in P} \delta_p^j y_p \leq 1 \qquad \forall j \in J \tag{4.11}$$

$$\sum_{p \in P} \delta_p^{ti^+} y_p \leq d_{i+}^t \qquad \forall i \in R, \forall t \in [a_i, b_i] \tag{4.12}$$

$$\sum_{p \in P} \delta_p^{ti^-} y_p \leq d_{i-}^t \qquad \forall i \in R, \forall t \in [a_i, b_i] \tag{4.13}$$

$$\sum_{p \in P} \delta_p^{tir} y_p \leq d_{ir}^t \qquad \forall r, i \in R, \forall t \in [a_{ir}, b_{ir}] \tag{4.14}$$

$$\sum_{\substack{j' \in J \\ j' \neq j}} \sum_{t'=t}^{\substack{\min\{t+Q_j, \\ b_{j'r}-Q'_j\}}} \sum_{p \in P} \delta_p^{j'rt'} y_p + Q_j \sum_{p \in P} \delta_p^{jrt} y_p \leq Q_j \qquad \begin{array}{l} \forall j \in J,\ r \in R, \\ t \in [a_{jr}, b_{jr} - Q_j] \end{array} \tag{4.15}$$

$$y_p \in \{0, 1\} \qquad \forall p \in P \tag{4.16}$$

Each subschedule $p \in P$ has a number of constants attached: $\delta_p^j = 1$ if subschedule p assigns job j, otherwise $\delta_p^j = 0$. Similarly, $\delta_p^{jt} = 1$ if subschedule p executes job j at time t, otherwise $\delta_p^{jt} = 0$. The constants $\delta_p^{ti^-} \geq 0$, $\delta_p^{ti^+} \geq 0$ and $\delta_p^{tir} \geq 0$ denote the amount of data going in and out of resource i, and between resources i and r at time t for subschedule p, respectively.

The objective function (4.10) maximizes the profit of executed jobs. The first constraint (4.11) ensures that each job is executed at most once. Constraints (4.12), (4.13) and

(4.14) ensure that bandwidth limitations are obeyed. Finally, constraint (4.15) says that a resource can execute at most one job at a time.

4.3.1 Pricing problem

The pricing problem decides which subschedules to add to the master problem. Recall, that a subschedule consists of assigning job j to a resource r. Let $\pi_j \geq 0$ be the dual of constraint (4.11), $\omega_{it} \geq 0$ be the dual of constraint (4.12), $\tau_{rt} \geq 0$ be the dual of constraint (4.13), $\rho_{tir} \geq 0$ be the dual of constraint (4.14), and $\lambda_{jrt} \geq 0$ be the dual of constraint (4.15). The reduced cost for pair (j, r) is rewritten into:

$$c_j - \pi_j \geq \sum_{t=a_{rj}}^{b_{rj}} (\tau_{rt} + Q_j \lambda_{jrt} + \sum_{\substack{j' \in J \\ j' \neq j}} \sum_{t'=\max\limits_{\{t-Q_{j'}+1, a_{j'r}\}}}^{\min\{t, b_{rj'}-Q_{j'}\}} \lambda_{j'rt'}) + \sum_{i \in R} \sum_{t=a_i}^{b_i} (\omega_{it} + \rho_{tir}) \quad (4.17)$$

The right hand side can be viewed as edge and execution costs when assigning job j to resource r.

To handle the time aspect in the pricing problem we transform the graph into a *time expanded* graph (see Figure 4.1), as done for the single commodity flow problem over time by [72, 74]. In the time expanded graph, sources and the target are connected through a set of added *time nodes*, where each time node represents a time stamp. Bandwidth limitations are represented as edge capacities. An edge going from source i to time node t has capacity $d_{it} = \min\{d^t_{ir}, d^t_{i^+}\}$, and an edge going from time node t to target r has capacity $d_{tr} = d^t_{r^-}$. The set of the edges (i, t) and (t, r) in the time expanded graph is denoted $\mathcal{E}$. The edge going out of resource i at time t is denoted (i, t) and has reduced cost $\bar{c}_{it} = (\omega_{it} + \rho_{tir})$ per flow unit. The edge going into r at time t is denoted (t, r) and has reduced cost $\bar{c}_{tr} = \tau_{rt}$ per flow unit. The sum

$$\bar{c}_t = Q_j \lambda_{jrt} + \sum_{j' \in J \setminus \{j\}} \sum_{t'=\max\{t-Q_{j'}+1, a_{j'r}\}}^{\min\{t, b_{rj'}-Q_{j'}\}} \lambda_{j'rt'}$$

is the reduced cost of executing job j at resource r at time t. The non-negative variable $f_{uv},\ (u, v) \in \mathcal{E}$ holds the amount of flow from node u to v in the time expanded graph. Using the variable x_t to indicate that job j starts at time t at resource r, the pricing problem is formulated as (PRICE)$_{jr}$:

$$\min \quad \sum_{it \in \mathcal{E}} \bar{c}_{it} f_{it} + \sum_{tr \in \mathcal{E}} \bar{c}_{tr} f_{tr} + \sum_{t \in [a_{rj}, b_{rj} - Q_j]} \bar{c}_t x_t \quad (4.18)$$

$$\text{s.t.} \quad \sum_{it \in \mathcal{E}} f_{it} = \sum_{tr \in \mathcal{E}} f_{tr} \qquad \forall t \in [a_{rj}, b_{rj} - Q_j[\quad (4.19)$$

$$\sum_{t' \in [a_{rj}, t[} f_{t'r} \geq S_j x_t \qquad \forall t \in [a_{rj}, b_{rj} - Q_j] \quad (4.20)$$

$$f_{uv} \leq d_{uv} \qquad \forall (u,v) \in \mathcal{E} \tag{4.21}$$

$$\sum_{t \in [a_{irj}, b_{irj}]} f_{it} = p_j^i \qquad \forall i \in R \backslash \{r\} \tag{4.22}$$

$$\sum_{t \in [a_{rj}, b_{rj}]} x_t = 1 \tag{4.23}$$

$$x_t \in \{0,1\} \qquad \forall t \in [a_{rj}, b_{rj} - Q_j]$$

$$f_{uv} \geq 0 \qquad (u,v) \in \mathcal{E}$$

The objective (4.18) is to minimize the reduced cost. The first constraint ensures flow conservation. Constraint (4.20) says that all data must arrive before the job is executed. Then in (4.21), edge capacities are obeyed and in (4.22) each resource transmits the demanded data. Constraint (4.23) ensures that the job j is executed. Finally, the bounds force variables to take on feasible values.

To overcome the problem of having to both send data and to find the optimal execution time, the pricing problem is instead considered for each feasible execution time, $t \in [a_{rj}, b_{rj} - Q_j]$. The later the execution time, the lower the data transmission costs are in (4.18), because a late execution time increases the number of ways data can be sent. Hence, we start with the latest execution time when solving the pricing problem and then decrease the time until a solution with positive reduced cost is found.

Also, the pricing problem is only solved for jobs with $c_j - \pi_j > 0$, because otherwise (4.17) will never be satisfied and a column with positive reduced cost cannot be found due to the dual variables taking on non-negative values.

The data transmission problem in the time expanded graph can be transformed into the polynomially solvable Linear Multicommodity Flow Problem (MFP). The amount of data to be transmitted corresponds to the amount of flow to be routed. Several data files are to be transmitted, i.e., in the MFP representation several commodities of flow must be routed. Now, solving the pricing problem on the time expanded graph corresponds to solving the MFP.

The mathematical formulation of the pricing problem viewed as an MFP is:

$$\begin{aligned}
\min \quad & \sum_{it \in \mathcal{E}} \bar{c}_{it} f_{it} + \sum_{tr \in \mathcal{E}} \bar{c}_{tr} f_{tr} & \\
\text{s.t.} \quad & \sum_{it \in \mathcal{E}} f_{it} = \sum_{tr \in \mathcal{E}} f_{tr} & \forall t \in [a_{rj}, b_{rj} - Q_j[\\
& f_{uv} \leq d_{uv} & \forall (u,v) \in \mathcal{E} \\
& \sum_{t \in [a_{irj}, b_{irj} - Q_j]} f_{it} = p_j^i & \forall i \in \mathcal{R} \\
& f_{uv} \geq 0 & (u,v) \in \mathcal{E}
\end{aligned}$$

Since the problem is a linear program, it can be solved in polynomial time. However,

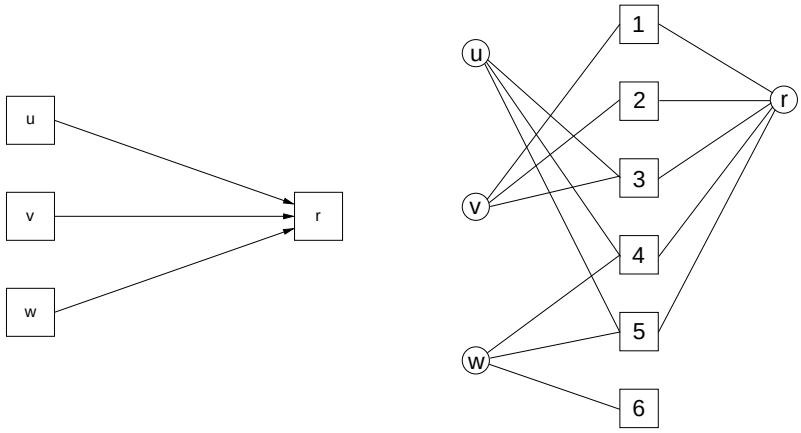


Figure 4.1: An example of how a network is transformed into a time expanded graph. The graph on the left hand side represents an instance consisting of three source resources: u with time window $[3, 5]$, v with time window $[1, 3]$ and w with time window $[4, 6]$. The target node, r, has time window $[1, 5]$. The figure on the right hand side shows the time expanded graph. Nodes representing times $1, \ldots, 6$ are introduced, and u, v, and w are connected to r via nodes, representing time slots where both parts are available.

larger instances of MFP can be difficult to solve. Proposed solution methods in the literature include Lagrangian methods, partition methods, decomposition techniques, dual ascent algorithms, bundle methods, interior point methods, etc., see [18, 119] for surveys of the problem and [134] for a review of solution techniques. Small instances are typically solved using the Simplex algorithm. To the best of our knowledge no straight-forward combinatorial algorithm exists for the MFP [52], therefore we choose to solve the pricing problem heuristically whenever possible.

The heuristic for the pricing problem is a greedy algorithm. Given a job j, an executing resource r and an execution time t, the heuristic works as follows. For each resource containing job data ($i \in R : i \neq r, p_j^i > 0$) the job data is sent to the executing resource via edges with lowest reduced cost. Edges are chosen in a greedy manner for each data source, i.e., the overall cost of sending all data may not be optimal.

When the heuristic cannot find a solution with positive reduced cost, the pricing problem is solved to optimality using a standard LP solver.

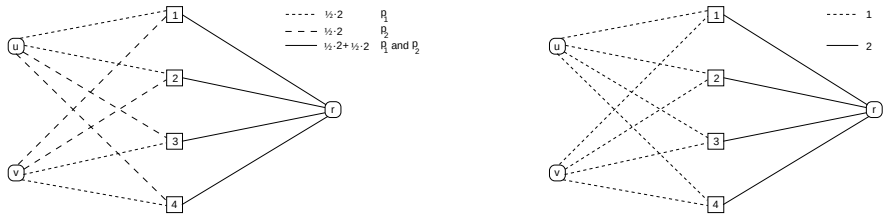


Figure 4.2: Illustration of fractional and integer solutions. The instance is shown in the time expanded graph representation: A job j is given with time window $[1, 10]$, execution time 5, and job data: 4 units at resource u, 4 units at resource v. Two resources u and v are available in time window $[1, 4]$ having bandwidth limitation 2 units of outgoing data per time slot, for all time slots in the time window. A resource r is available in time window $[1, 10]$ having bandwidth limitation 2 units of incoming data per time slot, for all time slots in the time window. The graph on the left hand side shows a fractional solution and the graph on the right hand side shows an integer solution.

4.3.2 Reaching feasible solutions

In the branch-and-price algorithm, a linear relaxation of the master problem is solved in order to find an upper bound in each search tree node. It may be necessary to branch to find an integer solution, thus a branching strategy must be implemented. In the following, we present a method to reach feasibility in certain types of fractional solutions. Three branching strategies are also presented, which combined ensure that integrality is eventually achieved.

4.3.2.1 Reaching feasibility from certain fractional solutions

We may have a fractional solution where a job is executed via several subschedules and where these subschedules have the same executing resource and same execution time. The fractional subschedules only differ in the way job data is transmitted. To reach a feasible solution, the fractional subschedules are replaced by a new subschedule with same executing resource and same execution time, but with a combination of the ways job data is transmitted. This is illustrated in the following example depicted in Figure 4.2: a fractional solution consists of two paths, p_1 and p_2, each used $1/2$ times, i.e., $y_{p_1} = y_{p_2} = 1/2$. Using the time expanded graph representation, the fractional solution is illustrated on the left hand side of Figure 4.2. That is, path p_1 sends data from u to r in sizes 2 at time 1 and 2, and from v to r in sizes 2 at time 3 and 4. Path p_2 sends data from u to r in sizes 2 at time 3 and 4, and from v to r in sizes 2 at time 1 and 2.

To avoid branching on a solution, which actually is feasible despite the fractional variables, p_1 and p_2 are replaced with a new subschedule. Again, using the time expanded graph representation, the new solution is illustrated on the right hand side of Figure 4.2. The new subschedule sends data from u to r in sizes 1 at time 1, 2, 3 and 4, and it sends data from v to r in sizes 1 at time 1, 2, 3 and 4. In this way, a feasible solution is reached without branching.

4.3.2.2 Branching strategies

The branching strategies consist of three branching constraints:

Each job can be executed at most once. In a fractional solution some jobs may be partially executed. For this reason, the following branching strategy is introduced:

$$\sum_{p \in P} \delta_p^j y_p = 0 \quad \text{vs.} \quad \sum_{p \in P} \delta_p^j y_p = 1 \quad j \in J \tag{4.24}$$

That is, job j is either executed or not. Next, in a fractional solution a job may be executed at several resources. This leads to the branching strategy:

$$\sum_{p \in P} \delta_p^{rj} y_p = 0 \quad \text{vs.} \quad \sum_{p \in P} \delta_p^{rj} y_p = 1 \quad j \in J,\, r \in R \tag{4.25}$$

That is, for job j and for resource r the job is either executed at the resource or not. Finally, a fractional solution may have two fractional subschedules using the same paths for sending data but having different execution times. For this reason, the following strategy is imposed:

$$\sum_{t' \in [a_j, t]} \sum_{p \in P} \delta_p^{jt'} y_p = 0 \quad \text{vs.} \quad \sum_{t' \in [a_j, t]} \sum_{p \in P} \delta_p^{jt'} y_p = 1 \quad j \in J,\, t \in [a_j, b_j - Q_j] \tag{4.26}$$

That is, for job j and time stamp t, the job is either executed before the time stamp or not.

The branching constraints are applied in the presented order, i.e., first branching candidates of type (4.24) are generated, which is followed by candidates of type (4.25), and finally of type (4.26).

The branching strategies result in constraints being added to the master problem and thus extra dual variables must be taken into account in the pricing problem. For constraint (4.24) the dual variables are ψ_j which can be added on all execution time for job j on all resources. For constraint (4.25) the dual variables are ψ_{jr}, which can be added on all execution times for job j on executing resource r. For cut (4.26) the dual variables are ψ_{jt}, which can be added to the reduced cost for executing job j at time t for all executing resources.

4.4 Stabilized column generation and improvements

In order to improve the basic branch-and-price algorithm we propose a number of refinements. First, Section 4.4.1 shows how the size of the model may be reduced by only adding constraints when they are violated, making the algorithm a branch-and-cut-and-price approach. Then in Section 4.4.2 we show how interior-point stabilization may limit the fluctuation in the dual variables, making the column generation converge faster. Finally, Sections 4.4.3 and 4.4.4 describe how we find an initial solution and how the size of the problem instance may be decreased by preprocessing.

4.4.1 Reducing the size of the master problem

The number of constraints in the master problem (4.10)-(4.15) may be very large, especially as the number of time intervals grow. Having a very large master problem increases the memory and time consumption of the branch-and-price algorithm, hence it may be beneficial to reduce the size by only including violated constraints.

The first constraint (4.11) ensures that each job is selected at most once. Thus the number of these constraints will always be relatively low and the constraints are always included in the master problem. The number of remaining constraints (4.12)-(4.15) depends on the time intervals and may thus be large. We only include these constraints when violated. To decide when constraints are violated, we develop a set of separation routines. Each routine consists of calculating network and resource consumption in the current solution by investigating each column and by maintaining network and resource matrices. When a constraint is violated, it is added as a cut to the current master problem.

The dual variables of constraints (4.12)-(4.15) are only to be included in the reduced costs, when the corresponding cuts are added to the master problem. This is handled in the pricing problem by investigating all added cuts when calculating the reduced costs.

Compared to the branch-and-price algorithm, less time is spent on maintaining the master problem in the branch-and-cut-and-price algorithm. But the cut separation routines in each iteration and the cut investigation when calculating the reduced costs may be time consuming. Thus the trade-off between the two solution methods lies in how much time is spent on handling the cuts versus having a large master problem.

4.4.2 Stabilized column generation

In a branch-and-price scheme, the dual variables of the master problem are used for deciding which columns to price in. The dual variables, however, may not always converge evenly towards their optimal values [115]. Many LP-solvers, for instance, return an extreme point in the dual solution space, which may lead to fluctuation in the dual variables. Also, degenerated problems can have many optimal solutions and thus many different optimal points in the dual solution space. "Unstable" dual variables may increase the number of iterations needed to converge [136]. The goal of stabilized column generation is thus to make the dual variables converge more evenly in order to save both memory and time consumption.

Stabilization methods generally consist of setting bounds on how much the values of the dual variables may change between two iterations in the pricing process. This can be done by setting bounding boxes for each dual variable [170] or by linearly punishing the distance between the former value and the current value of each dual variable [61]. Rousseau e.a. [170] suggest an interior-point stabilization method where a set of extreme points in the dual solution space is found, and where the dual variables are defined as a linear combination of the extreme points.

Reconsider the master problem (4.10)-(4.16). Let P^* be the set of variables $y_P > 0$ and let S^* be the set of constraints (4.11)–(4.15), which are not tight. Due to the Complementary Slackness Condition (CSC) the dual constraints corresponding to the variables in P^* must be tight and the dual variables corresponding to the constraints in S^* must be set to zero. Recall the dual variables. Taking P^* and S^* into account gives the dual problem:

$$\min \quad \sum_{j\in J}\sum_{i\in R}\left(\sum_{t=a_i}^{b_i}(\omega_{it}+\tau_{it})+\sum_{r\in R}\sum_{t=a_{ir}}^{b_{ir}}\rho_{tir}+\sum_{j\in J}\sum_{t=a_{ji}}^{b_{ji}-Q_j}\lambda_{jit}\right)+\pi_j$$

$$\text{s.t.} \quad c_j-\pi_j-\sum_{t=a_{rj}}^{b_{rj}}\left(\tau_{rt}+Q_j\lambda_{jrt}+\sum_{\substack{j'\in J\\ j\neq j'}}\sum_{t'=a'}^{b'}\lambda_{j'rt'}\right)-\sum_{i\in R}\sum_{t=a_i}^{b_i}(\omega_{it}+\rho_{tir})>0 \qquad \forall j\in J,\forall r\in R,\forall p\in P\backslash P^*$$

$$c_j-\pi_j-\sum_{t=a_{rj}}^{b_{rj}}\left(\tau_{rt}+Q_j\lambda_{jrt}+\sum_{\substack{j'\in J\\ j\neq j'}}\sum_{t'=a'}^{b'}\lambda_{j'rt'}\right)-\sum_{i\in R}\sum_{t=a_i}^{b_i}(\omega_{it}+\rho_{tir})=0 \qquad \forall j\in J,\forall r\in R,\forall p\in P^*$$

$$\pi_i,\tau_i,\lambda_i,\rho_i=0 \qquad \forall i\in S^*$$

$$\pi_i,\tau_i,\lambda_i,\rho_i>0 \qquad \forall i\in S\backslash S^*$$

Where $a'=\max\{t-Q_{j'}+1,a_{j'r}\}$, $b'=\min\{t,b_{rj'}-Q_{j'}\}$ and S is the set of all

constraints in the primal problem.

The interior-point method wishes to find several extreme points and then defines the dual variables as a linear combination of these points. To find k different extreme points we multiply the dual objective function with a random vector $0 \leq u_i \leq 1$ for each of the $i \in \{1, 2, \ldots, k\}$ times the stabilized dual problem is solved. In practice it can be very time and space consuming to set up the stabilized dual problem k times, hence we instead solve the dual of the stabilized dual problem. The dual of the stabilized dual problem is:

$$
\begin{aligned}
\max \quad & \sum_{j \in J} c_j \sum_{p \in P} \delta_p^j y_p & & \\
\text{s.t.} \quad & \sum_{p \in P} \delta_p^j y_p \leq u & & \forall j \in J,\ \forall s \in S \backslash S^* \\
& \sum_{p \in P} \delta_p^{ti^+} y_p \leq u\, d_{i^+}^t & & \forall i \in R,\ \forall t \in [a_i, b_i],\ \forall s \in S \backslash S^* \\
& \sum_{p \in P} \delta_p^{ti^-} y_p \leq u\, d_{i^-}^t & & \forall i \in R,\ \forall t \in [a_i, b_i],\ \forall s \in S \backslash S^* \\
& \sum_{p \in P} \delta_p^{tir} y_p \leq u\, d_{ir}^t & & \forall r, i \in R, \forall t \in [a_{ir}, b_{ir}], \forall s \in S \backslash S^* \\
\sum_{\substack{j' \in J \\ j' \neq j}} \sum_{t'=t}^{\substack{\min\{t+Q_j,\\ b_{j'r}-Q'_j\}}} & \sum_{p \in P} \delta_p^{j'rt'} y_p + Q_j \sum_{p \in P} \delta_p^{jrt} y_p \leq Q_j & & \forall j \in J,\ r \in R, \\
& & & t \in [a_{jr}, b_{jr} - Q_j], \forall s \in S \backslash S^* \\
& y_p \in \{0, 1\} & & \forall p \in P \backslash P^* \\
& y_p = 0 & & \forall p \in P^*
\end{aligned}
$$

This model is reached by modifying the original primal problem slightly. A solution is found by letting the LP-solver resolve the stabilized problem. The trade-off in interior-point stabilization lies in the amount of time spent on finding the stabilized dual variables and the amount of time saved by possibly decreasing the number of iterations in the column generation.

4.4.3 Starting solution

A start solution to the problem instance must be found in order to get values for dual variables. The scheduling problem is solved for each job until a feasible start solution is found. Three greedy heuristics are implemented for solving the scheduling problem.

Assign home: If a job has all data placed on exactly one resource, the job is assigned to that resource if possible. *First come, first serve:* For each resource, the algorithm assigns the first job which can be executed on that resource. *Best first:* Each job is

assigned to the resource on which the job execution finishes first. The data transmission problem is solved heuristically by taking one source at a time and then using the required time and bandwidth to transmit all job data to the target.

The heuristics do not guarantee a feasible solution even if a problem instance is solvable. In the case where a heuristic solution cannot be found, the starting problem is solved using a modified version of the exact algorithm from the pricing problem. Reduced costs are replaced with negated real costs for executing a job (the algorithm seeks to minimize, hence the negation). The modified algorithm is denoted the *exact start solution method*.

We have through preliminary testing concluded that the starting algorithm should work as follows. The heuristics are run in the order *assign home, first come first serve, best first* until a solution is found. If the heuristics are unable to reach a feasible solution, the *exact start solution method* is run. If this does not result in a start solution, then the problem instance is unsolvable.

4.4.4 Preprocessing

Preprocessing can be used to a-priori limit the solution space. We use the following preprocessing rules:

Problem instance size: When a job is not available, none of the resources need to be considered. Hence, a system start time, $A = \min_{j \in J} a_j$, and a system end time, $B = \max_{j \in J} b_j$, are found and resource time windows are set to $[\max\{a_r, A\}, \min\{b_r, B\}]$, $\forall r \in R$. *Job availability:* If the job execution time is greater than the time space in which the job is available, then the job cannot be executed. *Job data source availability:* If a resource containing job data is not available in the same time space as the job, then the job cannot be executed. *Job data source bandwidth:* If a resource containing job data does not have enough available bandwidth to send out all data in time for execution (before $b_j - Q_j$) and if the resource cannot execute the job itself, then the job cannot be executed. *Transmission of job data:* For each resource and for each job, it is investigated whether or not all job data can be sent to the resource in time for execution. If not, then the job cannot be executed on that resource.

The five preprocessing steps can be checked in polynomial time. With regard to the mathematical formulation of the problem we predict that especially the first step, where resource time windows are narrowed, can have great effect: the number of edges in the formulation is lowered, thus a subset of the capacity constraints can be left out. With regard to finding combinations of which jobs to execute on which resources, then the remaining four steps of preprocessing have a good effect.

4.5 Computational results

The proposed solution methods have been computationally evaluated as follows: first, problem instances are generated and details regarding the data generation are presented in the following. Next, preprocessing is performed to limit the solution space. Finally, the computational results are reported and discussed. The edge formulation (EDGE) was solved by CPLEX.

4.5.1 Data generation

Test instances are generated to reflect activity in a grid during 24 hours. The instances are randomly generated, but the number of jobs, resources and the size of time intervals reflect actual scheduling problems. The resources are generated such that their start and end times lie within the 24 hour time span. Furthermore, the end time is set to be at least one time slot later than the start time and bandwidth limitations are set randomly between 0 and 10 Gb/s. Jobs are generated such that job data is distributed on up to all resources and such that each job data source holds at most five units of job data. The job start time lies within the 24 hour time span, and the end time is set to be twice the estimated computations time of the job later than the start time.

We consider instances, where the number of jobs and resources is set to 10, 20, 50, 100, 200, 500 and 1000. The exponential-like growth of jobs and resources will hopefully reveal any connection between problem instance size and the complexity of the scheduling problem. Two types of time granularity are used: 15 and 30 minutes. Smaller granularity is not considered since jobs taking less than 15 minutes might as well be computed on the user's home computer. The test instances with 15 minute time intervals are scaled into corresponding instances with 30 minute time intervals in order to show any connection between the time granularity and the complexity of the scheduling problem.

The scaling algorithm divides all time units with 2 in order to go from a 15 to a 30 minute time interval size. After scaling all start times are floored and all end times are ceiled such that all time windows are of at least size 1. Obviously, this imposes inaccuracy in the instances and an optimal solution value to the scaled problem may be greater than that of the original problem. For this reason, an optimal solution to a scaled problem may be infeasible for the original problem.

After scaling, the estimated computing time for a job is ceiled such that each job takes at least one time period to run. The estimated computing time may become larger, hence the solution for the scaled problem may compute fewer jobs than the solution of the original problem. By computationally evaluating instances with both time interval

sizes, we hope to show how scaling time intervals affects the solution quality.

4.5.2 Results

The exact solution methods have been implemented and tested in C++ using the branch-and-cut-and-price framework COIN [140]. Results are compared to test runs using CPLEX for solving the (EDGE) formulation from Section 4.2. The methods are tested on a 2.66 GHz Intel Xeon machine with 8 Gb RAM. Note, that CPU times in the following stem from using one core. ILOG CPLEX 10.2 is used as LP-solver.

Through preliminary results, we have decided to set $|J|/2$ as upper bound on the number of columns computed in each iteration. Furthermore, all branching candidates are found when branching occurs and a best-first search strategy is used in the branch-and-bound tree. Computations regarding selection of branching candidate and branching child are handled by COIN.

First, Table 4.1 shows results from solving the (EDGE) model with CPLEX. The instances have up to 100 jobs, up to 1000 resources, and time granularity of 15 minutes. We have chosen to only generate edge-based models for instances with up to 100 jobs, since the generated models become very large and take up several gigabytes of space. This also explains why CPLEX runs out of memory in the (EDGE) model. Time usage explodes as the number of resources increases.

Next, the three exact algorithms are analyzed. Test data results for instances with 100, 200, 500 and 1000 jobs and up to 1000 resources are seen in Table 4.2 for a 15 minute time interval and Table 4.3 for a 30 minute time interval. The algorithm B&P (branch-and-price) denotes the simple branch-and-price algorithm described in Section 4.3. Algorithm B&C&P (branch-and-cut-and-price) solves the same model but constraints are only added when they are violated as described in Section 4.4.1. Finally, algorithm B&C&P+S (branch-and-cut-and-price with stabilized column generation) also includes stabilization as described in Section 4.4. The time consumption for all test runs is bounded by 1800 seconds.

The scaled instances generally consist of fewer constraints and columns, and the number of iterations is generally lower than for the original problem, leading to faster solution times. Furthermore, the solution values of some of the original instances and of the scaled instances differ. The solution values of the scaled instance are generally lower than or equal to those of the original instances, but some are also greater. As discussed in Section 4.5.1 solutions for scaled instances with values greater than those for the corresponding original problem, are infeasible. When not counting instances with memory or time problems, solving the scaled instances results in approximately 18% infeasible solutions, approximately 10% optimal solutions to the original problem, and

approximately 71% solutions worse than the optimal solutions for the corresponding original problems.

The pure branch-and-price algorithm does not perform very well: it runs out of memory and time even for smaller instances. The algorithm, however, is capable of solving larger instances than CPLEX. Reasons for the performance difficulties lie in the size of the master problem, where the number of rows explodes. The branch-and-price algorithm also generally suffers from a large number of column generation iterations.

The branch-and-cut-and-price algorithm without the stabilized column generation has much better performance than the branch-and-price algorithm. While it also sometimes runs into memory and time problems, it is capable of solving the majority of the test instances. The number of rows in the master problem is reduced significantly and the number of columns is also decreased. As was the case for the previous solution approach, the branch-and-cut-and-price algorithm suffers from a large number of column generation iterations, which takes up much time.

Finally, we consider the branch-and-cut-and-price algorithm with stabilized column generation. This method has by far the best performance and solves all instances within minutes. The size of the master problem is dramatically reduced, especially when comparing to the branch-and-price algorithm. Furthermore, the stabilized column generation decreases the number of column generation iterations significantly, which indicates that the stabilization has a very beneficial impact on the values of the dual variables.

Jobs	Resources	Objective	Time
10	10	8	0.00
10	20	6	0.01
10	50	8	0.04
10	100	10	0.35
10	200	10	1.03
10	500	8	6.88
10	1000	10	48.41
20	10	20	0.01
20	20	16	0.01
20	50	18	0.05
20	100	24	0.57
20	200	24	3.07
20	500	22	26.93
20	1000	30	173.28
50	10	68	0.04
50	20	52	0.16
50	50	66	0.69
50	100	46	2.96
50	200	34	8.93
50	500	66	76.20
50	1000	-	out of memory
100	10	136	0.09
100	20	104	0.24
100	50	94	1.07
100	100	106	5.85
100	200	98	18.16
100	500	110	153.99
100	1000	-	out of memory

Table 4.1: Test run results for solving the edge-based model using CPLEX. First column is the number jobs (`Jobs`), second column is the number of resources (`Resources`), third column is the objective value (`Objective`) and finally time consumption in seconds is given (`Time`). The running time includes generation and reading of the MIP file.

		B&P				B&C&P				B&C&P+S				
Jobs	**Res.**	**Rows**	**Cols**	**Iter.**	**Time**	**Rows**	**Cols**	**Iter.**	**Time**	**Rows**	**Cols**	**Iter.**	**Time**	**z**
100	10	61617	628	90	94.306	100	386	19	0.240	100	99	3	0.108	136
100	20	103020	791	44	79.597	100	478	13	0.300	100	87	2	0.100	104
100	50	224860	1292	28	69.728	100	820	17	1.220	100	95	2	0.164	94
100	100	447665	1953	44	236.547	100	1452	31	6.588	100	145	4	0.456	106
100	200	899992	2018	42	294.906	100	1918	40	39.630	100	49	1	0.304	98
100	500	2339064	2357	51	space	100	2305	48	328.869	100	55	3	6.032	110
100	1000	4659483	1449	29	*1818.034	100	2200	45	962.300	100	49	1	2.328	98
200	10	82040	631	455	1020.732	200	397	14	0.312	200	101	3	0.272	184
200	20	197758	1333	46	277.457	200	777	16	0.680	200	111	3	0.304	184
200	50	492045	3270	92	*1800.133	200	2007	26	4.208	200	174	3	0.456	208
200	100	970942	5000	88	*1832.179	200	3198	35	17.253	200	197	4	0.964	198
200	200	1931695	8089	85	*2307.456	200	6057	64	155.010	200	119	2	1.180	238
200	500	-	-	-	space	200	6992	71	space	200	92	2	3.768	184
200	1000	-	-	-	space	200	2206	22	*1808.061	200	107	2	12.057	214
500	10	209842	1382	49	*1986.252	500	833	23	1.780	500	242	8	1.688	404
500	20	432726	3161	39	*2084.098	500	1841	22	2.672	500	291	6	1.708	472
500	50	1250982	8883	54	*1913.804	500	5211	32	13.749	500	384	5	2.352	522
500	100	2551162	10261	41	*2937.144	500	10494	49	71.880	500	264	4	3.032	528
500	200	4910410	10243	41	*5617.423	500	17128	73	space	500	245	3	4.596	490
500	500	-	-	-	space	500	8508	34	*1820.942	500	260	3	14.353	520
500	1000	-	-	-	space	500	2252	9	*1921.916	500	254	3	42.279	508
1000	10	416830	3543	246	space	1001	2017	57	21.877	1001	540	10	7.928	892
1000	20	1057108	9205	97	space	1000	5286	68	27.598	1000	766	8	6.980	1170
1000	50	2424838	10510	21	*2503.764	1000	9722	45	39.110	1000	645	7	8.008	1042
1000	100	5064106	10513	21	*5082.330	1000	20819	59	170.547	1000	897	10	14.925	1054
1000	200	-	-	-	space	1000	40015	93	space	1000	671	11	35.814	1026
1000	500	-	-	-	space	1000	8501	17	*1866.329	1000	801	4	33.006	1008
1000	1000	-	-	-	space	1000	3008	6	*2198.537	1000	513	6	167.498	1026

Table 4.2: Test run results for solving the instances with time granularity of 15 minutes with the three algorithms. The number of jobs (`Jobs`) is reported, as well as the number of resources (`Res.`). For each of the three algorithms the table holds information on the number of rows (`Rows`) and columns (`Cols`) in the master problem, the number of iterations (`Iter.`) in column generation and the time consumption (`Time`) in seconds. The optimal objective value is reported in column `z`. An '`*`' indicates that the test ran out of time and `space` indicates that the test ran out of memory.

		B&P				B&C&P				B&C&P+S				
Jobs	**Res.**	**Rows**	**Cols**	**Iter.**	**Time**	**Rows**	**Cols**	**Iter.**	**Time**	**Rows**	**Cols**	**Iter.**	**Time**	**z**
100	10	31210	222	6	1.252	101	222	6	0.068	101	70	3	0.080	126
100	20	52528	295	6	2.144	105	295	6	0.116	105	65	3	0.116	98
100	50	115129	595	12	10.993	105	595	12	0.868	105	111	3	1.212	92
100	100	228401	1150	23	48.363	100	1150	23	5.492	100	100	2	0.384	102
100	200	459373	1001	21	122.180	100	1001	21	31.694	100	50	1	0.984	100
100	500	1189208	1258	27	621.255	100	1258	27	306.907	100	56	1	6.584	112
100	1000	2379079	1350	28	1713.423	100	1350	28	983.425	100	49	1	18.573	98
200	10	42216	318	9	4.300	278	329	9	0.188	249	185	6	0.364	176
200	20	100796	452	6	7.620	220	455	6	0.324	220	126	4	0.328	174
200	50	251046	1290	17	97.574	217	1290	15	2.912	217	301	4	0.644	204
200	100	495746	1797	18	167.234	209	1797	18	13.137	209	239	3	1.148	196
200	200	985909	3921	40	852.429	214	3921	40	141.969	202	308	4	6.624	240
200	500	2499760	3092	31	*1811.205	207	4392	44	941.383	207	92	2	30.226	182
200	1000	4970713	1309	13	*1930.293	200	2209	22	*1825.454	200	109	1	45.503	218
500	10	107427	521	23	70.900	1477	592	13	1.296	1644	459	16	3.544	368
500	20	222384	1130	5	33.158	813	1176	6	0.740	581	689	12	2.660	440
500	50	637036	3511	18	369.943	706	3511	18	10.661	699	895	8	3.908	506
500	100	1298170	7513	31	1409.156	575	7513	31	52.559	516	762	5	6.300	516
500	200	2502637	5245	21	*1800.117	561	9087	39	351.502	514	875	5	24.918	490
500	500	6236304	2011	8	*1846.055	530	8761	35	*1839.439	512	511	4	129.732	520
500	1000	-	-	-	space	555	2505	10	*1923.164	503	940	5	501.943	510
1000	10	214426	791	5	56.272	6185	1206	10	5.080	6344	883	12	29.882	644
1000	20	537070	2689	18	664.286	4088	3173	22	11.765	3912	2040	29	23.301	1006
1000	50	1234462	5723	22	1761.130	2256	5924	20	24.138	2257	1928	9	15.165	954
1000	100	2585158	6522	13	*1951.422	1521	13728	31	157.978	1495	2427	8	29.050	1014
1000	200	4943786	3513	7	*2041.944	1328	25070	57	space	1340	2999	8	72.389	1018
1000	500	-	-	-	space	1272	8006	16	*1845.023	1061	3506	8	421.270	1026
1000	1000	-	-	-	space	1072	3016	6	*2241.496	1002	1016	3	695.603	1030

Table 4.3: Test run results for solving the instances with time granularity of 30 minutes. The columns are explained in Table 4.2. An '*' indicates that the test ran out of time and `space` indicates that it ran out of memory.

4.6 Conclusion

This paper has formalized and formulated the integrated job scheduling and routing problem. Computational experiments showed that the simple formulation (EDGE) could not be used to solve large-sized instances to optimality, hence three exact algorithms were proposed: a branch-and-price algorithm, a branch-and-cut-and-price algorithm and a branch-and-cut-and-price algorithm with stabilized column generation. The methods are based on a new mathematical formulation of the integrated job scheduling and network routing problem. Furthermore, the exact solution methods include new branching strategies. The branch-and-price algorithm includes all constraints of the master problem, while the branch-and-cut-and-price algorithms only include violated constraints. Furthermore, the branch-and-cut-and-price algorithm is extended with stabilized column generation, which consists of calculating the dual variables from a number of extreme points in the solution space.

The solution methods have been computationally evaluated on instances with up to 1000 jobs and resources, 24 hour scheduling activity and a 15 minute time granularity. CPLEX can only solve instances with up to 100 jobs using model (EDGE). The branch-and-price algorithm is capable of solving more instances, but still shows somewhat poor performance due to large memory and time consumptions. The branch-and-cut-and-price algorithm without stabilization performs well, solving the majority of all instances. However, the algorithm still has some time problems due to many column generation iterations. Finally, adding stabilized column generation to the branch-and-cut-and-price algorithm improves performance dramatically. The number of column generation iterations is reduced significantly and the algorithm solves all instances within minutes. Furthermore, the algorithms have been tested on instances using a 15 and 30 minute time granularity, respectively. The computational evaluation showed that the time granularity affects time consumption and solution quality; the larger time granularity, the faster the instances are solved, but the solution quality decreases.

Overall, the branch-and-cut-and-price algorithm with stabilized column generation performs particularly well. Within a few minutes, the algorithm solves instances with 1000 jobs and resources covering 24 hours of scheduling activity with time granularity as small as 15 minutes. Hence, the algorithm can easily be used both as an actual scheduling algorithm for planned jobs or job queue emptying in grid computing, and as a tool for analyzing grid performance.

Acknowledgement
We would like to thank GlobalConnect A/S for their support of this work.

CHAPTER 5

A Survey of the Routing and Wavelength Assignment Problem

Mette Gamst
DTU Management Engineering, Technical University of Denmark

In an all-optical network, optical fibres are used to transmit data. An optical fibre carries light along its length at high rates and with little loss. Several wavelengths on a single fibre can be used to transfer data, when using wavelength-division multiplexing. In this way, several data transmissions at very high speed can take place on a single fibre.

When transmitting data in an all-optical network, data connections must be established in such a way that two or more connections never share a wavelength on the same fibre. The routing and wavelength assignment (RWA) problem consists of finding a path and a wavelength for a set of data connections. The objective is typically to maximize the profit of established data connections or to minimize the cost of establishing all data connections. The RWA is $\mathcal{NP}$-hard, thus much research has been conducted to finding a good way of approaching the problem.

Technical Report, ISBN 978-87-90855-56-7, DTU, 2009

This paper introduces the RWA and lists a number of restrictions from the literature on the RWA and on the underlying network topology. An overview of heuristic, metaheuristic and exact solution methods is given. Running times for the heuristic methods are presented and computational results from the literature are discussed.

5.1 Introduction

The use of optical fibres in telecommunication infrastructure is ever increasing. An optical fibre carries light along its length at very high rates and with little loss. When data is sent via an optical fibre, it is transmitted on a certain wavelength of light. A fibre can carry several independent transmissions, each by a different wavelength. The *wavelength-division multiplexing* (WDM) technology allows multiple optical carrier signals on a single optical fibre. WDM works on a circuit switched network, i.e., in a network where the connection between nodes and terminals is established before use and where the wavelength is not shared with other traffic. For a technical overview of optical fibres, see Halsall [98], and for more information on the WDM, see Thiele and Nebeling [187] or the thesis of Jue [114].

The problem of finding a good way of establishing data connections and of assigning wavelengths to the different connections, is denoted the *routing and wavelength assignment* (RWA) problem. Two or more data connections are not allowed to share the same wavelength on the same fibre. Constraints can be set on whether or not wavelengths can be converted. If wavelength conversion is possible, then further constraints can be set on where conversion may take place and on the range of wavelengths, into which a wavelength can be converted.

The RWA problem can be considered as a *static* problem, where all wavelengths of every future connection are established at all times. Another viewpoint is the *dynamic* version of RWA, where a wavelength is not reserved before it is needed and where the wavelength is released when the corresponding data connection is no longer needed. The objective is typically to maximize the number of established connections or to minimize the number of used wavelengths.

The RWA is $\mathcal{NP}$-hard, thus several solution approaches are presented in the literature. A common approach is to decompose the RWA into two subproblems: the routing problem and the wavelength assignment problem. The complexity of the routing problem depends on the chosen objective, while the wavelength assignment problem always is $\mathcal{NP}$-hard, see Zang et al. [206]. Another approach is to solve the RWA problem as one problem. Methods for this include metaheuristics and integer linear programming formulations. An overview of the proposed methods is presented in Table 5.1. The table shows what problem each method works on, the complexity of each method, and

finally gives references to the literature. Theoretical running times are only given for the constructive heuristics.

Some surveys on the RWA problem exist in the literature: Zang et al. [206] present a survey containing few routing approaches and many heuristics for the wavelength assignment problem. The latter are compared experimentally. Choi et al. [48] present a classification of existing methods for the RWA, where approaches are argued to be either search methods or selection methods. Furthermore, Choi et al. compare the performance of methods, but apart from a few theoretical running times, it is not clear what the comparisons are based on.

The contribution of this survey is the presentation of a much larger variety of solution methods than included in the surveys of Zang et al. and Choi et al. The presented methods include recently presented approaches from the literature. This paper not only considers the decomposed RWA, but also presents metaheuristics and exact formulations of the overall RWA. Furthermore, experiments from the literature is gathered and discussed. No general benchmark instances are used in the literature and the objective of solution methods differs. For these reasons, it is not trivial to decide which methods perform better, thus this survey also presents theoretical running times and uses these along with test results in a performance analysis of the proposed solution methods. Finally, we give recommendations on future work in the RWA research area.

This survey is structured as follows. First, in Section 5.2, the RWA problem and variants hereof are defined. The network topology is presented, i.e., constraints on whether or not wavelength conversion is allowed, etc. In Section 5.3, methods for solving the RWA problem heuristically are presented. These methods are all based on the decomposition of RWA into the two subproblems: the routing problem and the wavelength assignment problem. The section includes an overview of experimental results from the literature along with theoretical running times for the constructive heuristics. In Section 5.4 methods for the overall RWA is presented. These methods include metaheuristics and integer linear programming formulations. The section contains experimental results from the literature. Concluding remarks are given in Section 5.5. This section includes conclusions on the performance analysis of the presented solution methods and our recommendations on further work on the RWA.

Approach	Problem	Sta./Dyn.	Complexity	Ref.
FIXED ROUTING	Routing	Both	$\mathcal{O}(E + V \log V)$	[38, 48]
FIXED-ALTERNATE ROUTING	Routing	Both	$\mathcal{O}(E + V \log V + k)$	[22, 38, 66]
ADAPTIVE ROUTING	Routing	Dyn.	$\mathcal{O}(E + V \log V)$	[206]
LEAST CONGESTED PATH ROUTING	Routing	Both	$\mathcal{O}(E(E + V \log V))$	[43, 139]
SHORTEST PATH ADAPTIVE ROUTING	Routing	Both	$\mathcal{O}(E(E + V \log V))$	[139]
ROUTING WITH REDUCTION OF WAVELENGTH CONTINUITY CONFLICTS	Routing	Both	Polynomial	[125, 126]
ANT COLONY ROUTING	Routing	Sta.	Metaheuristic	[195]
GENETIC ALGORITHM	Routing	Sta.	Metaheuristic	[22]
INTEGER PROGRAMMING	Routing	Sta.	$\mathcal{NP}$-hard	[206]
GRAPH COLORING	WA	Both	$\mathcal{NP}$-hard	[206]
RANDOM ASSIGNMENT	WA	Both	$\mathcal{O}(WE)$	[183]
FIRST FIT ASSIGNMENT	WA	Both	$\mathcal{O}(WE)$	[130]
LEAST USED ASSIGNMENT	WA	Both	$\mathcal{O}(W \log W + WE)$	[150, 206]
MOST USED ASSIGNMENT	WA	Both	$\mathcal{O}(W \log W + WE)$	[150, 206]
EXHAUSTIVE SEARCH ASSIGNMENT	WA	Both	$\mathcal{O}(WE)$	[150]
MINIMUM PRODUCT ASSIGNMENT	WA	Both	$\mathcal{O}(WE)$	[110]
LEAST LOADED ASSIGNMENT	WA	Both	$\mathcal{O}(WE)$	[117, 206]
MAXIMUM SUM ASSIGNMENT	WA	Both	$\mathcal{O}(kWE)$	[28, 183]
RELATIVE CAPACITY LOSS ASSIGNMENT	WA	Both	$\mathcal{O}(kWE)$	[207]
DISTRIBUTED RELATIVE CAPACITY LOSS HEURISTIC	WA	Both	$\mathcal{O}(kWE)$	[206]
WAVELENGTH RESERVATION ASSIGNMENT	WA	Dyn.	$\mathcal{O}(1)$	[38]
PROTECTING THRESHOLD ASSIGNMENT	WA	Dyn.	$\mathcal{O}(1)$	[38]
GENETIC ALGORITHM	WA	Sta.	Metaheuristic	[101]
SIMULATED ANNEALING	WA	Sta.	Metaheuristic	[101]
TABU SEARCH	WA	Sta.	Metaheuristic	[101]
BIN PACKING HEURISTIC	WA	Sta.	Metaheuristic	[179]

ANT COLONY OPTIMIZATION	RWA	Sta.	Metaheuristic	[16]
GENETIC ALGORITHM	RWA	Sta.	Metaheuristic	[6, 178]
MIXED INTEGER PROGRAMMING	RWA	Sta.	$\mathcal{NP}$-hard	[164]
INTEGER PROGRAMMING	RWA	Sta.	$\mathcal{NP}$-hard	[206]
INTEGER PROGRAMMING	RWA	Sta.	$\mathcal{NP}$-hard	[156]
INTEGER PROGRAMMING	RWA	Sta.	$\mathcal{NP}$-hard	[106]
INTEGER MULTICOMMODITY FLOW PROBLEM	RWA	Sta.	$\mathcal{NP}$-hard	[31]
INTEGER PROGRAMMING	RWA	Sta.	$\mathcal{NP}$-hard	[138]
INTEGER PROGRAMMING	RWA	Sta.	$\mathcal{NP}$-hard	[107]

Table 5.1: An overview of all the methods, which are presented in this survey. The first column contains the name of the methods. Then follows problem types: the routing problem, the wavelength assignment problem (WA), or the RWA problem. The third column denotes whether or not, the method works on the static problem (Sta.) or the dynamic problem (Dyn.). The next column contains complexity: theoretic running times are only given for the heuristics. Finally, the right most column gives references to the literature for each method.

5.2 Problem definition

In this section, details on the RWA and on the all-optical network are presented. First, we discuss common assumptions on the network in which to establish data connections. Next, the two main variants of the RWA, the static and the dynamic RWA, are further introduced.

5.2.1 Network topology

The optical network is considered in an abstract manner. Technical details are omitted, instead we consider a *network* consisting of *nodes* and *edges*. Edges represent fibre links. An edge can hold several fibres, each potentially holding several wavelengths. Single-fibre is when each edge consists of only one fibre and multi-fibre is when each edge consists of several fibres. In this paper, we work on single-fibre networks unless else is mentioned. A node corresponds to any active equipment with an ingoing and/or outgoing edge. This could be a switch, a hub, an amplifier etc. A data connection request consists of a source and a target. A path with an assigned wavelength is to be found between the source and the target nodes. In the RWA, paths of different data connections are to be generated such that no two paths share the same edge and the same wavelength. That is, two paths using the same wavelength, must be edge disjoint. An example of the network representation is seen in Figure 5.1.

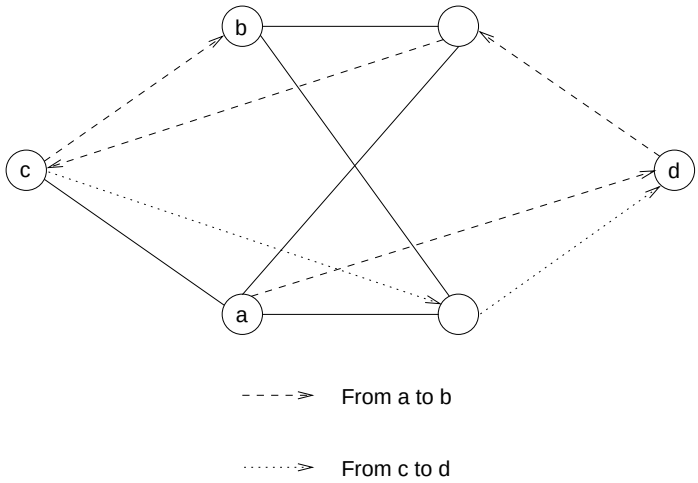


Figure 5.1: An example of a network representation of an optical network. Two data connections are routed through the network using the same wavelength. Thus, the two paths are edge disjoint.

When working on the RWA, some assumptions on *wavelength conversion* are made. A data connection may change wavelength when wavelength converters are available at intermediate nodes of the data connection path. In the literature, RWA works on different networks:

- There are no wavelength converters. In this case, a wavelength continuity constraint is imposed, see Zang et al. [206].
- Only a subset of nodes includes wavelength converters. This is denoted sparse wavelength conversion, see Iness and Mukherjee [104].
- All nodes include wavelength converters. The network is said to be wavelength convertible, see Ramamurthy and Mukherjee [163].

In the network representation, a switch with a wavelength converter attached is simply considered as one node. Comparisons of the different types of networks have been performed by Barry and Humblet [29], among others.

Furthermore, constraints on the usage of wavelength converters may be imposed. These constraints include sharing of converters and limiting the range of possible conversions. Sharing converters may be beneficial. If converters are not shared, then the number of converters at a node increases. Lee and Li [137] have shown that when the number of wavelength converters at a node exceeds some threshold, then the performance of the network decreases.

Some converters only support changes of wavelengths within a certain range. E.g., the wavelength λ_i can be converted to wavelengths in the range $\lambda_{(i-k)}, \ldots, \lambda_i, \ldots, \lambda_{(i+k)}$, where k is the range limitation factor. For more information on the limited-range wavelength converters, see the work of Iness and Mukherjee [104] or of Yates et al. [205].

When wavelength converters are only placed on certain nodes, much research has been conducted on network design, i.e., where to place the converters. Dutta and Rouskas [62] present a survey and a number of heuristics for the problem of designing the network. Koster and Zymolka [127, 128, 129] give lower bounds and then solve the problem of minimizing the number of required wavelength converters to optimality. A thorough analysis on the overall design of a WDM network is performed by Jue [114] and an analysis on how to place the components of an optical network is done by Iness [103].

5.2.2 Variants of the RWA

In the following, we consider both the static and the dynamic RWA. Recall that in the static RWA, all data connection requests are known in advance, they are to be established at the same time, and they are assumed to exist forever. An instance may hold more data connection requests than can be established; if a connection cannot be established, it is said to be *blocked*. Hence, the objective of the static RWA is typically to maximize either the number of established data connections or the profit of established data connections. The static RWA is proved to be $\mathcal{NP}$-hard by Chlamtac et al. [47]. The problem may be formulated mathematically as a mixed integer problem, see Ranaswami and Sivarajan [164].

In the dynamic RWA, data connection requests arrive with time; they are to be established at arrival time and they are to be shut down at a given time. This means that wavelengths can be reused; when a data connection is shut down, its wavelength is released. As for the static case, blocking may occur. The objective of the dynamic RWA is typically to maximize the number of established data connections. Because no knowledge exists on future data connection requests, solutions to the dynamic RWA are local optimums.

The far majority of methods for solving the RWA apply to both the static and the dynamic RWA. In the following sections, solution methods from the literature are presented.

5.3 Decomposition of the RWA

Both the dynamic and the static RWA are difficult to solve. A reason for this is that the problems consist of two parts: routing data connections and assigning wavelength to data connections.

Both the static and the dynamic RWA are often solved by splitting the problem into two subproblems: the routing problem and the wavelength assignment problem, see e.g. Arteta et al. [16], Zang et al. [206] and Zheng and Mouftah [208]. First, routes for all connections are found. Next, wavelengths are assigned. The division of the problem makes it easier to solve, but solving the subproblems instead of the whole problem does not guarantee an optimal solution. Instead, dividing the RWA into two parts is a heuristic method.

Much research has been put into decomposing the RWA into these two parts. In this

section, we present some of the routing algorithms and methods for wavelengths assignment from the literature.

5.3.1 Routing

The routing problem consists of finding a path between the source and the target of each data connection. The complexity of the routing problem depends on the objective. If we simply wish to connect a set of node pairs, then the problem can be solved polynomially using a shortest path algorithm. If the objective is to minimize the maximal number of paths on an edge, then the problem is $\mathcal{NP}$-hard, see e.g. Zang et al. [206].

Fixed Routing
The routing problem can be solved in polynomial time as an *all pairs shortest path* problem, see Ahuja et al. [5] for more information. This method is denoted FIXED ROUTING. The definition of *shortest* path varies; the length of a path may be measured in the number of used edges, or in the number of available bandwidths etc., see Birman and Kershenbaum [38] and Choi et al. [48]. In FIXED ROUTING, exactly one path is found per data connection.

Fixed-Alternate Routing
Another routing method is to find several paths between the pair of terminals for all data connections. If the paths for a data connection are edge disjoint, then the approach can be considered somewhat fault tolerant, i.e., if a connection fail on one path, then the corresponding data connection can be routed on the other path. This method is denoted FIXED-ALTERNATE ROUTING, see e.g. Birman and Kershenbaum [38]. When the number of shortest paths for each data connection is limited to $k, k > 0$, then the FIXED-ALTERNATE ROUTING may be referred to as the k-shortest path method, see Banerjee et al. [22] or for a general k-shortest path algorithm, see Eppstein [66]. As there are more paths to choose from, the risk of being unable to assign wavelengths to certain data connection is generally lowered. The wavelength assignment may, though, become harder to solve because of the potential many combinations of paths to choose from.

Adaptive Routing
ADAPTIVE ROUTING is yet another routing method. It consists of finding paths with respect to previously chosen paths. Given is a network with an edge for each pair of fibre and wavelength in the network. An edge has weight 1 when unused and ∞ when used. The path of a data connection request is found as the shortest path with respect

to edge weights. The weights of edges used by this path are set to ∞ and the next data connection request can now be considered. If some nodes have wavelength converters, then an appropriate cost for converting wavelengths can be introduced. See Zang et al. for more details [206].

Least Congested Path Routing
Another ADAPTIVE ROUTING method is the LEAST CONGESTED PATH ROUTING, see Chan and Yum [43]. A sequence of paths is preselected and once a data connection request arrives, the LEAST CONGESTED PATH ROUTING is chosen. Least congestion is measured on the number of available wavelengths on each edge; the congestion of a path is determined by the used edge with fewest available bandwidths.

Shortest Path Adaptive Routing
Yet another method is to use the SHORTEST PATH ADAPTIVE ROUTING, which is an extension of the methods described above. If several paths with same cost exist, then the least congested of those paths is chosen. To determine the least congested path, all edges on all paths for a data connection must be investigated. This can be time consuming, thus Li and Somani [139] have suggested to only check the first k edges.

Routing with Reduction of Wavelength Continuity Conflicts
Recall that when a node does not have a wavelength converter attached, then we say that a path must have wavelength continuity in this node, i.e., a path cannot change wavelength. When several paths compete for the same wavelength on an edge and the start node of that edge does not have a wavelength converter, then we have a *wavelength continuity conflict*. When finding paths for data connection requests, we obviously wish to reduce the number of wavelength continuity conflicts. For this, Koster and Scheffel [125] present a mathematical formulation for finding a lower bound on the number of connections which cannot be routed without wavelength conversion. The bound is based on the number of incident fibres and the number of wavelengths per fibre as shown by Koster in [126]. The mathematical formulation is a variant of the linear MULTICOMMODITY FLOW PROBLEM (MCFP), which is polynomially solvable. Koster and Scheffel solve the formulation using column generation. If the routing problem is solvable, then Koster and Scheffel show that it is possible to assign a wavelength to all selected paths.

Ant Colony Routing
The ANT COLONY ROUTING approach is a metaheuristic. Ants are capable of finding shortest paths when working together: assume that two ants have encountered some food and that two different paths back to the nest exist. Each ant takes its own path; on

their way the ants lay pheromone for signaling. The path of the first ant to arrive at the nest is the shorter of the two paths and it is the only path with pheromone all the way to the nest at this moment. Once the first ant has returned to the nest, a number of ants are sent out towards the food, all leaving pheromone on their way. The strength of the pheromone determines which path the ants choose. Thus all ants will eventually choose the shortest path. The behaviour of ants has inspired the ANT COLONY OPTIMIZATION (ACO). When establishing several paths, a colony of ants is assigned to each path. Ants are only attracted to the pheromone from their own colony. Varela and Sinclair [195] have proposed several ACOs, where ants not only are attracted to pheromone of their own colony; they are also repelled by the pheromone of other colonies.

Genetic Algorithm for Routing
Banerjee et al. [22] use a GENETIC ALGORITHM (GA) for solving the routing problem of RWA. The GA is a metaheuristic. Banerjee et al. seek to minimize the number of used wavelengths *and* the average delay on a network satisfying the wavelength continuity constraint. In GA a number of chromosomes are given; each chromosome consists of a number of genes.

The GENETIC ALGORITHM of Banerjee et al. works as follows: k-shortest path is used as routing heuristic. Each gene in a chromosome represents a path. The cost of each chromosome equals the total cost of the used edges. The cost of an edge depends on the number of paths in the chromosome using that edge. If the edge is only used once, then the cost is relatively low. If the edge is used by several (different) data connection requests, then the cost is very large. Banerjee et al. seek to minimize the cost of selected chromosomes. They thus seek to limit blocking occurring from several paths using the same edge.

Linear programming
The routing problem is formulated mathematically by Zang et al. [206]. The objective is to minimize the maximal number of paths on an edge. Zang et al. argue that this is an INTEGER MULTICOMMODITY FLOW PROBLEM (IMCFP), where a data connection is represented by a commodity with one amount of flow. The IMCFP is $\mathcal{NP}$-hard, see e.g. Barnhart et al. [26], thus Zang et al. suggest reducing the search space by only considering a subset of possible paths. Furthermore, they suggest using random rounding when solving an LP-relaxed formulation.

5.3.1.1 Performance of routing methods

So far the performance of the presented methods has not been discussed. In the literature, the test instances and the objective function vary. An often used objective is

blocking probability, which gives the probability of a data connection request to be blocked, because there is no available wavelength on its path. In this section, we attempt to give an overview of problem instances and results. Despite the difference of used test instances and of objectives, we seek to provide an insight into the overall performance of the proposed methods.

Birman and Kershenbaum [38] compare FIXED ROUTING and FIXED-ALTERNATE ROUTING on a single-hop mesh network with 6 nodes, 9 edges, a data connection request for each pair of nodes, and 24 wavelengths per edge. The objective is blocking probability and their results show, that FIXED-ALTERNATE ROUTING performs better than FIXED ROUTING. No running times are reported.

Chan and Yum [43] test the LEAST CONGESTED PATH ROUTING heuristic on a fully-connected network with 7 nodes and with 30 wavelengths per edge. The computational evaluation is based on changing parameters in the algorithm and in the network. The objective is blocking probability and they test the effect of having different network topologies and different settings for wavelength converters rather than comparing with an existing routing heuristic. Running times are not mentioned.

Furthermore, Li and Somani [139] have compared the LEAST CONGESTED PATH ROUTING heuristic with the shortest path algorithm on a 4×4 mesh-torus network and on the NFS network with 14 nodes and 21 edges. Their objective is blocking probability, and the least congest path routing heuristic has best performance. Running times are not reported.

ROUTING WITH REDUCTION OF WAVELENGTH CONTINUITY CONFLICTS is tested by Koster and Scheffel [125] on a German, European and US network, where the number of eligible paths between two nodes is limited to 100. The number of wavelengths per fibre is set to 40 and 80 in different test runs. In their test, they incorporate the routing scheme in a mathematical formulation for the RWA. They solve the formulation by using CPLEX, version 9.1 and compare different settings of the algorithm instead of comparing with other heuristics. A fixed time limit is set to 10.000 seconds; apart from that, time usage is not mentioned.

Varela and Sinclair [195] test their variants of the ANT COLONY ROUTING approach on three networks. The first has 4 nodes and 20 wavelengths. The second has 9 nodes and 98 wavelengths. The last network has 15 nodes and 269 wavelengths. The objective is to minimize the number of required wavelengths and running times are not considered. The ANT COLONY ROUTING approach is compared to a heuristic with FIXED-ALTERNATE ROUTING like method and with FIRST FIT ASSIGNMENT, and the latter has slightly better performance than the metaheuristic.

Banerjee et al. [22] test the GENETIC ALGORITHM for routing on a number of networks. The considered simulation networks are real life networks: the 20 node ARPA

network, 18 node European optical network, 22 node UK network and 14 node NSF network. Several sets of data connections are tested: 20, 40, 60, 80, and 100 data connections. The objective is to minimize the number of required wavelengths. For less than 80 data connections, the FIRST FIT ASSIGNMENT heuristic and the GENETIC ALGORITHM perform equally well. For 80 or more data connections, the GENETIC ALGORITHM finds better solutions, i.e., solutions requiring fewer wavelengths. Running times are not reported.

5.3.1.2 Theoretical running times

We now report theoretical running times for the presented constructive heuristics for the routing problem. To calculate the times, some notation must be introduced. Given a network, G, let N be the number of nodes and E the number of edges. The number of wavelengths is denoted W and let k be taken from the k-shortest path algorithm. Running times for the heuristics for the routing problem are calculated as the time it takes to find path(s) for each data connection.

The FIXED ROUTING and the ADAPTIVE ROUTING heuristics are shortest path problems, which can be solved in $\mathcal{O}(E + V \log V)$ time using Dijkstra's algorithm, see e.g. Cormen et al. [52].

The FIXED-ALTERNATE ROUTING problem finds the k shortest paths, which can be found in $\mathcal{O}(E + V \log V + k)$ time, see e.g. the work of Eppstein [66].

In the literature, only very large running times are given for the LEAST CONGESTED PATH ROUTING and the SHORTEST PATH ADAPTIVE ROUTING problems, see [139]. Here, we thus present a somewhat naïve algorithm for the LEAST CONGESTED PATH ROUTING with lower running time. The problem consists of finding a path, where the least number of available wavelengths on any used edge is maximized. Now, given a network and a data connection, delete the edge with fewest available wavelengths and set all other edge weights to 0. Solve the shortest path problem using Dijkstra's algorithm. If the problem is solvable, then delete the edge with second fewest available wavelengths. Resolve the problem. Continue until the problem is no longer solvable. Then we know that we have to use the just deleted edge, which has fewer available wavelengths than the remaining edges. This very straight-forward method has running time $\mathcal{O}(E(E + V \log V))$, which can surely be improved. The running time of the SHORTEST PATH ADAPTIVE ROUTING problem is the same, as the problem is a mix of the FIXED-ALTERNATE ROUTING and the LEAST CONGESTED PATH ROUTING.

ROUTING WITH REDUCTION OF WAVELENGTH CONTINUITY CONFLICTS is presented by Koster [126] and is a variant of the polynomially solvable linear MCFP. Koster solves the problem using column generation. To the best of our knowledge, no

constructive solution method for the linear MCFP exists [52]. In the literature, large instances of the linear MCFP are typically solved using Lagrangian methods, partition methods, decomposition techniques, dual ascent algorithms, bundle methods, interior point methods, etc., see Awerbuch and Leighton [18] and Kennington [119] for surveys of the problem and Larsson and Yuan [134] for a review of solution techniques. Small instances are typically solved using the Simplex algorithm. An exact running time for ROUTING WITH REDUCTION OF WAVELENGTH CONTINUITY CONFLICTS is thus difficult to calculate; instead, we simply state that the problem is polynomial.

5.3.2 Wavelength Assignment

When paths are found for all data connections, then wavelengths must be assigned to each path. Wavelength assignment is an $\mathcal{NP}$-hard problem.

In this section, three different types of approaches are described: theoretical results on the number of needed wavelengths, an exact graph coloring approach, and finally a number of heuristics and metaheuristics for the wavelength assignment problem.

The theoretical results on the number of needed wavelengths often depend on the network topology. The research area is quite vast, so we only give a short overview here.

Solving the wavelength assignment problem to optimality is typically done through a graph coloring problem. Much research has been conducted on the graph coloring problem; here we only show the transformation from the wavelength assignment problem to the graph coloring problem and then give references for further information on solution methods.

For the heuristics, we assume that the number of available wavelengths is fixed. The wavelength assignment problem thus consists of finding a feasible solution, rather than finding a feasible solution which minimizes the number of used wavelength. The heuristics may be used for both the static and the dynamic wavelength assignment problem. Each path is treated separately without paying attention to the wavelength assignments of other paths. Some of the heuristics work on both the single-fibre and the multi-fibre network.

Theoretical Results on the Number of Needed Wavelengths
Once routing is done, wavelengths are to be assigned to the data connections. Much research is done on theoretical bounds on the number of required wavelengths. Especially lower bounds on the number of wavelengths are given, i.e., given a set of paths then at least a certain number of wavelengths are needed for assignment of those paths.

The bounds can be used to quickly determine whether or not all data connections can be assigned wavelengths. The bounds, however, often depend on the chosen routing algorithm. Work has also been performed on upper bounds; these bounds can be used to ensure feasibility, i.e., given a routing and given a number of available wavelengths larger than the upper bound, then a feasible wavelength assignment is guaranteed.

The research area of bounds on wavelengths is vast, as much work is done on specific network topologies. In the following, a selection of results from the research area is presented.

First, Aggarwal et al. [4] present previous work on lower and upper bounds in wavelength assignment and then Aggarwal et al. improve the upper bounds. Their bounds apply for specific instances of the RWA. Two variants of the dynamic RWA is considered: (1) all data connections can always be rerouted and (2) no data connection can ever be rerouted. Furthermore, they make the assumption that FIXED ROUTING is used. The network topologies include star networks, having no converters or having converters at all nodes. Aggarwal et al. find upper bounds close to previously found lower bounds. For more details on their bounds and on earlier found bounds, see the overview of previous work presented by Aggarwal et al.

Raghavan et al. [162] present heuristic algorithms for the static RWA on certain network topologies. The algorithms have bounds on the number of wavelengths needed. The network topologies include sparse, bounded degree rings, trees, and meshes, all with constraints on how to forward data in a node. Furthermore, Raghavan et al. discuss using their algorithms for the dynamic RWA.

When calculating bounds, Barry and Humblet [29] allow blocking; that is, some data connections may be blocked instead of the telecommunication provider upgrading the network. The same applies for Ramaswami and Sivarajan [164], and Yates et al. [205].

Gersel et al. [91] present algorithms with known worst upper bounds on the number of wavelengths needed for the RWA with no blocking. Their work is on certain undirected network topologies: line, ring networks and trees, all with no wavelength converters. Furthermore, they extend their results when wavelength conversion is allowed. Their algorithms are greedy heuristics, where they have added lower and upper bounds on the number of wavelengths to avoid blocking.

Koster [126] solves the wavelength assignment problem by transforming it into an *edge coloring* problem. This transformation is only possible, when no path uses more than two edges, which is the case in a star network. Koster gives lower bounds on the number of wavelengths to assign. In the case that all paths must be assigned wavelengths, Koster gives a lower bound on the number of needed wavelength converters.

The results in this section suffer from only working on specific instances of the RWA

and of the underlying network. Furthermore, some results do not take blocking into account. Much more work has been conducted on finding bounds for the number of wavelengths, but to the best of out knowledge, all this work depends on constraints set on the network topology, on the paths to assign wavelengths, etc.

Graph Coloring

The wavelength assignment problem can be solved using graph coloring methods, see e.g. Zang et al. [206]. Garey et al. [90] prove that the graph coloring problem is $\mathcal{NP}$-hard. For more general information on the graph coloring problem, see Jensen and Toft [109]. An auxiliary graph G' is constructed such that each path in the routing solution is represented by a node in G' and such that two nodes are connected in G' if the corresponding paths travel on the same fibre in the routing solution. Now, the graph coloring problem is to assign colors to all nodes in G' such that two adjacent nodes do not share the same color. This corresponds to assigning wavelength to paths such that two paths using the same fibre do not share wavelength. In graph coloring, the *chromatic number* denotes the minimum number of needed colors. Minimizing the chromatic number thus corresponds to minimizing the number of needed wavelengths. The graph coloring problem solves the wavelength assignment problem to optimality.

For information on exact, heuristic and approximate graph coloring algorithms, we refer to Pardalos et al. [158] and to the bibliography maintained by Chiarandini [46].

Random Assignment Heuristic

The RANDOM ASSIGNMENT algorithm consists of assigning a random available wavelength to each path. If FIXED ROUTING is used, then the RANDOM ASSIGNMENT algorithm is straight-forward. If FIXED-ALTERNATE ROUTING or another routing protocol is used, where each data connection request can choose from several paths, then RANDOM ASSIGNMENT chooses a path, which can be assigned a wavelength. If more than one path can be assigned a wavelength, then the algorithm randomly selects one of these.

The RANDOM ASSIGNMENT is used by e.g. Subramaniam and Barry [183].

First Fit Heuristic

This FIRST FIT ASSIGNMENT method consists of assigning the first available wavelength to the current path. How the first available wavelength is defined is not that relevant, as long as the order of wavelengths is predefined. The FIRST FIT ASSIGNMENT heuristic is widely used, see e.g. the work of Kovacevic and Acampora [130].

Least Used Heuristic
The LEAST USED ASSIGNMENT heuristic selects the wavelength that is least used so far. The idea is to balance the load among all wavelengths. This approach, however, causes trouble for longer paths, as different wavelengths are all used throughout the network. Hence, the approach eventually only assigns wavelengths to short paths. For more details, see Mokhtar and Azizoglu [150] or Zang et al. [206].

Most Used Heuristic
The MOST USED ASSIGNMENT approach is the opposite of the LEAST USED ASSIGNMENT heuristic. Instead of selecting the least used wavelength, this heuristic chooses the wavelength which is most used in the network. The MOST USED ASSIGNMENT heuristic is described in details by Mokhtar and Azizoglu [150] and Zang et al. [206].

Exhaustive Search Heuristic
The EXHAUSTIVE SEARCH ASSIGNMENT algorithm works on top of FIXED-ALTERNATE ROUTING or another routing scheme generating several paths per data connection request. The wavelength assignment heuristic checks all available wavelengths and chooses the one, which gives the shortest path. Mokhtar and Azizoglu [150] argue that the method has quite high complexity as it needs to check all wavelengths on all paths.

Minimum Product Heuristic
The MINIMUM PRODUCT ASSIGNMENT approach consists of minimizing the number of fibres used in a multi-fibre network and is introduced by Jeong and Ayanoglu [110]. Let D_{ij} denote the number of assigned fibres on edge i and for wavelength j. Then this heuristic calculates $\Pi_i D_{ij}$ for all wavelengths j.

Least Loaded Heuristic
The LEAST LOADED ASSIGNMENT approach is also designed for a multi-fibre network. Given a path, the heuristic finds the wavelength, whose smallest availability is larger than that for all other wavelengths. Let M_i be the number of fibres on edge i and let D_{ij} be the number of assigned fibres on edge i for wavelength j. Then the LEAST LOADED ASSIGNMENT approach selects the wavelength j with $\max_j \min_i (M_i - D_{ij})$. For more details, see Zang et al. [206] and Karasan and Ayanoglu [117].

Maximum Sum Heuristic
Subramaniam and Barry [28, 183] present a MAXIMUM SUM ASSIGNMENT algorithm

for assigning wavelengths. Given is a network, where paths are preselected. Now, when a new data connection request arrives and a path is found, the heuristic of Subramaniam and Barry seeks to find a wavelength, where after assignment the remaining capacity is as large as possible. Subramaniam and Barry designed the algorithm for multi-fibre network, but it also applies for single-fibre networks.

Relative Capacity Loss Heuristic

The RELATIVE CAPACITY LOSS ASSIGNMENT heuristic is introduced by Zhang and Qiao [207] and it is a variant of the MAXIMUM SUM ASSIGNMENT approach. The latter selects the wavelength, which minimizes the capacity loss (or maximizes the remaining capacity) on all edges. The RELATIVE CAPACITY LOSS ASSIGNMENT chooses the wavelength which minimizes the relative capacity loss, i.e., the capacity loss divided with the available capacity.

Distributed relative capacity loss heuristic

Zang et al. [206] propose the DISTRIBUTED RELATIVE CAPACITY LOSS ASSIGNMENT heuristic for assigning wavelength. The algorithm is a variant of the RELATIVE CAPACITY LOSS ASSIGNMENT heuristic. It reduces complexity of the former heuristic by generating a look-up table, such that the relative loss capacity of wavelengths is readily available. The look-up table is built by investigating the network and by exchanging information between nodes in a manner similar to that of the Bellman-Ford shortest path algorithm, see Cormen et al. [52] for the Bellman-Ford algorithm.

Wavelength Reservation Heuristic

As the name of the WAVELENGTH RESERVATION ASSIGNMENT heuristic indicates, this method reserves wavelengths for certain data connections. An example is that a wavelength λ is reserved for all data going from a node a to node c. If several paths have a and c as intermediate nodes, then they compete for the reserved wavelength, λ. Note that another wavelength assignment method must be used to determine which path to select for the current data connection and which wavelength to reserve. Birman and Kershenbaum [38] introduce the wavelength heuristic approach for multi-hop connections and they show that it reduces the blocking for multi-hop connections, but it also increases the blocking for single-hop connections.

Protecting Threshold Heuristic

Birman and Kershenbaum [38] introduce the PROTECTING THRESHOLD ASSIGNMENT approach, which consists of only selecting a wavelength when the number of idle wavelengths on the edge is above a certain threshold. Note that another wavelength assignment must be used to determine which path to select for the current data

connection and which wavelength to assign to the path. Birman and Kershenbaum have developed the heuristic for single-hop data connections.

Genetic algorithm
GENETIC ALGORITHMS (GA) try to simulate evolution of genotypes and natural selection, see e.g. Goldberg [93]. Hyytiä and Virtamo [101] suggest a GA for solving the wavelength assignment problem as a graph coloring problem. Two chromosomes are given, each representing a solution to the graph coloring problem. A new chromosome is generated from the two previous chromosomes; the reuse of a chromosome depends on the quality of the corresponding solution (which is the number of used wavelengths). The new chromosome represents a solution to the wavelength assignment problem.

Simulated annealing
SIMULATED ANNEALING (SA) is based on resolving the problem and accepting a new and better solution with some probability. This probability depends on a *temperature* parameter, which decreases with time. Hence, the name simulated *annealing*. For more details, see van Laarhoven and Aarts [193]. Hyytiä and Virtamo [101] present a SA approach used on the wavelength assignment problem. The problem is considered as a graph coloring problem and the SA consists of assigning different colors to nodes, calculating the objective cost, i.e., the number of used wavelength, and then accepting the new solution with some probability.

Tabu search
Finally, TABU SEARCH (TS) is based on a random search approach where certain moves are forbidden or *tabu*, see e.g. Glover and Laguna [92]. Hyytiä and Virtamo [101] suggest solving the wavelength assignment problem represented by a graph coloring problem, by using TABU SEARCH. The objective is to maximize the number of established connections rather than to minimize the number of used wavelengths.

Bin Packing Heuristic
The RWA on a network with no wavelength converters can be solved by applying the bin packing problem. For more information on the bin packing problem, see Pisinger and Sigurd [161]. Skorin-Kapov [179] represents the RWA as a bin packing problem by letting paths be items and by letting copies of the network be bins. Each bin represents a wavelength, and each bin has capacity equal to the number of edges in the network. Two items cannot be packed in the same bin if the corresponding paths use the same edge. Now, the bin packing problem is to pack items into as few bins as possible. This corresponds to minimizing the number of assigned wavelengths.

5.3.2.1 Performance of wavelength assignment methods

Again, the performance of the presented methods has not been discussed, because in the literature the test instances and the objective function vary. In this section, we give an overview of the performance of the wavelength assignment methods, including a description of the evaluated problem instances and the corresponding evaluation results.

Kovacevic and Acampora [130] compare the FIRST FIT ASSIGNMENT heuristic for wavelength assignment with the RANDOM ASSIGNMENT approach. The test instance is a 11×11 mesh network with 5 wavelengths per edge and with varying network load. The objective is blocking probability and the results show that the FIRST FIT ASSIGNMENT heuristic generally gives better results than the RANDOM ASSIGNMENT. Running times are not mentioned.

Mokhtar and Azizoglu compare the EXHAUSTIVE SEARCH ASSIGNMENT with the MOST USED ASSIGNMENT algorithm, the FIRST FIT ASSIGNMENT, and RANDOM ASSIGNMENT[150]. The test instances are two networks: the ARPA-2 network with 21 nodes, 26 edges and 4 or 8 wavelengths, and a randomly generated topology with 15 nodes and 32 edges. Traffic arrives according to the Poisson process. The objective is blocking probability. The MOST USED ASSIGNMENT, RANDOM ASSIGNMENT and LEAST USED ASSIGNMENT heuristics are tested on both networks. The MOST USED ASSIGNMENT heuristic performs best, followed by RANDOM ASSIGNMENT. Then the EXHAUSTIVE SEARCH ASSIGNMENT algorithm is compared to the MOST USED ASSIGNMENT. The EXHAUSTIVE SEARCH ASSIGNMENT algorithm gives slightly better results, but Mokhtar and Azizoglu note that the increased complexity of the EXHAUSTIVE SEARCH ASSIGNMENT overshadows the better results. FIRST FIT ASSIGNMENT is compared with the MOST USED ASSIGNMENT heuristic and FIRST FIT ASSIGNMENT performs almost equally well to the MOST USED ASSIGNMENT method. Time usage is not given, but theoretical complexities are computed for the heuristics.

Karasan and Ayanoglu [117] implement the LEAST LOADED ASSIGNMENT heuristic. They test it on a 30-node mesh network where traffic is distributed uniformly. The network reflects the geographical location of major cities in the US. Connection requests arrive according to the Poisson process. The network is either single-fibre or multi-fibre, each fibre having 8 wavelengths. The objective is blocking probability. Results show that the LEAST LOADED ASSIGNMENT heuristic performs better than the MOST USED ASSIGNMENT approach.

Subramaniam and Barry [183] test the RANDOM ASSIGNMENT, FIRST FIT ASSIGNMENT, LEAST LOADED ASSIGNMENT, MOST USED ASSIGNMENT, MINIMUM PRODUCT ASSIGNMENT and the MAXIMUM SUM ASSIGNMENT heuristics. The instances have uniform Poisson traffic and are either a 20 node ring network with 1 or 10 fibres per edge, or a 5×5 bidirectional mesh-network with 1 or 3 fibres per edge. Subra-

maniam and Barry use blocking probability as objective. Running times are not mentioned. According to Subramaniam and Barry the MINIMUM PRODUCT ASSIGNMENT heuristic performs slightly better than the MOST USED ASSIGNMENT heuristic with respect to blocking probability. Then follows the FIRST FIT ASSIGNMENT, LEAST LOADED ASSIGNMENT, MAXIMUM SUM ASSIGNMENT and finally the RANDOM ASSIGNMENT heuristics.

Zhang and Qiao [207] test the FIRST FIT ASSIGNMENT, the MAXIMUM SUM ASSIGNMENT approach and the RELATIVE CAPACITY LOSS ASSIGNMENT heuristic on a simulation of the NFS network and on a 4×4 torus network. They use blocking probabilities to calculate their objective function value. The RELATIVE CAPACITY LOSS ASSIGNMENT method has best performance.

Zang et al. [206] compare a number of heuristics for wavelength assignment: RANDOM ASSIGNMENT, FIRST FIT ASSIGNMENT, LEAST USED ASSIGNMENT, MOST USED ASSIGNMENT, MINIMUM PRODUCT ASSIGNMENT, LEAST LOADED ASSIGNMENT, MAXIMUM SUM ASSIGNMENT, and RELATIVE CAPACITY LOSS ASSIGNMENT. A network consisting of six nodes is used for testing, where the number of wavelengths and fibres vary. The objective is blocking probabilities and practical running times are not mentioned. In a single fibre network, the MOST USED ASSIGNMENT heuristic performs well, along with the MAXIMUM SUM ASSIGNMENT and RELATIVE CAPACITY LOSS ASSIGNMENT approaches when the load is low. When the load is high, then all heuristics have similar performance. In a multi-fibre network, the MOST USED ASSIGNMENT, MINIMUM PRODUCT ASSIGNMENT and RELATIVE CAPACITY LOSS ASSIGNMENT methods have best performance, while the LEAST LOADED ASSIGNMENT and MAXIMUM SUM ASSIGNMENT heuristics work best with high load. Zang et al. conclude, however, that the difference between the performances of all heuristics is quite insignificant.

Birman and Kershenbaum [38] compare the WAVELENGTH RESERVATION ASSIGNMENT and the PROTECTING THRESHOLD ASSIGNMENT heuristics on a single-hop mesh networks with 6 nodes, 9 edges, a data connection request for each pair of nodes, and 24 wavelengths per edge. The objective is blocking probability and the results show that the PROTECTING THRESHOLD ASSIGNMENT algorithm tends to give better results than the WAVELENGTH RESERVATION ASSIGNMENT approach. No running times are given.

The metaheuristics suggested by Hyytiä and Virtamo [101] include a GENETIC ALGORITHM, SIMULATED ANNEALING and TABU SEARCH. The methods are compared with each other and with a FIRST FIT ASSIGNMENT heuristic, on randomly generated instances not described any further. The results show that the greedy heuristic has significantly better running time. The GENETIC ALGORITHM has better running time than the SIMULATED ANNEALING, which is faster than the TABU SEARCH. The methods are also compared with respect to the number of generated wavelengths. Here,

the TABU SEARCH has best performance, followed by the GENETIC ALGORITHM, the FIRST FIT ASSIGNMENT, and finally the SIMULATED ANNEALING.

Skorin-Kapov [179] tests the BIN PACKING HEURISTIC on a series of random 100-node networks with average degrees of 3, 4, and 5. Random sets of data connections requests were created for each test network with a fixed probability of there being a data connection request between two nodes. The number of requests varies from 2054 to 9900. The objective is to minimize the number of required wavelengths along with the length, in hops, of data connections. The results show that the heuristics find optimal or near-optimal solutions. Running times are mentioned to be low in general: solving an instance with 100 nodes and 9900 data connection requests takes less than 8 minutes on a P4 2.8 GHz processor.

Zang et al. [206] argue that the routing algorithm has larger influence on the amount of blocking probability, than the wavelength assignment algorithm. They base this on the performed tests, where algorithms using ADAPTIVE ROUTING generally gives significantly better results than algorithms using FIXED ROUTING - no matter which wavelength assignment algorithm is used. Zang et al., however, do not take running times into account, so even if more complicated routing algorithms give better solutions, one could fear that the algorithms may also have larger time usage.

5.3.2.2 Theoretical running times

We now report theoretical running times for the presented constructive heuristics for the wavelength assignment. Recall the notation: given a network, G, let N be the number of nodes and E the number of edges. The number of wavelengths is denoted W and let k be taken from the k-shortest path algorithm. Running times for the wavelength assignment heuristics are calculated as the time it takes to assign a wavelength to a single path.

The RANDOM ASSIGNMENT selects a random wavelength. In the case of no wavelength converters, the heuristic investigates all edges on the path to see if the wavelength is available; if not, it repeats the process with another randomly picked wavelength. In the case of wavelength converters, the heuristic investigates if the wavelength is available on each edge and if not, it selects another wavelength and check again. The running time is $\mathcal{O}(WE)$. The FIRST FIT ASSIGNMENT only differs in how to pick the wavelength and it thus has the same running time.

The LEAST USED ASSIGNMENT and MOST USED ASSIGNMENT heuristics run through all used edges and calculate how much each wavelength is used. The wavelengths are sorted according to usage and paths are assigned wavelengths from the sorted list in a FIRST FIT ASSIGNMENT manner. The running time is $\mathcal{O}(W \log W + WE)$.

The EXHAUSTIVE SEARCH ASSIGNMENT needs to check all available wavelengths on all k paths for the current data connection. This takes $\mathcal{O}(kWE)$ time.

The MINIMUM PRODUCT ASSIGNMENT heuristic calculates the product $\Pi_i D_{ij}$ for all fibres i and for all wavelengths j. This takes $\mathcal{O}(WE)$ time. The LEAST LOADED ASSIGNMENT heuristic is very similar to the MINIMUM PRODUCT ASSIGNMENT method and thus has the same running time, $\mathcal{O}(WE)$.

The MAXIMUM SUM ASSIGNMENT heuristic investigates how much each wavelength is available on each edge of all the paths, the current data connection can choose from. Let the number of paths be bounded by k; the running time is $\mathcal{O}(kWE)$. The RELATIVE CAPACITY LOSS ASSIGNMENT heuristic and the DISTRIBUTED RELATIVE CAPACITY LOSS ASSIGNMENT heuristic work in a similar manner and thus have the same running time, $\mathcal{O}(kWE)$.

Finally, the WAVELENGTH RESERVATION ASSIGNMENT heuristic and the PROTECTING THRESHOLD ASSIGNMENT heuristic are used on top of other wavelength assignment algorithms. Thus, their running times depend on the other heuristic: the WAVELENGTH RESERVATION ASSIGNMENT and PROTECTING THRESHOLD ASSIGNMENT methods themselves have constant running time, $\mathcal{O}(1)$.

5.4 Overall methods for solving the RWA problem

In this section, important results for solving the RWA as one problem are presented. Instead of splitting the RWA into two subproblems, the following methods approach the entire RWA. Methods include both metaheuristics and exact formulations.

5.4.1 Metaheuristics

In this section, metaheuristics for solving the RWA are presented. The metaheuristics proposed in the literature are GENETIC ALGORITHM (GA) and ANT COLONY OPTIMIZATION algorithms (ACO).

Ant Colony Optimization algorithm
Arteta et al. [16] use a MULTI-OBJECTIVE (MO) ACO metaheuristic for solving the RWA. ACO defines a method of investigating the neighbourhood of a current solution. The MO consists of optimizing the hop count and the number of wavelength conversions. In the ACO this means that the pheromone matrix, i.e., the probabilities

defining which pheromone track an ant chooses, depends on the path's hop count and on the number of wavelength conversions in the path.

Arteta et al. have implemented several MOACOs: for more details on each MOACO, see the corresponding reference. The MULTIPLE OBJECTIVE ANT Q ALGORITHM (MOAQ) of Mariano and Morales [144] maintains a colony per objective. The BI-CRITERION ANT (BIANT) of Iredi et al. [105] uses a probability matrix per objective and hence also a colony per objective. PARETO ANT COLONY OPTIMIZATION (PACO) of Doerner et al. [58] has several pheromone matrices for each objective. The MULTI-OBJECTIVE ANT COLONY SYSTEM (MOACS) by Schaerer and Barán [25] uses several heuristics when calculating entries in the probability matrix. The MULTI-OBJECTIVE MAX-MIN ANT SYSTEM (M3AS) by Pinto and Barán [159] has a global pheromone matrix. COMPETANTS (COMP) by Doerner et al. [59] uses several heuristics, pheromone matrices and the colony sizes vary. MULTI-OBJECTIVE OMICRON ACO (MOA) by Gardel el al. [88] uses a specific updating rule for the pheromone matrices, and finally MULTI-OBJECTIVE ANT SYSTEM (MAS) by Paciello et al. [157] has a slightly different order of updating the pheromone matrices.

Genetic Algorithms

Sinclair [178] solves the RWA through a GENETIC ALGORITHM (GA). Instead of using the classical mutation and crossover operations in GA, Sinclair uses heuristics to generate new solutions. The heuristics are: k-shortest path routing with FIRST FIT ASSIGNMENT, rerouting and reassignment of wavelengths of a subset of connections, rerouting a path with high wavelength in order to reach the lowest possible wavelength, and shifting the path with the highest wavelength to having a lower wavelength such that all paths blocking the new low wavelength must be rerouted.

Ali et al. [6] solve a variant of the RWA problem using a GENETIC ALGORITHM. The variant consists of taking power into account, i.e., they wish to preserve proper power levels on all paths. They use a k-shortest path method to generate routes, where power loss is taken into account when measuring the length of a path.

5.4.2 Linear programming

This section presents methods from the literature for finding LP bounds for the RWA. Several of the methods presented in the following may be integer or mixed integer programs, but the suggested solution methods all work on LP relaxed formulations.

Ramaswami and Sivarajan [164] present an INTEGER PROGRAMMING (ILP) formulation for the static RWA with no wavelength conversion and where the objective is

to maximize the number of established data connections. They note that their model is a variant of the MCFP. Given the data connections and corresponding paths, Ramaswami and Sivarajan solve the problem using rounding algorithms. Data connections and paths are generated randomly.

An ILP formulation of the static RWA is presented by Zang et al. [206]. Wavelength conversion is not allowed and the objective is to minimize the maximal edge flow. It is noted that the model is a variant of the MCFP. Zang et al. also present an overview of a model for the static RWA with wavelength conversion, which again is a variant of the MCFP.

Banerjee and Mukherjee [21] present an ILP for the RWA, where the objective is to minimize the hop distance. The network allows wavelength conversion. They, however, solve the problem heuristically. Banerjee and Mukherjee argue that their model can be used to design a balanced network with high utilization of transceivers and wavelengths. Furthermore, it is noted, that the model of Banerhee and Mukherjee is a variant of the MCFP, where each commodity represent a data connection.

Ozdaglar and Bertsekas [156] work on an ILP formulation of the quasi-static RWA. They define quasi-static RWA to be the problem, where several data connection requests first are to be handled and then later more data connection requests may arrive. The formulation is a variant of the MCFP. Ozdaglar and Bertsekas relax the ILP and show that the relaxed formulation yields integer solutions for several network topologies including line and ring networks, with wavelength converters at either all or no nodes.

Jaumard et al. [106] present a number of different ILP formulations for the RWA in WDM optical networks, using a unified notation. The variants of the RWA include instances with symmetric and with asymmetric traffic. Jaumard et al. show edge- and path-based formulations as well as models from the literature. Formulations for the RWA with asymmetric traffic are shown to give the same optimal solution value, though the number of constraints and variables differ.

5.4.3 Integer programming

This section presents exact solution methods for the RWA. The methods are all based on Dantzig-Wolfe decomposing the RWA, see [54]. The resulting formulations are solved to optimality using branch-and-price, where the master and subproblems vary according to the used Dantzig-Wolfe decomposition.

If wavelengths may be changed in every node, the RWA problem can be reduced to the INTEGER MULTICOMMODITY FLOW PROBLEM (IMCFP), see Beauquier et al. [31].

The IMCFP consists of sending an amount of flow between several sources and targets with respect to edge capacities, see Ahuja at el. [5] for more details. When wavelengths can be converted at all nodes, then the wavelength limitation can be described as edge capacities: each edge can carry at most k different wavelengths, for some integer $k > 0$. Now, we need to send 1 amount of flow between all data connection terminals without violating edge capacities. This corresponds to the integer MCFP. The integer MCFP is a well-studied problem with many solution approaches. An example is the branch-and-bound algorithm by Barnhart et al. [26].

Another ILP formulation for the RWA is of Lee et al. [138] which is based on finding a set of paths with the same wavelength for a subset of data connection. The formulation maximizes the number of established data connections subject to the RWA constraints. Lee et al. propose a column generation for the formulation, where the subproblem is to find a set of paths with the same wavelength for some data connections. To find an optimal solution Lee et al. present a branch-and-price algorithm.

Jaumard et al. [107] analyze column generation formulations for the RWA from the literature and present a new formulation. First a straight forward path formulation of the RWA is presented, where a path consists of both the visited edges and the used wavelengths. It is argued that the formulation yields symmetry problems with respect to the used wavelengths. Then Jaumard et al. review the formulation of Ramaswami and Sivarajan [164] where wavelength assignment and path variables are kept separately. Jaumard et al. propose a column generation method for generating paths for the formulation, however, the method has some drawbacks: the size of the subproblem depends on the number of paths for a data connection which may be exponential and the column generation technique solves the LP relaxed formulation and does thus not return an optimal solution to the original problem. Jaumard et al. present the formulation of Lee et al. based on finding a set of paths with the same wavelength for a subset of data connections. Jaumard et al. suggest solving the subproblem as a multicommodity linear flow problem. Based on the formulations of Ramaswami and Sivarajan [164] and Lee et al. [138], Jaumard et al. propose a new mathematical model where each column consists of a set of paths for a subset of data connections and where wavelengths are assigned in the master problem. A branch-and-price algorithm is presented where the subproblem corresponds to that of the formulation of Lee et al. and the branching strategy add cuts on the number of used wavelengths to the master problem. Jaumard et al. have implemented and tested the column generation formulation of Lee et al. and of their own model.

5.4.4 Comparison of overall solution methods

Once again, the test instances and the objective function vary in the literature. An overview of tested instances and corresponding results for the overall solution methods

is presented in this section.

Arteta et al. [16] test their MOACO metaheuristics for solving the RWA on the Japanese NTT network topology. The network has 55 nodes and 144 edges. The algorithms were run 10 times, each time of at most 100 iterations. The objective is to minimize the amount of wavelength conversion and the hop length, along with pareto front and error. Running times are not considered. Using this objective, the MOACOs outperform simpler, greedy heuristics.

Sinclair [178] solves the RWA through a GENETIC ALGORITHM. Five test networks are generated, each with 15 nodes, and with 34 to 39 edges. The objective is to minimize the cost of used edges and running times are not taken into account. Sinclair shows that the proposed GENETIC ALGORITHM can compete with greedy heuristics.

Ali et al. [6] solve a variant of the RWA problem using a GENETIC ALGORITHM. They test their algorithm on a network with 13 nodes and the objective is to maximize the number of established data connections and in time usage. The proposed GENETIC ALGORITHM outperforms a FIRST FIT ASSIGNMENT like heuristic with respect to the number of data connections, but it spends significantly more time.

Ramaswami and Sivarajan [164] present an ILP. They solve the problem using a rounding method and they compare their bounds with a FIRST FIT ASSIGNMENT like heuristic. The test instances are two networks with data connection requests arriving according to a Poisson process and lasting for a duration that is exponentially distributed. The networks are a 5 node pentagon and a 20 node network representing a skeleton of ARPA, respectively. First off, Ramaswami and Sivarajan show that they reach their theoretically calculated bounds on carried traffic. They compare their rounding method for the ILP with the heuristic with respect to blocking probability and their rounding method gives best results. Running times are not taken into account.

Banerjee and Mukherjee [21] present an ILP to derive a minimal hop distance solution in a network with wavelength converters. Two heuristics are proposed: one which attempts to find paths between the node pairs, which have more data connection requests and which are only separated by a single hop. The other heuristic attempts to maximize the number of established data connections with respect to the number of hops between the sources and targets. Banerjee and Mukherjee test the heuristics and the ILP on the NFS network with a randomly generated traffic matrix. They show that the average packet hop distance for the heuristic solutions is not far from that obtained by the ILP. Running times are not mentioned.

Jaumard et al. [106] test the models on NSF and EON networks with asymmetrical traffic matrices, which correspond to 268 connections for the NSF instance and 374 for the EON. For symmetrical traffic, the former are modified such that for a pair of nodes s, d, then the selected connections are the connections from s to d, unless the number

of connections from d to s is larger. This gives 191 connections for the NSF and 270 for the EON. Formulations are compared through computational evaluation and they show that benchmark problems from Krishnaswamy and Sivarajan [131] can be solved to optimality or with a small gap. Only bounds are compared in the computational study, hence running times are not mentioned.

Lee et al. [138] test their branch-and-price algorithm using test instances based on the SONET ring topology with 10, 15 and 20 nodes and where each node pair requires one to three data connections. Their test results show that the bounds found in the root node of the branch-and-bound tree are of good quality and optimal solutions are found for the majority of instances. An upper bound on 20.000 branch-and-bound nodes is applied. Small instances are solved to optimality in seconds, while larger instances take up to 15 minutes to solve.

In the later work of Jaumard et al. [107], the column generation algorithm from Lee et al. [138] and the branch-and-price algorithm for the new formulation proposed by Jaumard et al. are implemented. They are tested and compared with solving an edge-based formulation to optimality using CPLEX. The test instances are modified NSF and EON benchmarks taken from Krishnaswamy and Sivarajan [131]. Some edges are removed from the NSF instances and extra data connections are added to the EON instances. Finally some test instances resembling a Brazilian network topology proposed by Noronha and Ribeiro [155] are used. The computational results show that the branch-and-price algorithm finds better bounds than the column generation method by Lee et al. and in less time. Furthermore, the branch-and-price algorithm is capable of finding an optimal solution for the far majority of instances and thus finds more optimal solutions than when using CPLEX on the edge-based formulation. Running times for the branch-and-price and column generation algorithms span from less than a minute for smaller instances up to days for the larger instances.

5.5 Conclusion

A wide variety of solution methods for the RWA have been presented. Most work in the literature is based on heuristics, more specifically on dividing the RWA into two parts: the routing problem and the wavelength assignment problem. For the main part, the heuristics apply on both the static and on the dynamic RWA.

Some work has also been concentrated on metaheuristics, both for the routing problem, the wavelength assignment problem, but also for the entire RWA. The metaheuristics work on the static RWA, as they generally seek to iteratively improve a solution.

Less work is based on finding optimal solutions to the static RWA. In the literature it

is argued that since the RWA is $\mathcal{NP}$-hard, then finding an optimal solution is too hard. The exact solution approaches presented and tested in the literature, however, perform fairly well.

In this survey, experimental results from the literature and theoretical running times are presented. A general issue for comparing solution methods is the inconsistency in test instances and objective functions.

Running times seem to be of little interest in most experiments performed on the proposed methods. In this case, we believe that future work should focus on the MCFP representation of the problem. The RWA is a variant of the well-studied MCFP, thus algorithms for the MCFP need to be modified, when solving the RWA.

If running times are of interest, then the heuristics for the decomposed RWA seem to give good results fast. All greedy heuristics run in polynomial time and their theoretical running times are generally small.

When focusing on solution qualities, then the most used objective is blocking probability. This is relevant given instances, where not all data connections can be established and given that no general benchmark instances are used. Blocking probability tries to give a measure for the probability of the establishment of a data connection. We, however, fear that this objective is difficult to compare across the many different types and sizes of problem instances. We thus recommend the use of general instances, e.g., like the Solomon benchmark instances are used for the Vehicle Routing Problem with Time Windows [180]. General benchmark instances for the RWA could be generated randomly, be based on known problems from general graph theory, or from some of the widely used test instance libraries available. E.g., several benchmark instances for mixed integer problems are found in the `MIPlib` (`http://miplib.zib.de/`), and a data library for fixed telecommunication network design is found in `SNDlib` (`http://sndlib.zib.de`).

As is the case in most situations dealing with $\mathcal{NP}$-hard problems, the trade-off lies between solution quality and time usage. Optimal solutions are generally only reached quickly, when the problem instances are very small. A large part of the networks, which are used for testing in the literature, are not too large. For the static RWA problem, it may thus be beneficial to focus more on MCFP formulations of the RWA problem. The MCFP and many variants hereof are well-studied and many exact algorithms with good performance are presented in the literature. For example, the branch-and-price-and-cut algorithm for the $\mathcal{NP}$-hard IMCFP by Barnhart et al. [26] solves instances with up to nearly 93 commodities, 29 nodes and 61 edges to optimality. As another example, instances for the linear MCFP with up to 80.000 commodities, 3600 nodes and 14.000 edges are solved to near-optimality by a Lagrangian algorithm presented by Larsson and Di Yuan [134].

For the dynamic RWA, the heuristics for the decomposed RWA have good performance and we believe that any further work should concentrate on either these heuristics or on heuristics for the entire RWA.

In this survey, network design has been left out. From the perspective of a telecommunications provider, however, network design may be important, as optical networks are constantly being extended in order to reach new customers. The research area for network design is vast, thus a separate survey for this area should be consulted for further details, see e.g. Dutta and Rouskas [62], Iness [103], Jue [114] or Zymolka [209].

Solving the RWA can be used in several contexts. A solution can decide which data connections to establish. The objective may be to maximize the number of established connections, to minimize the cost of setting up connections, to minimize delays on established connections, to minimize blocking, etc. Furthermore, solution methods can be used as an analytic tool to measure performance, to measure which parts of the network is subject to most usage etc. The presented solution methods have a trade-off between solution quality and time usage. When solving the RWA, it is thus important to decide which is more important; solution quality or time usage.

Acknowledgements
We would like to thank GlobalConnect A/S for supporting this work.

CHAPTER 6

On the integrated job scheduling and constrained network routing problem

Mette Gamst
DTU Management Engineering, Technical University of Denmark

This paper examines the problem of scheduling a number of jobs on a finite set of machines such that the overall profit of executed jobs is maximized. Each job demands a number of resources, which must be sent to the executing machine via constrained paths. A job cannot start before all its demand has arrived at the machine. Furthermore, two resource demand transmissions cannot use the same edge in the same time period. The problem has application in grid computing, where a number of geographically distributed machines work together for solving large problems. The machines are connected through an optical network.

The problem is formulated as a MIP problem and is shown to be $\mathcal{NP}$-hard. An exact solution approach based on Dantzig-Wolfe decomposition is proposed. Also, several

In submission 2010

heuristic methods are developed by combining heuristics for the job scheduling problem and for the constrained network routing problem.

The methods are computationally evaluated on test instances arising from telecommunications with up to 500 jobs and 500 machines. Results show that solving the integrated job scheduling and constrained network routing problem to optimality is very difficult. The exact solution approach performs better than using a standard MIP-solver; however, it is still unable to solve several instances. The proposed heuristics generally have good performance. Especially the First Come First Serve scheduling heuristic combined with a routing strategy, which proposes several good routes for each demand, has good performance with an average solution value gap of 3%. All heuristics have very small running times.

Key words: Job Scheduling; Network Routing; Routing and Wavelength Assignment; Grid Computing; Heuristics; Branch-and-Bound; Dantzig-Wolfe Decomposition; Column Generation;

6.1 Introduction

Heuristic and exact solution methods for The Integrated Job Scheduling and Constrained Network Routing Problem (JSCNR) are presented. The JSCNR consists of scheduling jobs on machines with respect to job demand transmission in an undirected constrained network. The objective is to maximize the profit of scheduled jobs. It is assumed that the set of jobs, the set of machines, and the state of the network is known in advance; hence the problem can be viewed as being *offline*. Each job has a certain demand and a time window for execution. The demand must arrive at the machine before execution can begin. Each machine also has a time window and can execute at most one job at a time. Finally, the demand must be routed through an undirected network such that two demands do not share an edge in the same time slot. If the demand exceeds the capacity of an edge, then the demand transmission may occupy the edge in several time slots.

The problem has application in distributed production systems where a set of jobs can be carried out at various plants. If the total job execution exceeds the total amount of available machines and if the transportation paths are limited, it is necessary to consider both problems simultaneously. A typical application is the steel industry where the production can be placed at various sites, but the transportation of iron ore and coal by e.g. train constitutes a substantial logistic problem.

The problem also has application in grid computing where jobs are to be executed at

various grid resources (machines) and where the grid resources are connected through an undirected optical network. A job cannot be executed before its input data has arrived at the executing grid resource and two data transmissions cannot use the same wavelength on the same fiber at the same time.

An example is The Large Hadron Collider (LHC) Physics Program by The European Organization for Nuclear Research (CERN). It is estimated that the LHC experiments generate 15 petabytes of data annually [41], thus the project utilizes grid computing not only for distributing the scientific work, but also for distributing data storage. The network connections for the grid computing system must support high bandwidth availability, like e.g. optical networks. For details on the Worldwide LHC Computing Grid, see their homepage [41]. See Bates [30] for a thorough description of optical networks and its applications.

The contribution of this paper is to model and solve JSCNR. We show that the problem is $\mathcal{NP}$-hard and propose several heuristic and exact solution methods. The exact solution method is based on applying Dantzig-Wolfe decomposition such that the master problem determines where and when jobs are executed and the pricing problem calculates routing schemes. The heuristics are based on combining methods for The Integrated Job Scheduling and Network Routing Problem (JSNR) and for The Constrained Network Routing Problem (CNR).

Two types of test instances are generated: a tandem topology with 10-200 jobs and 10-500 machines and a real-life network topology taken from the Nordic DataGrid Facility with 10-200 jobs and 14 machines. The suggested solution methods are evaluated on the test instances. The exact solution method performs better than applying CPLEX on a MIP formulation; however, it is unable to solve several of the considered test instances within a half hour time frame. The heuristics are capable of solving all instances within minutes. Best general heuristic performance is reached when using the First Come First Serve strategy for JSNR and a routing scheme which suggests two different paths for each demand for CNR. This setting gives an average solution value gap of 3%.

This paper is structured as follows. First JSCNR is defined in Section 6.2. Related work from the literature is also presented in this section along with notation and a mathematical model. In Section 6.3 heuristic methods are presented as combinations of methods for JSNR and for CNR. The heuristics are presented prior to the exact approach in Section 6.4, because they are used for solving the pricing problem and for finding a feasible start solution in the exact method. The suggested solution methods are computationally evaluated in Section 6.5 and final remarks are given in Section 6.6.

6.2 Problem definition

This section defines The Integrated Job Scheduling and Network Routing Problem (JSNR) and The Constrained Network Routing Problem (CNR). The problems are combined into The Integrated Job Scheduling and Constrained Network Routing Problem (JSCNR). For each problem an overview of work in the literature is given.

JSNR is closely related to JSCNR and only differs in the routing of job demands. Given is a set of jobs where each job has a certain demand, an estimated execution time, and a time window for execution. We also have a set of machines where each machine has an availability time window and can execute at most one job at time. Jobs must be assigned to machines and all job demand must arrive at the machine before execution can begin. The demand is routed through a capacitated network consisting of nodes and edges; the amount of demand on an edge in a time slot must not exceed the corresponding edge capacity. If the demand is larger than the edge capacity, then the demand can visit the edge in several time slots until all demand has been sent. The objective of the problem is to maximize the profit of executed jobs.

JSNR has application in production systems where transportation of goods from storage to production centers may constitute a logistical problem. The problem also has application in telecommunications; specifically in grid computing where jobs are executed on grid resources and where job input files must be sent to the executing grid resource through a (non-optical) network before execution can begin.

A simple version of JSNR consisting of sharing bandwidths in grid computing context was proved to be $\mathcal{NP}$-hard and greedy heuristics were presented by Marchal et al. [142].

An offline scheduler consisting of two steps was presented by Agarwal et al. [3]: first jobs were scheduled to grid resources such that the total penalty of delayed job executions was minimized, then the overall starting and end times of job schedules were determined.

Elghirani et al. [64] proposed a tabu search algorithm, which assigned jobs to a set of grid resources. The solution neighbourhood consisted of moving a scheduled job to another available grid resource and often used moves were penalized to avoid move cycles. When no improvement was reached in a certain time interval, the tabu list was cleared, a new random solution was found, and the tabu procedure started all over.

Varvaigos et al. [196] considered job routing and scheduling to support advance reservation. Advance reservation consists of reserving bandwidth and a grid resource for

later execution of a given job. Varvaigos et al. considered one job and data transmission at a time; hence their algorithm can be viewed as being an online algorithm.

JSNR was shown to be $\mathcal{NP}$-hard and solved to optimality by Gamst and Pisinger [86]. The solution method was based on Dantzig-Wolfe decomposition where the pricing problem assigned a single job to a single machine, the branching strategy added cuts to strengthen the formulation, and the master problem found an overall feasible solution. Results showed that their branch-and-cut-and-price algorithm outperformed both simpler exact algorithms and CPLEX. The algorithm was capable of solving instances with up to 1000 jobs and 1000 machines within minutes.

The telecommunication application of JSNR was solved heuristically by Gamst [79] using a number of greedy heuristics, a swap-based metaheuristic and the adaptive large neighbourhood metaheuristic. Results showed that though the metaheuristics found better solution values than the greedy methods, they also had relatively large running times.

CNR consists of sending demand through a network such that two routes never use the same edge at the same time. Given is a network consisting of nodes and capacitated edges. The network takes time into account, i.e., an edge can be visited at different time slots. Also given is a set of routing requests each consisting of a source, a destination, a routing time window, and an amount of demand. To satisfy a routing request, the demand must be sent from the source to the destination within the time window. If the amount of demand exceeds an edge capacity, then it takes several time slots to route the demand on that edge. Two routes cannot use the same edge at the same time.

CNR has application in the transportation sector. When routing trains through a railway infrastructure, two trains cannot use the same section of railway tracks at the same time. Also, the length of the train determines how long it takes to travel across a stretch of railway tracks. Each train has some starting and ending point and the goods on the train must arrive before a certain time.

CNR also has application in telecommunications where it corresponds to the $\mathcal{NP}$-hard static Routing and Wavelength Assignment Problem (RWA). The problem is to establish a number of connections (or light paths) in an optical network such that each connection travels from its source to its destination in a certain time window using one or more wavelengths. Two connections cannot use the same wavelength on the same fiber at the same time. The RWA is static since we have full knowledge on the problem instance in advance.

Most work on the RWA in the literature focuses on maximizing the number of established data connections. The underlying optical network is typically considered to be one of three topologies: wavelengths cannot be converted, see Zang et al. [206],

wavelengths can be converted in all nodes, see Ramamurthy and Mukherjee [163], and wavelengths can be converted in a subset of nodes, see Iness and Mukherjee [104]. The RWA was proved $\mathcal{NP}$-hard by Chlamtac et al. [47].

The RWA problem is typically solved using a heuristic decomposition which consists of a routing problem and a wavelength assignment problem. The routing problem suggests one or more paths for each data connection. The wavelength assignment problem finds an available wavelength and assigns it to one of the proposed paths for each data connection. An overview of heuristics from the literature for solving the decomposed RWA is presented by Zang et al. [206].

JSCNR consists of combining JSNR and CNR: jobs must be assigned to machines such that all job demand arrives at the machine before execution begins. The job demand is routed through an undirected network such that two routes never travel on the same edge at the same time. The network topology connects edges in such a way that the corresponding RWA does not support wavelength conversion. Jobs must be assigned to machines for execution such that the total profit of executed jobs is maximized. JSCNR is $\mathcal{NP}$-hard as it contains both the $\mathcal{NP}$-hard JSNR and the $\mathcal{NP}$-hard CNR as special cases.

6.2.1 Mathematical formulation

Notation from applying JSCNR in a telecommunications context is used in the following formalization. This means that we consider the problem of assigning jobs to grid resources where job data must be routed through an optical network. The optical network is dedicated to the job scheduling process, hence paths between all terminal nodes are known in advance.

The set of jobs is denoted J, the set of resources is R, the set of edges is E and the set of time stamps is T. Note that time is discrete, i.e., is given in time stamps $t \in T$.

The set of wavelengths on edge $(i, k) \in E$ is denoted λ_{ik} and the set of all wavelengths is denoted λ. For a wavelength $l \in \lambda$ let E_l denote the set of edges which are capable of carrying data on wavelength l. All wavelengths on all edges have the same bandwidth capacity d.

Let t_{λ^+} denote the time it takes to establish a new wavelength on an edge and let t_{λ^-} denote the time it takes to release a wavelength on an edge. The reason for introducing these time buffers is to make the solution more robust: if a data transmission is delayed, then it will not be interfered by a new transmission if the delay is less than t_{λ^-}. Furthermore, a data transmission does not start until t_{λ^+} time after the wavelength is assigned

thus leaving even further room for the previous transmission to finish. Introducing these extra time buffers has a drawback; the extra time buffers may prevent some jobs from being executed. When solving the problem, the grid administrator should thus experiment with the size of the time buffers in order to reach an appropriate trade-off between robustness and job execution.

Each job $j \in J$ is assigned a time window $[a_j, b_j]$, the estimated computation time Q_j, the total size of the job data S_j, the amount of data p_j^r placed on each resource $r \in R$, and a profit $c_j \in \mathbb{R}^+$ for execution.

Each resource $r \in R$ is assigned an availability start time a_r and end time b_r. To simplify notation, the time window $[a_{ik}, b_{ik}]$ is introduced, where $a_{ik} = \max\{a_i, a_k\}$ and $b_{ik} = \min\{b_i, b_k\}$ for $i, k \in R \cup J$. For further notational convenience, two sets are introduced: J_t and R_t. The set J_t consists of jobs j with $a_j \leq t \leq b_j$. Similarly, the set R_t consists of resources r with $a_r \leq t \leq b_r$.

Now, the mathematical model includes two types of variables $x_j^{tr} \in \{0,1\}$ and $x_{ikl}^{tj} \in \{0,1\}$. If $x_j^{tr} = 1$ then job $j \in J$ is executed on resource $r \in R$ with execution beginning at time $t \in T$. If $x_j^{tr} = 0$ then the job is not executed on the resource with this beginning time. If $x_{irl}^{tjk} = 1$ then edge $(i,r) \in E$ is carrying data original stored on resource $k \in R$ on wavelength $l \in \lambda_{ir}$ at time $t \in T$ for job $j \in J$. Otherwise, $x_{irl}^{tjk} = 0$. JSCNR is formulated as:

$$\max \quad \sum_{r \in R} \sum_{j \in J} \sum_{t=a_{rj}}^{b_{rj}-Q_j} c_j x_j^{rt} \tag{6.1}$$

$$\text{s.t.} \quad \sum_{r \in R} \sum_{t=a_{rj}}^{b_{rj}-Q_j} x_j^{rt} \leq 1 \qquad \forall j \in J \tag{6.2}$$

$$\sum_{i \in R} \sum_{t'=a_{ri}}^{t-1} \sum_{l \in \lambda_{ri}} x_{ril}^{t'jr} \geq \left\lceil \frac{p_r^j}{d} \right\rceil \sum_{i \in R \setminus \{r\}} x_j^{it} \qquad \begin{array}{l} \forall r \in R, \forall j \in J, \\ \forall t \in [a_{rj}, b_{rj} - Q_j] \end{array} \tag{6.3}$$

$$\sum_{\substack{i \in R \\ i \neq r}} \sum_{\substack{k \in R \\ k \neq r}} \sum_{t'=a_{ir}}^{t-1} \sum_{l \in \lambda_{ir}} x_{irl}^{t'jk} \geq \left\lceil \frac{S_j - p_r^j}{d} \right\rceil x_j^{rt} \qquad \begin{array}{l} \forall r \in R, \forall j \in J, \\ \forall t \in [a_{rj}, b_{rj} - Q_j] \end{array} \tag{6.4}$$

$$\sum_{t'=t+1}^{b_{rj}-Q_j} x_j^{rt'} = 0 \Rightarrow \sum_{\substack{i \in R_t \\ (i,r) \in E_l}} x_{irl}^{tjk} - \sum_{\substack{i \in R_t \\ (r,i) \in E_l}} x_{ril}^{(t+1)jk} = 0 \qquad \begin{array}{l} \forall k, r \in R : p_k^j > 0, \\ \forall j \in J, \forall l \in \lambda, \\ \forall t \in [a_r, b_r] \end{array} \tag{6.5}$$

$$\sum_{k \in R : p_k^j > 0} \sum_{j \in J_t} \sum_{t'=t}^{t+t_{\lambda^-}+t_{\lambda^+}} (x_{irl}^{t'jk} + x_{ril}^{t'jk}) \leq 1 \qquad \begin{array}{l} \forall (i,r) \in E, \forall l \in \lambda_{ir} \\ \forall t \in [a_{ir}, b_{ir}] \end{array} \tag{6.6}$$

$$\sum_{j' \in J \setminus j} \sum_{t'=t}^{\substack{\min\{t+Q_j,\\ b_{j'r}-Q_{j'}\}}} x_{j'}^{rt'} + Q_j x_j^{rt} \leq Q_j \qquad \begin{array}{l}\forall j \in J,\ r \in R,\\ t \in [a_{jr}, b_{jr} - Q_j]\end{array} \tag{6.7}$$

$$x_j^{rt} \in \{0,1\} \qquad \begin{array}{l}\forall j \in J, \forall r \in R,\\ \forall t \in [a_{rj}, b_{rj}]\end{array} \tag{6.8}$$

$$x_{irl}^{tjk} \in \{0,1\} \qquad \begin{array}{l}\forall j \in J, \forall k \in R,\\ \forall (i,r) \in E,\\ \forall l \in \lambda_{ir}, \forall t \in [a_{rj}, b_{rj}]\end{array} \tag{6.9}$$

The objective (6.1) maximizes the profit of executed jobs. The first constraint (6.2) says that each job can be executed at most once. If a job is executed on some resource $i \in R$ then data from all other resources $r \in R$ must be sent out on the network (6.3). Constraint (6.4) says that if a job is executed at resource $r \in R$ then all data must arrive before execution time. Flow conservation is ensured in (6.5). Data arriving at some node at time t must leave the node again at time $t+1$ unless the job is executed at this node. Constraint (6.6) forbids several paths from using the same wavelength on the same edge at the same time. Finally, the last constraint (6.7) says that a resource can execute at most one job at a time. Bounds ensure that variables take on feasible values.

6.3 Greedy heuristic solution approach

In this paper, we consider a heuristic approach for The Integrated Job Scheduling and Constrained Network Routing Problem (JSCNR), which combines greedy heuristics for The Integrated Job Scheduling and Network Routing Problem (JSNR) and for The Constrained Network Routing Problem (CNR). JSNR was solved heuristically by Gamst [79]. The data transmission part of the heuristics, however, must be replaced by algorithms for the CNR. The latter has application in telecommunications as the Routing and Wavelength Assignment Problem (RWA) for which many solution methods are presented in the literature, see e.g. the survey of Zang et al. [206].

Let us first consider heuristics for JSNR in the literature (see Gamst [79] or Sørensen [181] for more details):

- First Come First Serve. The first job on queue is assigned to the resource at which execution finishes first. Let $|\mathrm{T}_{\mathrm{data}}|$ denote the running time of transmitting data. The theoretical running time for the First Come First Serve heuristic is $\mathcal{O}(|J||R||\mathrm{T}_{\mathrm{data}}|)$, since the heuristic in worst case attempts to assign each job to all resources.

- Best First. The job with highest profit is assigned to the resource at which job execution finishes first. The running time is $\mathcal{O}(|J|\log|J|+|J||R||\mathrm{T}_{\mathrm{data}}|)$ where $|\mathrm{T}_{\mathrm{data}}|$ is the running time for the data transmission problem, since jobs first are sorted according to profit and then the heuristic in worst case tries to assign each job to all resources.
- First Fit. For each resource, the job with earliest execution finish time is executed. If a draw between several jobs are reached then the job with highest profit is selected. The theoretical running time is $\mathcal{O}(|R||J|^2|\mathrm{T}_{\mathrm{data}}|)$ where $|\mathrm{T}_{\mathrm{data}}|$ is the running time for the data transmission problem, because for each resource the heuristic assigns all pairs of jobs in order to compare the execution finish times.
- Random Fit. Randomly selected jobs are assigned to each resource. The running time is $\mathcal{O}(|J||R||\mathrm{T}_{\mathrm{data}}|)$ where $|\mathrm{T}_{\mathrm{data}}|$ is the data transmission running time, because in worst case the heuristic tries to assign each job to all resources.

These four heuristics need to know how long it takes to transmit job data to a resource in order to determine execution start and end times. The time it takes to transmit job data is found by solving the CNR problem heuristically.

CNR is solved as the RWA and we propose using a subset of the heuristics for the RWA in the literature. When solving the RWA as part of the JSCNR, the RWA may be solved a large number of times. Thus if the heuristic for the RWA has high complexity, then the overall solution procedure will suffer. The selected heuristics have relatively small running times and all divide the RWA into a routing problem and a wavelength assignment problem. The selected heuristics for the routing problem are:

- Fixed-Alternate Routing. Several paths are found for each data connection request; see Banerjee et al. or Birman and Kershenbaum [22, 38]. The heuristic corresponds to the k-shortest path problem, when the number of generated paths for the data connection corresponds to k. Thus the theoretical running time for establishing a single data connection equals that of the k-shortest path problem; $\mathcal{O}(|E|+|V|\log|V|+k)$ where $|V|$ is the number of nodes in the network, see Eppstein [66].
- Adaptive Routing. This method runs a shortest path algorithm on the graph where edge costs are based on previously chosen routes; see Zang et al. [206]. The theoretical running time for establishing a single data connection corresponds to the running time for a shortest path algorithm, e.g. $\mathcal{O}((|E|+|V|)\log|V|)$ which is the running time of Dijkstra's algorithm using a binary heap, see Cormen et al. [52].

The selected heuristics for wavelength assignment are:

- First Fit. The first available wavelength is assigned to the current data connection request; see Birman and Kershenbaum or Kovacevic and Acampora [38, 130]. The running time for assigning a wavelength to a single data connection is $\mathcal{O}(|\lambda||E|)$ where $|\lambda|$ is the number of wavelengths, as the heuristic in worst case investigates the availability of each wavelength on all edges.

- Most Used. Among the available wavelengths for a data connection request, the wavelength which so far has been used the most is assigned to the data connection request, see Subrarnaniam and Barry [183]. The theoretical running time is $\mathcal{O}(|\lambda| \log |\lambda| + |\lambda||E|)$, because first the availability of all wavelengths on all edges is found, then the wavelengths are sorted according to usage, and finally the heuristic investigates the availability of each wavelength from the sorted list on all edges.

- Random Assignment. An available wavelength is randomly selected and assigned to the current data connection request. Running time is $\mathcal{O}(|\lambda||E|)$ where $|\lambda|$ is the number of wavelengths, because in worst case the heuristic investigates the availability of each wavelength on all edges.

6.3.1 Heuristics for JSCNR

Combining the heuristics from the previous section results in heuristics for JSCNR. The heuristics are displayed in Table 6.1 along with their theoretical running times. The upper table uses Fixed-Alternate routing, the lower Adaptive routing. The first row in each part consists of the name of the JSNR heuristic. The remaining three rows in each part consist of the name of the wavelength assignment heuristics and the corresponding theoretical running time for combining the JSNR and CNR heuristics. The theoretical running times in the Table are used for comparison with practical running times when computationally evaluating the heuristics in Section 6.5.

6.4 Exact solution approach

The exact solution approach is based on Dantzig-Wolfe decomposing The Integrated Job Scheduling and Constrained Network Routing Problem (JSCNR) such that the master problem decides where and when to execute jobs according to data transmission. The pricing problem decides when to send all data for each job according to the reduced costs. Recall the mathematical formulation (6.1)-(6.9). The master problem includes constraints (6.2), (6.6), and (6.7) and the pricing problem takes care of the remaining constraints along with (6.6).

FCFS	Fixed-alternate
First fit	$\mathcal{O}(\|J\|\|R\|(\|V\|\log\|V\|+k+\|\lambda\|\|E\|))$
Most used	$\mathcal{O}(\|J\|\|R\|(\|V\|\log\|V\|+k+\|\lambda\|(\log\|\lambda\|+\|E\|)))$
Random	$\mathcal{O}(\|J\|\|R\|\|(\|V\|\log\|V\|+k+\|\lambda\|\|E\|))$
Best first	
First fit	$\mathcal{O}(\|J\|(\log\|J\|+\|R\|\|(\|V\|\log\|V\|+k+\|\lambda\|\|E\|)))$
Most used	$\mathcal{O}(\|J\|(\log\|J\|+\|R\|(\|V\|\log\|V\|+k+\|\lambda\|(\log\|\lambda\|+\|E\|))))$
Random	$\mathcal{O}(\|J\|(\log\|J\|+\|R\|(\|V\|\log\|V\|+k)+\|\lambda\|\|E\|))$
First fit	
First fit	$\mathcal{O}(\|J\|^2\|R\|(\|V\|\log\|V\|+k+\|\lambda\|\|E\|))$
Most used	$\mathcal{O}(\|J\|^2\|R\|(\|V\|\log\|V\|+k+\|\lambda\|(\log\|\lambda\|+\|E\|)))$
Random	$\mathcal{O}(\|J\|^2\|R\|(\|V\|\log\|V\|+k+\|\lambda\|\|E\|))$
Random fit	
First fit	$\mathcal{O}(\|J\|\|R\|(\|V\|\log\|V\|+\|\lambda\|\|E\|))$
Most used	$\mathcal{O}(\|J\|\|R\|(\|V\|\log\|V\|+\|\lambda\|(\log\|\lambda\|+\|E\|)))$
Random	$\mathcal{O}(\|J\|\|R\|(\|V\|\log\|V\|+\|\lambda\|\|E\|))$

FCFS	Adaptive
First fit	$\mathcal{O}(\|J\|\|R\|((\|E\|+\|V\|)\log\|V\|+\|\lambda\|\|E\|))$
Most used	$\mathcal{O}(\|J\|\|R\|((\|E\|+\|V\|)\log\|V\|+\|\lambda\|(\log\|\lambda\|+\|E\|)))$
Random	$\mathcal{O}(\|J\|\|R\|((\|E\|+\|V\|)\log\|V\|+\|\lambda\|\|E\|))$
Best first	
First fit	$\mathcal{O}(\|J\|(\log\|J\|+\|R\|((\|E\|+\|V\|)\log\|V\|+\|\lambda\|\|E\|)))$
Most used	$\mathcal{O}(\|J\|(\log\|J\|+\|R\|((\|E\|+\|V\|)\log\|V\|+\|\lambda\|(\log\|\lambda\|+\|E\|))))$
Random	$\mathcal{O}(\|J\|(\log\|J\|+\|R\|((\|E\|+\|V\|)\log\|V\|+\|\lambda\|\|E\|)))$
First fit	
First fit	$\mathcal{O}(\|J\|^2\|R\|((\|E\|+\|V\|)\log\|V\|+\|\lambda\|\|E\|))$
Most used	$\mathcal{O}(\|J\|^2\|R\|((\|E\|+\|V\|)\log\|V\|+\|\lambda\|(\log\|\lambda\|+\|E\|)))$
Random	$\mathcal{O}(\|J\|^2\|R\|((\|E\|+\|V\|)\log\|V\|+\|\lambda\|\|E\|))$
Random fit	
First fit	$\mathcal{O}(\|J\|\|R\|((\|E\|+\|V\|)\log\|V\|+\|\lambda\|\|E\|))$
Most used	$\mathcal{O}(\|J\|\|R\|((\|E\|+\|V\|)\log\|V\|+\|\lambda\|(\log\|\lambda\|+\|E\|)))$
Random	$\mathcal{O}(\|J\|\|R\|((\|E\|+\|V\|)\log\|V\|+\|\lambda\|\|E\|))$

Table 6.1: Theoretical running times for all heuristics. The running times consist of multiplying the running time for the grid heuristic with the sum of the running times of the routing and the wavelength assignment heuristics.

Let the decision variable $y_p^{jrt} \in \{0,1\}$ indicate if job j is executed on resource r at time t where job data is sent according to p. The pricing problem generates ways of sending data $p \in P$ for a given job, resource and execution time according to the reduced cost of the current solution. The master problem is:

$$\max \quad \sum_{j\in J}\sum_{r\in R}\sum_{t=a_{rj}}^{b_{rj}-Q_j}\sum_{p\in P} c_j y_p^{jrt} \tag{6.10}$$

$$\text{s. t.} \quad \sum_{r\in R}\sum_{t=a_{rj}}^{b_{rj}-Q_j}\sum_{p\in P} y_p^{jrt} \leq 1 \qquad \forall j \in J \tag{6.11}$$

$$\sum_{j\in J_t}\sum_{u\in R_t}\sum_{t'=t+t_{\lambda-}+t_{\lambda+}}^{t+t_{\lambda-}+t_{\lambda+}}\sum_{p\in P}(\delta_p^{irl} y_p^{jut'} + \delta_p^{ril} y_p^{jut'}) \leq 1 \qquad \begin{array}{l}\forall (i,r) \in E, \forall l \in \lambda_{ir},\\ \forall t \in [a_{ir}, b_{ir}]\end{array} \tag{6.12}$$

$$\sum_{j'\in J\setminus j}\sum_{t'=t}^{\min\{t+Q_j,\, b_{j'r}-Q'_j\}}\sum_{p\in P} y_p^{j'rt'} + Q_j \sum_{p\in P} y_p^{jrt} \leq Q_j \qquad \begin{array}{l}\forall j \in J,\ \forall r \in R,\\ t \in [a_{jr}, b_{jr} - Q_j]\end{array} \tag{6.13}$$

$$y_p^{jrt} \in \{0,1\} \qquad \begin{array}{l}\forall j \in J,\ r \in R,\\ t \in [a_{jr}, b_{jr} - Q_j]\end{array} \tag{6.14}$$

The objective (6.10) maximizes the profit of executed jobs. The first constraint (6.11) ensures that a job can be executed at most once and the second constraint (6.12) ensures that each wavelength on each edge is visited by at most one data connection. Finally, constraint (6.13) says that a resource can execute at most one job at a time and the bound (6.14) forces variables to take on feasible values.

6.4.1 Pricing problem

The dual variables of the master problem are $\pi_j \geq 0$, $\omega_{irlt} \geq 0$ and $\rho_{jrt} \geq 0$ for constraints (6.11), (6.12), and (6.13), respectively. The reduced cost for a given job j, resource r and execution time t is:

$$c_j - \pi_j - Q_j\rho_{jrt} - \sum_{j'\in J\setminus\{j\}}\sum_{\substack{t'=\max\\ \{t-Q_{j'}+1, a_{j'r}\}}}^{\min\{t, b_{rj'}-Q_{j'}\}} \rho_{j'rt'} > \sum_{(i,r)\in E}\sum_{l\in L}\sum_{t'=a_{ir}}^{b_{ir}} \left(\omega_{irl}^{t'} + \omega_{ril}^{t'}\right) \tag{6.15}$$

When solving the pricing problem for a given job j, resource r, and execution time t we wish to minimize the right hand side of the reduced cost, because the value of the left hand side is already known. Hence the pricing problem is to find a way of sending all job data for job j to resource r in time for job execution at time t such that the right hand side of (6.15) is minimized.

The decision variable $y_{iul}^{t'k} \in \{0,1\}$ is introduced to indicate data transmission in the pricing problem. Let $y_{iul}^{t'k}$ denote whether or not data stored on resource $k \in R$ is travelling on edge $(i,u) \in E$, using wavelength $l \in \lambda$ at time $t' \in [a_{ir}, t[$. The pricing problem is:

$$\min \quad \sum_{k\in R}\sum_{(i,u)\in E}\sum_{l\in L}\sum_{t'\in[a_{ir},t[}\left(\omega_{iul}^{t'}y_{iul}^{t'k}+\omega_{uil}^{t'}y_{uil}^{t'k}\right) \tag{6.16}$$

$$\text{s. t.} \quad \sum_{i\in R}\sum_{t'=a_{ki}}^{t-1}\sum_{l\in\lambda_{ki}} y_{kil}^{t'k} \geq \left\lceil \frac{p_k^j}{d} \right\rceil \qquad \forall k \in R \tag{6.17}$$

$$\sum_{\substack{k\in R\\ p_k^j>0}}\sum_{(i,r)\in E}\sum_{t'=a_{ir}}^{t-1}\sum_{l\in\lambda_{ir}} y_{irl}^{t'k} \geq \left\lceil \frac{S_j - p_r^j}{d} \right\rceil \tag{6.18}$$

$$\sum_{\substack{i\in R_t\\ (i,u)\in E_l,\\ u\neq r}} y_{iul}^{t'k} - \sum_{\substack{i\in R_t\\ (u,i)\in E_l,\\ u\neq r}} y_{uil}^{(t'+1)k} = 0 \qquad \forall u,k \in R\backslash\{r\} : p_k^j > 0 \tag{6.19}$$

$$\forall l \in \lambda, \forall t' \in [a_u, t[$$

$$\sum_{j\in J_t}\sum_{\substack{k\in R\\ p_k^j>0}}\sum_{u\in R_t}\sum_{t''=t'}^{t'+t_{\lambda^-}+t_{\lambda^+}} (y_{iul}^{t''k}+y_{uil}^{t''k}) \leq 1 \qquad \begin{array}{l}\forall (i,u)\in E, \forall l \in \lambda_{ir},\\ \forall t' \in [a_{iu}, t[\end{array} \tag{6.20}$$

$$y_{irl}^{t'k} \in \{0,1\} \qquad \begin{array}{l}\forall k \in R : p_k^j > 0,\\ \forall (i,r)\in E, \forall l \in \lambda,\\ \forall t' \in [a_{ir}, t[\end{array} \tag{6.21}$$

The objective (6.16) minimizes the right hand side of (6.15). The first constraint (6.17) says that all job data must be sent from each data source. The next constraint (6.18) makes sure that all job data arrives at the executing resource r before job execution time t. Constraint (6.19) ensures flow conservation. Finally constraint (6.20) says that no more than one data connection can use a wavelength on an edge at a time and the bound (6.21) forces variables to take on feasible values.

The pricing problem is the Routing and Wavelength Assignment Problem (RWA) over time and is $\mathcal{NP}$-hard. Hence we try to generate columns heuristically and only solve the pricing problem to optimality when no heuristic columns with positive reduced cost can be found. The proposed greedy heuristics for the RWA in Section 6.3 are applied on the pricing problem to generate columns heuristically. The heuristics are modified slightly: when they can choose between several paths or wavelengths, then the cheapest option according to (6.16) is selected.

The exact solution approach is based on solving the mathematical formulation for the RWA problem over time. Recall that all paths between all pairs of resources are known in advance. In the mathematical formulation we generate a column for each path at

each possible start time using each wavelength. The exact solution approach is solved for a given job j, an executing resource r, and an execution time t. Let $\mathcal{P}$ denote the set of columns. The variable $y_p \in \{0,1\}$ indicates whether or not column $p \in \mathcal{P}$ is included in the current solution. Three constants are introduced: $\delta_p^{iklt'}$ denotes whether or not column p uses wavelength $l \in \lambda$ on edge $(i,k) \in E$ at time $t' \in [a_{ikj}, t[$, δ_p^k denotes whether or not column p routes data stored on resource k, and c_p denotes the reduced cost for column p. The model is:

$$\min \quad \sum_{p\in\mathcal{P}} \sum_{(i,k)\in E} \sum_{l\in\lambda} \sum_{t'\in[a_{ikj},t[} \delta_p^{iklt'} c_p y_p \tag{6.22}$$

$$\text{s. t.} \quad \sum_{p\in\mathcal{P}} \delta_p^{iklt'} y_p + \delta_p^{kilt'} y_p \leq 1 \qquad \forall (i,k) \in E, \forall l \in \lambda, \forall t' \in [a_{ikj}, t[\tag{6.23}$$

$$\sum_{p\in\mathcal{P}} \delta_p^k y_p = 1 \qquad \forall k \in R : p_k^j > 0 \tag{6.24}$$

$$y_p \in \{0,1\} \qquad \forall p \in \mathcal{P} \tag{6.25}$$

The objective function (6.22) minimizes the reduced cost. The first constraint (6.23) says that each wavelength on each edge can be used at most once and constraint (6.24) ensures that all data connections are established exactly once.

The number of columns in (6.22)-(6.25) is polynomial in the input size: the path between two terminal nodes is known in advance. We must decide when to travel on the path, thus we generate a path variable for each path at each available travel time and for each wavelength. Let $\mathcal{O}(|K|)$ be the number of data connections, $\mathcal{O}(|T|)$ be the number of available travel times, and $\mathcal{O}(|\lambda|)$ be the number of wavelengths; the number of variables is $\mathcal{O}(|\lambda||T||K|)$.

6.4.2 Branching strategy

Branching ensures that variables in the LP-relaxed master problem eventually take on binary values. To determine the branching strategy we investigate when variable values may become fractional:

1. A job is only partially executed
2. A job is executed on the same resource but at different times
3. A job is executed on different resources
4. A job is executed on a given resource at a given time using routing times which differ in the latest data arrival time
5. A job is executed on a given resource at a given time using routing schemes which differ in the used wavelengths

In the first case we generate two branching children in each of which we add the constraint:

$$\sum_{p \in P} \delta_p^j y_p = 0 \quad \text{vs.} \quad \sum_{p \in P} \delta_p^j y_p = 1 \tag{6.26}$$

which ensures that job j is either not executed or it is fully executed. The branching constraint adds a dual variable ω_j, which the pricing problem must handle. Because the pricing problem is solved for each job, the extra dual variable can be added to the left hand side of (6.15) and does not interfere with the pricing problem.

The second case is handled by finding a time stamp lying between the current execution times. Two branching children are generated: in the first child the job must be executed no later than the time stamp and in the second child the job must be executed after the time stamp. In each child, columns with illegal execution times are set to zero. The pricing problem is altered slightly into setting bounds on execution times and not allowing data to arrive later than the latest execution start time.

The third case is handled by choosing a resource on which the job is partially executed. Two branching children are generated: in the first child the job must be executed on the resource, and in the second child the job cannot be executed on the resource. In each branching child, columns using an illegal executing resource are set to zero. The pricing problem is modified slightly into either forcing execution on a certain resource or to not allowing execution on illegal resources.

In the fourth case the data transmission times and possibly the used wavelengths differ. The case is handled by finding a time stamp for routing. Two branching children are generated: in the first child all data must arrive before the time stamp and in the second child some data must arrive after the time stamp. In each child the variables with illegal routing times are set to zero. The pricing problem is modified into not allowing routing at illegal times by excluding predefined columns using illegal routing times.

In the fifth case the execution and data transmission times are equal for all non-zero variables. Only the used wavelengths differ. The case is handled by choosing a wavelength for a data transmission path. The chosen wavelength must be used by at least one of the fractional variables in the current solution. Two branching children are generated: in the first child the chosen wavelength must be used on the chosen path, thus all variables which use different wavelengths are set to zero. In the second branching child the chosen wavelength cannot be used on the chosen path, thus all variables which use the chosen wavelength are set to zero. The pricing problem is modified into including or excluding columns using the chosen wavelength on the chosen path, respectively.

6.4.3 Start solution

The master problem must initially hold one or more columns before values for dual variables can be found for the pricing problem. To reach a start solution we can apply the greedy heuristics from Section 6.3 on the problem instance. The heuristics, however, do not guarantee a feasible solution even if one exists. In this case, an exact solution approach must try to assign a job to a resource. We choose to run a modified version of the exact solution approach for the pricing problem; instead of minimizing the reduced cost, the exact approach only decides whether or not it is possible to assign a given job to a given resource.

6.4.4 Reducing the number of constraints

The master problem consists of a large number of constraints, especially as time window sizes increase. Some instances may not utilize large parts of the time windows; hence it would be beneficial to leave out constraints for unused time stamps. Through preliminary results we have noted a significant improvement of approximately 35% on time usage when including all constraints of type (6.11) and only violated constraints of type (6.12) - (6.14). Separation routines for identifying violated constraints consider all non-negative variables for all possible constraints and have polynomial running time in the input size.

Including only violated constraints does not impose any changes on neither the pricing problem nor the branching strategies. When calculating the reduced costs, only dual variables for constraints included in the master problem are considered.

6.4.5 Reducing the number of iterations

Preliminary results showed that the branch-and-cut-and-price algorithm runs through a relatively large number of iterations before finding a lower bound in a search tree node. The reason for this may be that the dual variables take on inappropriate values, hence the algorithm prices in many unused columns before finally converging toward the lower bound. A way to avoid this is to apply a method for stabilizing the values of dual variables. Several stabilization methods are presented in the literature. They typically consist of setting bounds on how much the values of the dual variables may change from one iteration to the next. The bounds may be in the form of boxes for each dual variable, see Rousseau et al. [170] or by adding a punishment in the objective function for the distance between the former and the current value of each dual variable, see DuMerle et al. [61]. Rousseau et al. [170] suggest an interior-point sta-

bilization method where the values of dual variables are set to a linear combination of extreme points in the dual solution space. The stabilization method can easily be applied to the master problem by changing the bounds on constraints and variables whose corresponding dual variables and constraints are not tight. For details, see Rousseau et al. [170] who show how to apply the stabilization method on the Set Cover problem. We have applied the interior-point stabilization method and preliminary results show that the method decreases time usage with up to 67%.

6.5 Computational experiments

The proposed solution methods are tested. In this section we first introduce the generated problem instances, then a computational evaluation of the proposed exact method and heuristics for JSCNR is presented.

6.5.1 Test instances

Two types of problem instances are generated. Both instance types arise in telecommunications and are denoted the NDGF and the Tandem instances, respectively.

NDGF
A set of instances is based on the network topology of the Nordic DataGrid Facility (NDGF), which consists of a grid computing system in the Nordic countries. Current projects on the NDGF include handling data from the Large Hadron Collider (LHC) by the European Organization for Nuclear Research (CERN), see CERN's homepage for more information [41]. The NDGF network topology was presented by Grønager [96] and consists of 14 nodes, which are connected in a sparse graph. An illustration can be seen in Figure 6.1. All data arrives from Europe to a grid resource in Denmark, which thus works as job data storage for all jobs. In three of the Nordic countries, grid resources are connected through a network node. These are marked as squares in Figure 6.1. The grid storage in Denmark and all network hubs are available at all times.

Tandem
A set of instances based on a *tandem* topology is generated. An example of a tandem network is given in Figure 6.2. All nodes but two are connected with exactly two other nodes. The two nodes in each end of the network are only connected with one other node. Hence, the number of edges in the test instances is always $|E| = 2(|V| - 1)$.

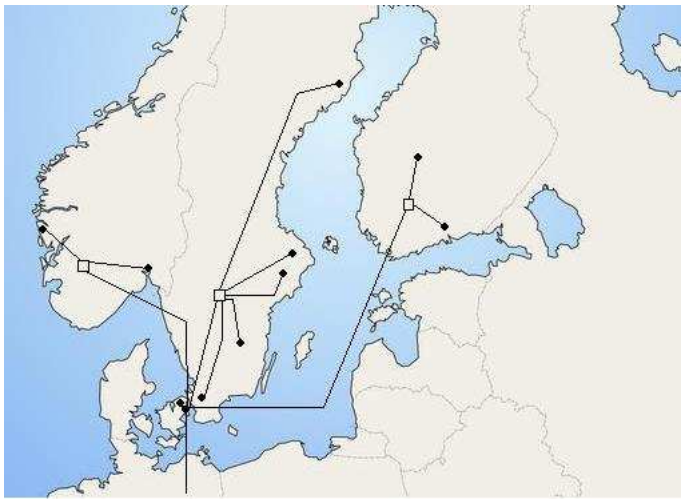


Figure 6.1: An illustration of the NDGF network. Resources are marked as filled circles, while the squares indicate nodes unable to execute jobs.

This set of instances is introduced in order to test how larger networks are handled. The number of edges and nodes thus vary from instance to instance.

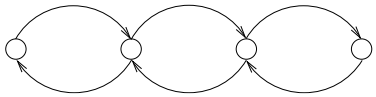


Figure 6.2: An example of a tandem network. Every node is only connected to its neighboring nodes.

Grid activity
The number of jobs, the number of wavelengths, and the amount of available bandwidth per wavelength vary from instance to instance. The size and distribution of job input files, the execution time, and the time window for each job are randomly generated. Similarly, the resource time windows are also randomly generated.

6.5.2 Results

The solution methods have been implemented in C++ and tested on a 2.66 GHz Intel Xeon machine with 16 GB RAM. Note that CPU times in the following stem from using one core. All test runs are given an upper time bound on 1800 seconds. First we

analyze the exact solution methods, i.e., we apply CPLEX on the mathematical formulation (6.1)-(6.9) and compare with the branch-and-cut-and-price algorithm. Then the heuristics are considered.

CPLEX
JSCNR can be solved to optimality by generating the edge based model (6.1)-(6.9) for each instance and then using CPLEX to solve the model. Test results are seen in Table 6.2. The results show that CPLEX runs out of memory or time even for the smaller instances. This motivates the need for a more sophisticated exact solution method.

Exact
Solving JSCNR with CPLEX was unsuccessful; hence we implemented the more sophisticated branch-and-cut-and-price (BCP) algorithm from Section 6.4. Test results are seen in Table 6.3-6.5.

The results show that the sophisticated BCP algorithm is also unable to solve several instances within the 1800 seconds. It does, though, generally perform better than when using CPLEX, both with respect to time usage and to the number of solved instances. An in-depth analysis of the test results for the BCP algorithm has shown that the bottleneck is solving the pricing problem to optimality. Recall that the pricing problem is the RWA over time, which is $\mathcal{NP}$-hard. The BCP algorithm solves the pricing problem heuristically until no columns are found at which point the pricing problem is solved to optimality. Separating cuts, solving the master problem, generating branching candidates, and branching take little time and the search tree is always small.

When comparing results for the tandem instances with results for the NDGF instances, we see that the BCP algorithm has equal difficulty with solving both instance types. The topology of the NDGF instances can be viewed as a combination of a tree and a star structure and not many paths share edges. Conversely, paths share many edges in the tandem instances. The reason why the BCP algorithm finds both instances hard to solve is probably that both the scheduling and the routing problem are $\mathcal{NP}$-hard, hence if either constitutes a bottleneck then the overall problem is very difficult to solve.

Heuristics
Solving JSCNR to optimality is very difficult even for smaller instances. Hence heuristics for the problem may be useful when larger instances are to be solved. The proposed heuristics in Section 6.3 have been implemented. First they are compared with the exact solution approach and then they are compared with each other. See the tables at `http://www.diku.dk/~gamst/tables.pdf` for detailed test results.

Jobs	Res.	BW	Results $\lambda = 5$	Time	Results $\lambda = 10$	Time	Results $\lambda = 20$	Time
10	10	10	12	0.18	12	0.30	12	0.65
10	10	25	12	0.17	12	0.33	12	0.66
10	20	10	2	0.85	2	1.77	2	3.57
10	20	25	2	0.88	2	1.70	2	3.50
10	50	10	69	31.14	69	65.98	69	155.35
10	50	25	69	31.29	69	66.19	69	157.83
10	100	10	7	177.31	–	oom	–	oom
10	100	25	7	179.32	–	oom	–	oom
20	10	10	26	1.12	26	0.94	26	1.95
20	10	25	26	0.50	26	0.92	26	1.91
20	20	10	63	4.65	63	9.09	63	18.40
20	20	25	63	4.61	63	8.93	63	18.30
20	50	10	159	108.75	–	oom	–	oom
20	50	25	159	108.10	–	oom	–	oom
50	10	10	80	1.20	80	1.96	80	3.80
50	10	25	80	1.09	80	1.89	80	3.67
50	20	10	153	6.50	153	17.85	153	36.28
50	20	25	153	9.64	153	17.90	153	37.98
50	50	10	–	oom	–	oom	–	oom
50	50	25	–	oom	–	oom	–	oom
100	10	10	147	6.94	147	8.99	147	14.84
100	10	25	147	4.94	147	8.26	147	15.29
100	20	10	285	36.85	285	139.70	285	151.07
100	20	25	285	40.36	285	64.20	285	150.36
100	50	10	–	oom	–	oom	–	oom
100	50	25	–	oom	–	oom	–	oom
200	10	10	164	7.66	164	8.86	164	14.77
200	10	25	164	6.54	164	8.77	164	15.14
200	20	10	316	71.97	316	122.54	316	223.24
200	20	25	316	82.61	316	116.43	316	218.04
200	50	10	–	oom	–	oom	–	oom
200	50	25	–	oom	–	oom	–	oom
10	14	10	41	0.67	41	1.34	41	2.53
10	14	25	41	0.66	41	1.28	41	2.53
20	14	10	116	1.57	116	3.47	116	5.24
20	14	25	116	1.54	116	3.35	116	5.39
50	14	10	266*	1866.91*	272*	1815.78*	273*	1922.77*
50	14	25	266*	1810.28*	272*	1810.80*	273*	1808.40*

Table 6.2: Test results for the CPLEX approach. The first three columns hold information on the number of jobs, resources and the amount of bandwidth. Instances with 14 resources are of type NDGF; all other instances are of the Tandem type. Then follows two columns for three different wavelength settings, i.e., number of wavelengths per fiber: $\lambda = 5, 10,$ and 20. The two columns for each setting give the result value and the running time in seconds. An entry with 'oom' means that the instance could not be solved due to memory problems (Out Of Memory). An entry with '*' indicates that the instance could not be solved within 1800 seconds and thus ran out of time. The best feasible solution is then given.

Jobs	**Res.**	**BW**	**Results** $\lambda = 5$	**Time**	**Results** $\lambda = 10$	**Time**	**Results** $\lambda = 20$	**Time**
10	10	10	12.00	0.01	12.00	0.00	12.00	0.00
10	10	25	12.00	0.00	12.00	0.00	12.00	0.00
10	20	10	2.00	0.00	2.00	0.01	2.00	0.00
10	20	25	2.00	0.00	2.00	0.00	2.00	0.00
10	50	10	69.00	0.02	69.00	0.01	69.00	0.02
10	50	25	69.00	0.01	69.00	0.02	69.00	0.02
10	100	10	7.00	0.04	7.00	0.05	7.00	0.05
10	100	25	7.00	0.05	7.00	0.06	7.00	0.05
10	500	10	4.00	5.27	4.00	5.39	4.00	5.44
10	500	25	4.00	5.26	4.00	5.37	4.00	5.46
20	10	10	26.00	0.02	26.00	0.05	26.00	0.09
20	10	25	26.00	0.02	26.00	0.04	26.00	0.09
20	20	10	63.00	0.05	63.00	0.11	63.00	0.23
20	20	25	63.00	0.06	63.00	0.11	63.00	0.23
20	50	10	159.00	5.36	159.00	15.08	159.00	48.43
20	50	25	159.00	5.35	159.00	15.20	159.00	48.18
20	100	10	134.00	4.84	134.00	10.98	134.00	27.68
20	100	25	134.00	4.85	134.00	11.01	134.00	27.58
20	200	10	39.00	0.41	39.00	0.41	39.00	0.40
20	200	25	39.00	0.40	39.00	0.37	39.00	0.41
50	10	10	80.00	0.00	80.00	0.01	80.00	0.01
50	10	25	80.00	0.00	80.00	0.00	80.00	0.01
50	20	10	134.00	1800.28*	148.00	1800.71*	148.00	1800.39*
50	20	25	134.00	1800.18*	148.00	1800.05*	148.00	1800.91*
50	50	10	275.00	56.49	275.00	16.98	275.00	48.96
50	50	25	298.00	26.83	314.00	9.65	314.00	26.71
50	100	10	166.00	5.35	166.00	11.92	166.00	29.77
50	100	25	166.00	5.36	166.00	12.07	166.00	29.93
50	200	10	69.00	4.34	69.00	8.22	69.00	16.38
50	200	25	69.00	4.31	69.00	8.17	69.00	16.25

Table 6.3: Results for the exact solution approach on the smaller tandem instances. The first two columns hold the number of jobs and resources for the instance, respectively. All instances are of the Tandem type. The third column gives information on the amount of bandwidth per edge. Then follows two columns for three different wavelength settings, i.e., number of wavelengths per fiber: $\lambda = 5, 10,$ and 20. The two columns for each setting give the result value and the running time in seconds. When time usage finishes with a star ('*'), then the test has run out of time.

Jobs	Res.	BW	Results $\lambda = 5$	Time	Results $\lambda = 10$	Time	Results $\lambda = 20$	Time
100	10	10	147.00	0.98	147.00	1.91	147.00	4.27
100	10	25	147.00	0.98	147.00	1.94	147.00	4.25
100	20	10	285.00	4.05	285.00	9.85	285.00	27.62
100	20	25	285.00	4.09	285.00	10.00	285.00	27.53
100	50	10	713.00	1809.70*	738.00	1835.00*	738.00	2001.58*
100	50	25	801.00	1802.84*	807.00	1605.63	738.00	2006.66*
100	100	10	810.00	1800.29*	685.00	1800.39*	749.00	1815.39*
100	100	25	619.00	1800.93*	743.00	1803.58*	794.00	1803.44*
100	200	10	240.00	1801.48*	240.00	1820.37*	240.00	1802.53*
100	200	25	240.00	1811.03*	240.00	1808.99*	240.00	1811.26*
100	500	10	219.00	6.00	219.00	5.96	219.00	5.33
100	500	25	219.00	6.02	219.00	6.04	219.00	5.90
200	10	10	148.00	1800.43*	148.00	1800.51*	148.00	1801.23*
200	10	25	148.00	1800.15*	148.00	1800.04*	148.00	1800.92*
200	20	10	296.00	1802.13*	316.00	1800.37*	316.00	1802.04*
200	20	25	296.00	1800.97*	316.00	1801.71*	316.00	1800.90*
200	50	10	347.00	1806.79*	499.00	2134.76*	626.00	5410.29*
200	50	25	370.00	1915.16*	535.00	1921.21*	669.00	4224.28*
200	100	10	354.00	1817.09*	528.00	1836.88*	480.00	2077.20*
200	100	25	354.00	1875.56*	486.00	1860.29*	535.00	1912.69*
200	200	10	371.00	193.02	371.00	36.72	371.00	80.67
200	200	25	371.00	17.86	371.00	36.85	371.00	80.45
200	500	10	227.00	60.40	234.00	123.38	234.00	251.89
200	500	25	227.00	60.48	234.00	123.26	234.00	251.50

Table 6.4: Results for the exact solution approach on the larger tandem instances. The first two columns hold the number of jobs and resources for the instance, respectively. All instances are of the Tandem type. The third column gives information on the amount of bandwidth per edge. Then follows two columns for three different wavelength settings, i.e., number of wavelengths per fiber: $\lambda = 5, 10,$ and 20. The two columns for each setting give the result value and the running time in seconds. When time usage finishes with a star ('*'), then the test has run out of time.

Jobs	Res.	BW	Results $\lambda = 5$	Time	Results $\lambda = 10$	Time	Results $\lambda = 20$	Time
10	14	10	41.00	0.00	41.00	0.00	41.00	0.00
10	14	25	41.00	0.00	41.00	0.00	41.00	0.00
20	14	10	116.00	0.01	116.00	0.00	116.00	0.00
20	14	25	116.00	0.00	116.00	0.00	116.00	0.00
50	14	10	295.00	0.02	295.00	0.01	295.00	0.02
50	14	25	295.00	0.02	295.00	0.01	295.00	0.02
100	14	10	555.00	8.90	555.00	14.90	555.00	28.41
100	14	25	555.00	8.56	555.00	14.97	555.00	28.04
200	14	10	668.00	2677.35*	658.00	3736.75*	658.00	3053.28*
200	14	25	668.00	2010.45*	658.00	3777.48*	658.00	3033.20*

Table 6.5: Results for the exact solution approach on the NDGF instances. The first two columns hold the number of jobs and resources for the instance, respectively. All instances are of type NDGF. The third column gives information on the amount of bandwidth per edge. Then follows two columns for three different wavelength settings, i.e., number of wavelengths per fiber: $\lambda = 5, 10,$ and 20. The two columns for each setting give the result value and the running time in seconds. When time usage finishes with a star ('*'), then the test has run out of time.

An overview of comparing the heuristics with the branch-and-cut-and-price algorithm can be seen in Table 6.6. The table illustrates average solution value gaps and time usages for instances, which the exact algorithm has solved to optimality. As can be seen in the table, the heuristics only use a very small fraction of time compared to the exact approach. The solution value gap is never larger than 16%. For the grid heuristics, First Come First Serve has best performance, followed by Random Fit, Best Fit and First Fit. Fixed-Alternate Routing with 2 paths per data connection finds the smallest gaps, followed by Fixed-Alternate with 5 paths per connection, 1 path per connection and finally Adaptive Routing. No clear pattern emerges when considering wavelength assignment. For the First Come First Serve and Random Fit grid heuristics, First Fit wavelength assignment performs well. Otherwise Most Used has good performance.

The Table only reports average gaps for a subset of the instances; hence it does not give a full picture of the performance of the heuristics. This is determined next when comparing the heuristics to each other. An overview of this comparison is seen in Table 6.7. The summary is based on ranking the performance of the heuristics: the lower the rank the better performance. The average ranking of solution values for all instances is given in the `Solution` columns of the table and the average ranking of running times is given in the `Time` columns. An overview of actual time usage is seen in Figure 6.3-6.5.

The ranked results and the time usage illustrations are analyzed with respect to each of the main three heuristic approaches: the overall grid heuristic, the routing heuristic and

the wavelength assignment heuristic.

Grid	WA	R=FA, p=1 Solution	R=FA, p=1 Time	R=FA, p=2 Solution	R=FA, p=2 Time	R=FA, p=5 Solution	R=FA, p=5 Time	R=A Solution	R=A Time
FCFS	FF	3.46%	<1‰	2.89%	<1‰	3.17%	<1‰	6.63%	<1‰
FCFS	MU	3.46%	<1‰	4.12%	<1‰	4.40%	<1‰	7.86%	<1‰
FCFS	RF	4.95%	<1‰	3.01%	<1‰	3.29%	<1‰	7.13%	<1‰
BF	FF	10.06%	<1‰	9.48%	<1‰	10.05%	<1‰	13.74%	<1‰
BF	MU	8.95%	<1‰	8.37%	<1‰	8.94%	<1‰	12.63%	<1‰
BF	RF	9.61%	<1‰	8.37%	<1‰	8.94%	<1‰	13.01%	<1‰
FF	FF	11.65%	<1‰	10.80%	<1‰	10.04%	<1‰	15.32%	<1‰
FF	MU	10.54%	<1‰	9.69%	<1‰	8.93%	<1‰	14.21%	<1‰
FF	RF	11.19%	<1‰	9.69%	<1‰	8.93%	<1‰	14.59%	<1‰
RF	FF	4.11%	<1‰	3.80%	<1‰	4.52%	<1‰	9.58%	<1‰
RF	MU	3.46%	<1‰	5.60%	<1‰	5.49%	<1‰	13.22%	<1‰
RF	RF	6.25%	<1‰	4.13%	<1‰	5.03%	<1‰	10.55%	<1‰

Table 6.6: Performance of the heuristics compared to the exact results. The first two columns denote the grid and the wavelength assignment heuristics, respectively. Then follows pairs of comparison results, where the difference is measured in percent: the first column holds the average gap between the optimal and heuristic solution values and the second column holds the average heuristic time usage in per mille of the exact time usage. R stands for routing, and the options are FA (Fixed-Alternate) and A (Adaptive). p denotes the number of paths generated per data connection.

For the NDGF instances we see that the Best Fit grid heuristic gives better solution values than Random Fit, followed by the First Fit and First Come First Serve heuristics. The wavelength assignment heuristics perform equally well. For the routing strategy the best setting seems to be using Fixed-Alternate routing with 2 paths per data connection. Looking at time usage, then the First Come First Serve and Random Fit grid heuristics perform better than both Best Fit and First Fit. However, the graph in Figure 6.3 illustrates that the time difference is small for NDGF instances. The Most Used assignment generally requires more time than the other two wavelength assignment strategies, but again the time difference is small as seen in Figure 6.4 for NDGF instances. Finally, Adaptive routing uses less time than Fixed-Alternate, which becomes more time consuming as the number of generated paths per data connection increases. The time difference in small, see Figure 6.5.

For the tandem instances the First Come First Serve grid heuristic finds the best solution values. All wavelength assignment strategies seem to perform equally well with respect to solution values, while the Adaptive and Fixed-Alternate routing with 2 paths finds better solutions than other strategies. Looking at time usage in Figure 6.3, the First Come First Serve and Random Fit strategies are the faster grid heuristics, especially for the large instances with 500 jobs. Most Used requires more time than the

Grid	WA	R=FA, p=1 Solution	R=FA, p=1 Time	R=FA, p=2 Solution	R=FA, p=2 Time	R=FA, p=5 Solution	R=FA, p=5 Time	R=A Solution	R=A Time
FCFS	FF	0.86	2.19	0.70	2.15	0.81	2.41	0.90	2.38
FCFS	MU	0.99	3.06	0.86	3.23	0.92	2.88	0.94	3.75
FCFS	RF	0.99	1.64	0.77	1.68	0.81	2.61	0.97	2.32
BF	FF	1.97	3.45	1.79	3.53	2.09	3.83	1.37	3.52
BF	MU	1.88	4.27	1.73	4.27	2.03	3.86	1.40	4.61
BF	RF	1.92	2.83	1.73	3.08	2.06	3.91	1.43	3.57
FF	FF	2.59	3.54	2.35	3.89	2.20	3.95	2.11	3.73
FF	MU	2.48	4.30	2.34	4.36	2.18	4.01	2.13	4.71
FF	RF	2.51	3.09	2.29	3.17	2.18	4.19	2.13	3.72
RF	FF	1.75	2.01	1.64	1.97	1.65	2.24	1.77	2.11
RF	MU	1.87	3.17	1.80	3.33	1.83	3.04	1.89	3.78
RF	RF	1.96	1.38	1.76	1.40	1.74	2.35	1.90	2.10
FCFS	FF	3.29	1.69	3.28	1.70	3.58	1.86	3.28	1.69
FCFS	MU	3.29	2.79	3.28	3.17	3.58	3.35	3.28	2.76
FCFS	RF	3.29	1.79	3.28	1.94	3.58	2.06	3.28	1.62
BF	FF	0.56	2.72	0.56	3.20	0.56	3.85	0.58	3.53
BF	MU	0.56	5.02	0.56	5.00	0.56	5.36	0.58	4.99
BF	RF	0.56	3.47	0.56	3.58	0.56	3.93	0.58	3.68
FF	FF	2.66	4.07	2.06	4.47	2.23	4.22	2.91	4.05
FF	MU	2.66	5.44	2.06	5.44	2.23	6.34	2.91	5.12
FF	RF	2.66	4.65	2.06	4.08	2.23	4.69	2.91	3.88
RF	FF	1.18	1.19	1.30	1.70	1.46	1.72	1.32	1.57
RF	MU	1.45	3.59	1.15	2.94	1.19	3.48	1.20	3.11
RF	RF	1.15	1.56	1.43	2.05	1.27	1.94	1.50	1.92

Table 6.7: Performance of the heuristics having been ranked for best time and solution value for the tandem (top) and the NDGF (bottom) instances. The table displays the average ranking, R stands for routing, and the options are FA (fixed-alternate) and A (adaptive). p denotes the number of paths generated per data connection.

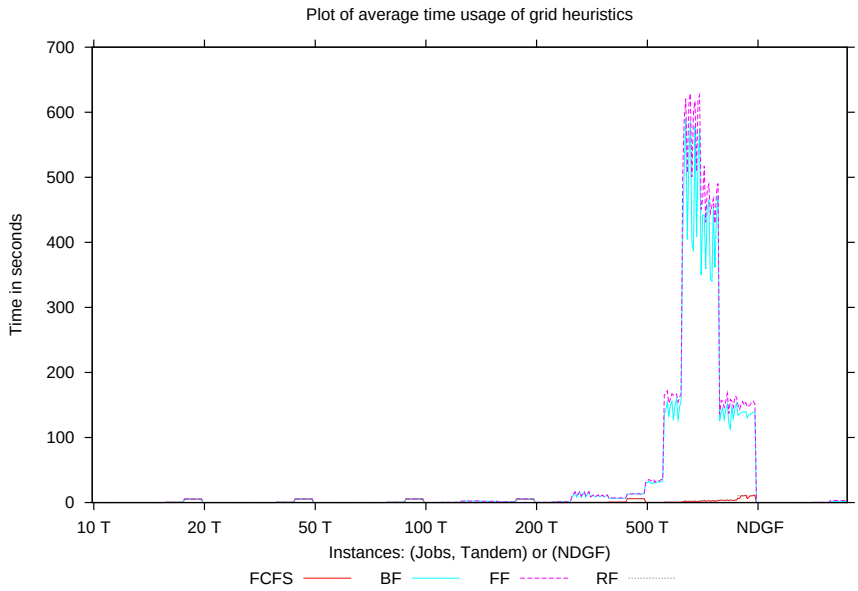


Figure 6.3: Illustration of time usage in seconds for the grid heuristics. The x-axis denotes instances, where the first number indicates the number of jobs for the tandem instances and where the last part indicates the NDGF instances. Plots for instances with the denoted number of jobs and with 10-500 resources are given between two tics on the x-axis.

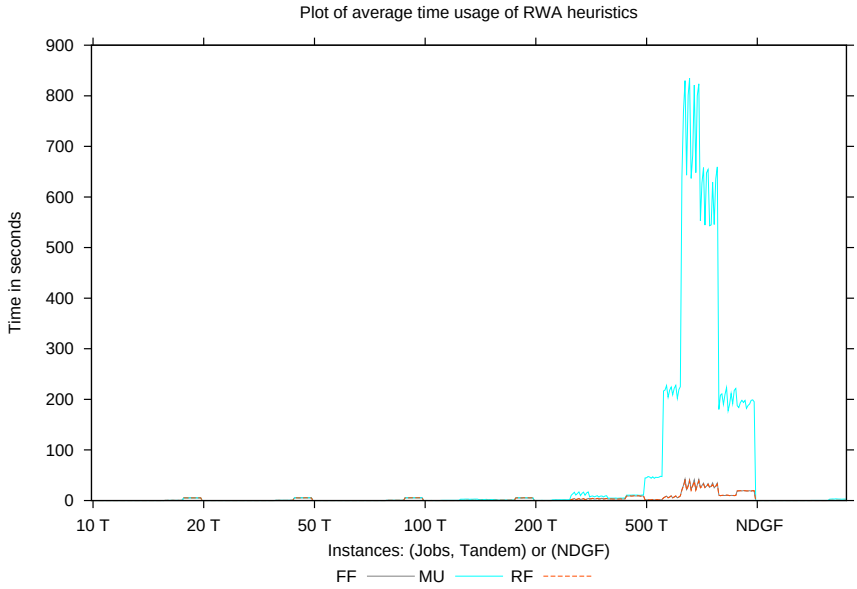


Figure 6.4: Illustration of time usage in seconds for the RWA heuristics. The x-axis denotes instances, where the first number indicates the number of jobs for the tandem instances and where the last part indicates the NDGF instances. Plots for instances with the denoted number of jobs and with 10-500 resources are given between two tics on the x-axis.

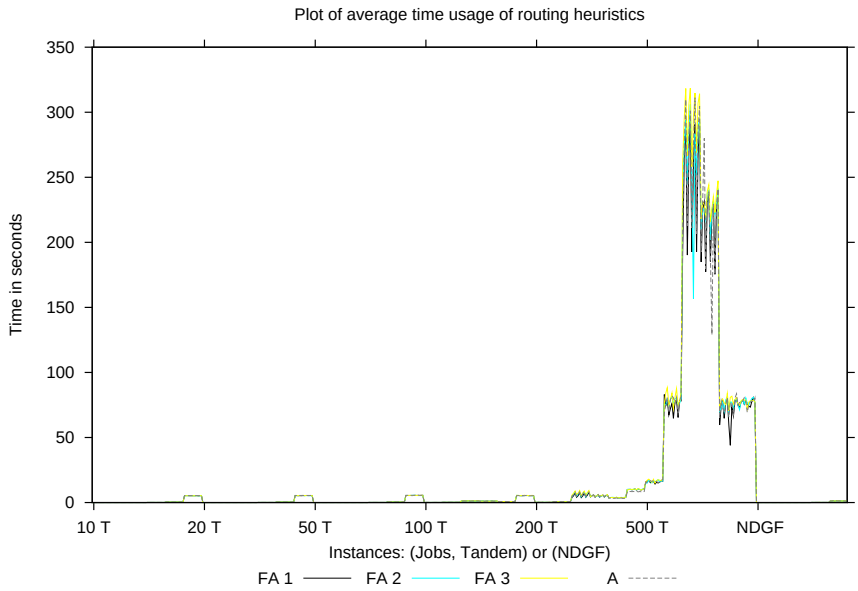


Figure 6.5: Illustration of time usage in seconds for the routing heuristics. The x-axis denotes instances, where the first number indicates the number of jobs for the tandem instances and where the last part indicates the NDGF instances. Plots for instances with the denoted number of jobs and with 10-500 resources are given between two tics on the x-axis.

other wavelength assignment strategies and for the large tandem instance, the time difference is significant as seen in Figure 6.4. The Fixed-Alternate becomes more time consuming as the number of generated paths per data connection increases and the Adaptive routing is even slightly slower. The time difference between routing heuristics is insignificant, though, which can be seen in Figure 6.5.

Looking at general time usage, the practical running times reflect the theoretical running times from Section 6.3.1. For wavelength assignment this means that the Most Used strategy generally requires more time than First Fit and Random Fit. Adaptive routing is generally faster than Fixed-Alternate routing whose running time increases with the number of generated paths per data connection. Finally, the First Come First Serve and Random Fit grid heuristics have smaller time usage than Best Fit and First Fit.

Comparing the heuristics with each other gives a slightly different pattern than when comparing heuristics with the exact solution results. This is due to two reasons. 1: Not all instances were considered when comparing with exact results, because the BCP algorithm was not able to solve all instances. 2: The average gap may be large if a heuristic gives very poor results for few instances but good results for all other instances. The ranking system does not care how far off a result may be and does thus not punish very poor performance equally hard.

Overall analysis

Using a black-box strategy for solving JSCNR may not always be the best choice. Instead the grid administrator should identify the current bottlenecks with respect to scheduling and network usage in order to find a good heuristic. The Best Fit grid heuristic utilizes available resources well for instances with no or little network problems. This is concluded from considering the NDGF instances, where paths share few edges. A reason for this is that Best Fit makes sure that jobs are placed according to them taking up as little time space in the network and on the resource as possible, hence giving good resource utilization. When the network constitutes a significant bottleneck, then First Come First Serve makes sure that jobs are forwarded to execution soon after arrival, which yields the best solution values. This is seen in the Tandem instances, where paths share many edges. A reason for this is that the strategy uses network wavelengths as early as possible instead of at some later time; when the latter is the case, then the time slots at which wavelengths become available after a subset of jobs are assigned, may become so small that data for the remaining jobs cannot arrive at the executing resource in time. Time usage must also be taken into account. If the grid system consists of many resources and/or many jobs, then it may be beneficial to choose a more straightforward grid heuristic like First Come First Serve, regardless of network constraints.

Most Used wavelength assignment may often give better results than both First Fit and Random Fit but also requires more time. The reason for the better results is that by choosing the most used wavelength, more wavelengths may be available for the next data connection request. First Fit and Random Fit assignment generally perform equally well both with respect to solution values and time usage.

Generally, the best routing strategies with respect to solution values are Fixed-Alternate routing with 2 paths per data connection closely followed by Adaptive Routing and Fixed-Alternate with 5 paths per data connection. The Fixed-Alternate routing considers previously routed data connection and thus has good performance when generating more than 1 path per data connection.

A final recommendation is based on the comparison with exact solution values in Table 6.6, on the comparison of heuristics in Table 6.7 and on time usage in Figure 6.3-6.5. We suggest using First Come First Serve grid scheduling, Fixed-Alternate routing with 2 paths per data connection and First Fit wavelength assignment. This setting generally gives lower gaps compared to exact values and it also generally has best performance when only considering the heuristics. The solution approach, however, should be decided based on an analysis of the grid topology and the expected CPU and network load. For small instances, an exact solution can be found within reasonable time. For larger instances some consideration should be given on which grid heuristic is more appropriate.

6.6 Conclusion

This paper introduced The Integrated Job Scheduling and Constrained Network Routing Problem (JSCNR) with application in production planning and telecommunication. JSCNR was formally presented and a mathematical formulation was given. JSCNR was shown to be $\mathcal{NP}$-hard, as it holds both the $\mathcal{NP}$-hard Integrated Job Scheduling and Network Routing Problem (JSNR) and the $\mathcal{NP}$-hard Routing and Wavelength Assignment Problem (RWA) as special cases.

A branch-and-cut-and-price (BCP) algorithm for JSCNR was presented, where the pricing problem assigns a job to a machine and the master problem merges the job assignments into an overall feasible solution. Finally, a number of heuristics for JSNR was presented along with a number of heuristics for RWA and they were merged into a total of 24 different heuristic solution methods for JSCNR.

The proposed methods were computationally evaluated on two types of test instances:

a tandem topology with 10-500 machines and a real-life network topology taken from the Nordic DataGrid Facility with 14 machines.

Using CPLEX to solve the mathematical formulation yielded somewhat poor results as only smaller instances were solved due to memory and time problems. The BCP algorithm was capable of solving more instances, however, it still timed out for several instances because its pricing problem is $\mathcal{NP}$-hard.

All heuristics were tested and compared with the exact approach and with each other. The computational results showed that First Come First Serve job assignment heuristic gave best results along with the routing strategy, which proposes two routes for each demand. The running times of the computational evaluations reflected the theoretical running times for the heuristics well. Furthermore, all instances were solved within minutes.

Future work on JSCNR could concentrate on finding optimal solutions. The proposed decomposition resulted in an $\mathcal{NP}$-hard pricing problem, which caused time issues. Future work could consider other decompositions with possibly less complex pricing problems.

It would also be relevant to consider metaheuristics, e.g., local search methods. The heuristics presented in this work could be used as base in metaheuristics. It is expected that metaheuristics would improve the solution quality, but would also have greater running times. Metaheuristics are expected to provide a good alternative with performance lying between that of the greedy heuristics and of the BCP algorithm with respect to solution quality and time usage.

Acknowledgement
We would like to thank GlobalConnect A/S for their support of this work.

CHAPTER 7

Analysis of internal network requirements for the distributed Nordic Tier-1

Gerd Behrmann
NDGF - Nordic DataGrid Facility, Kastruplundgade 22(1), DK-2770 Kastrup

Lars Fischer
NORDUnet, Kastruplundgade 22(1), DK-2770 Kastrup

Mette Gamst
Technical University of Denmark, Department of Management Engineering, Produktionstorvet, DTU - Building 424, DK-2800 Kgs. Lyngby

Michael Grønager
NDGF - Nordic DataGrid Facility, Kastruplundgade 22(1), DK-2770 Kastrup

Working paper 2010

Josva Kleist
NDGF and Aalborg University, Department of Computer Science, Selma Lagerlöfsvej 300, DK-9220 Aalborg SØ

The Nordic DataGrid Facility (NDGF) provides a grid computing system connected primarily by a Tier-1 network, i.e., a network which can be used without purchasing IP transit or paying settlements. The Tier-1 facility operated by NDGF differs significantly from other Tier-1s in several aspects. It is not located at one or a few locations but instead distributed throughout the Nordic countries. Also, it is not under the governance of a single organization but is instead built from resources under the control of a number of different national organizations. Being physically distributed makes the design and implementation of the networking infrastructure a challenge. To assess the suitability of the network usage and the capacity of the links, we present a model of the bandwidth needs for the NDGF Tier-1 and its associated Tier-2 sites. The model takes different types of workload into account and calculates bandwidth requirements based on the workload type characteristics. The model differs from work in the literature, which assumes full knowledge on each job and its data file requirements rather than on workload types. The model of the distributed Nordic Tier-1 is used as a strategic tool to calculate an optimal placement of workloads, to measure the impact of including caches on different locations and to suggest better resource distributions.

Key words: Grid Computing; Scheduling; Tier-1; Mathematical Programming; Optimization

7.1 Introduction

Dimensioning the network for a Tier-1 is always a challenge, particularly when the Tier-1 is distributed as is the case of the Nordic Tier-1 operated by Nordic DataGrid Facility (NDGF). The Tier-1 is defined as a network which can be used without purchasing IP transit or paying settlements - contrary to e.g. Tier-2s where IP transit must be purchased to reach parts of the network. See Kurose and Ross [132] for more details on Tier networks. The NDGF system is built from resources under the control of a number of different national organizations. The NDGF Tier-1 consists of the seven biggest Nordic compute sites (denoted the dTier-1s) with associated Tier-2 resources as far away as Slovenia, see Fischer et al. [71]. Storage and computing resources are widely scattered with a few central services. This gives many advantages in redundancy especially for 24x7 data taking, as reported by Field et al. [69]. Figure 7.1 shows the storage and compute sites participating in the NDGF system, including available resources as of the second quarter of 2009.

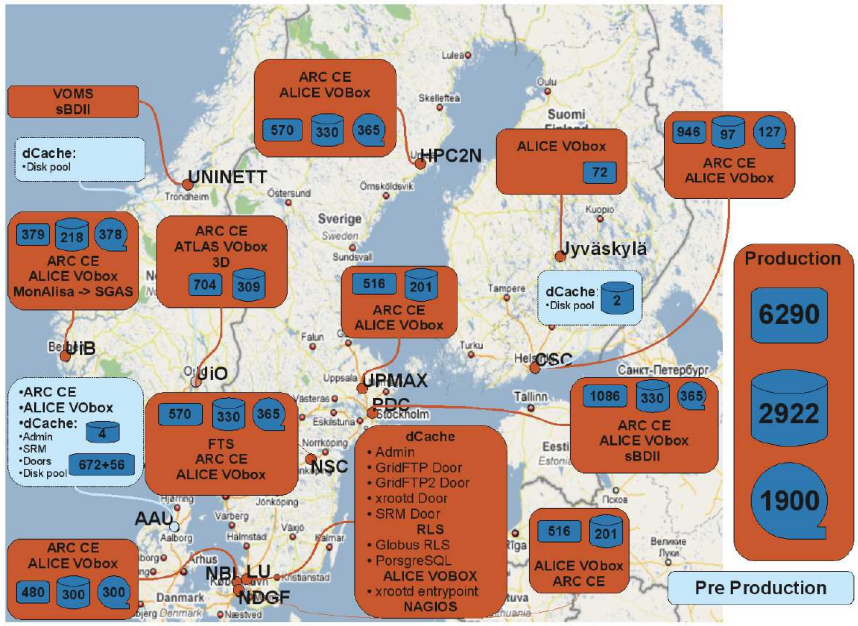


Figure 7.1: NDGF distributed storage and computational setup in the second quarter of 2009. The amounts of compute, disk and tape resources (if any) at each site are reported. Compute resources are given in KSI2K and storage resources in terabytes.

NDGF uses a dedicated optical network between all dTier-1 sites and the Slovenian Tier-2. The remaining Tier-2 sites are connected via the national research network. Figure 7.2 shows how NDGF sites are interconnected with special emphasis on Sweden; red lines depict dedicated private network lines and magenta lines depict public lines. The main network infrastructure forms a star. All data from the central NDGF router to each country travels on a single link, which makes it easy to calculate the load between the central NDGF router and the country.

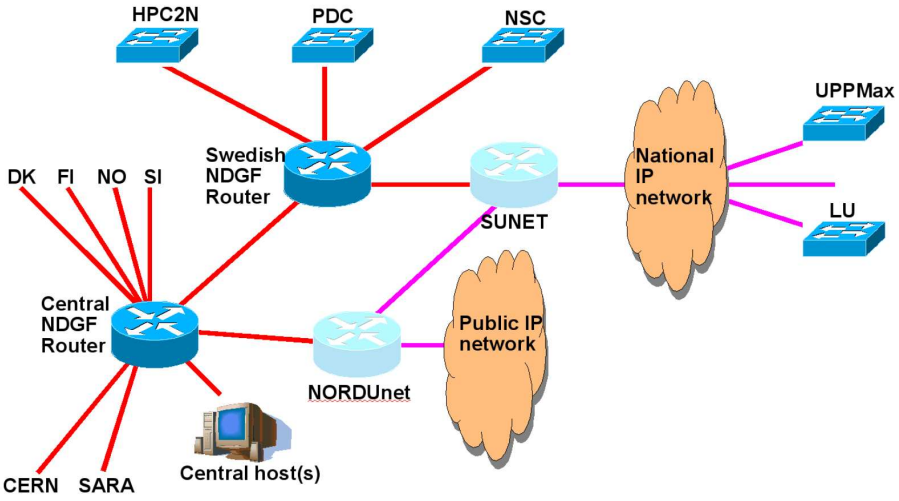


Figure 7.2: NDGF network layout with emphasis on Sweden. Red lines depict dedicated links and magenta lines depict public links.

The distributed Nordic Tier-1 is a grid computing system. Much research has been conducted on deciding the activity in grid context; the research concentrates on scheduling jobs for execution on grid sites. Especially online scheduling is investigated in the literature, i.e., the problem of assigning a job to a site where the assignment takes place at job arrival time or when the site becomes available, see e.g. Foster and Kesselman [76]. Work in the literature on activity in grid computing assumes detailed information on where all data files are stored, on which files are required by each job, on when each job is executed, on when data is transmitted etc., see e.g. the work of Chakrabarti et al. [42], Ranganathan and Foster [165], and Tang et al. [185]. When dimensioning the distributed Nordic Tier-1 we do not have detailed information on the grid activity. We do not have a specific order of the single jobs to be executed at each site, and we do not know the specific files required by each job. Instead, we know the job type mix at each site and the amount of data reads and writes required by each job type. Also, we initially do not want to change job or data placement, rather we wish to calculate bandwidth requirements given the current activity. In this way, this paper differs from work in the literature performed on bandwidth requirements in grid computing context.

Formalizing the NDGF network provides us with a number of equations to calculate the network usage. This is of the network and its capacities. We gather the calculations into a mathematical model, which is used to analyze changes to the activity of the distributed Nordic Tier-1. In this way we can use the formalization to identify bottlenecks in the network with the current activity and to identify future bottlenecks when changing the activity. The bottlenecks can help us decide how to extend the distributed Nordic Tier-1 with respect to storage and compute resources and bandwidth availability. Furthermore, we believe that the formalization can be viewed as a general framework, which can be applied to any distributed Tier-1 with only few changes to match specific data reads and writes.

This paper is organized as follows. First in Section 7.2, we present the calculations of bandwidth requirements for the distributed Nordic Tier-1. First some assumptions on activity in the network is made. Then we analyze site availability and general job requirements. From this we present a general framework for calculating bandwidth usage. Finally, the section presents the job types present in the NDGF network. Based on the equations from Section 7.2, we calculate the actual bandwidth requirements in the NDGF network in Section 7.3. The results seem to represent real-life activity well. We gather the equations for calculating bandwidth usage into a mathematical model in Section 7.4. The model is used for calculating changes in the activity of the NDGF network such that the maximal network link load is minimized. We investigate the impact of changing the placement of jobs according to network usage and according to users of the system. We discuss and investigate the effects of adding a basic caching mechanism, and finally we analyze if changing the distribution of storage and compute resources can lower network usage. Section 7.5 discusses the correctness of the model for calculating bandwidth usage. It also summarizes the results of imposing changes to the network activity and suggests interesting future analyses. Finally, the section proposes future work to be performed in order to ensure a precise model and an optimal utilization of the distributed Nordic Tier-1.

7.2 Calculating network requirements

We wish to introduce a model to calculate an estimate of bandwidth requirements by the distributed Nordic Tier-1. For this we need to analyze and formalize all data transmissions from workloads, data reads and data writes. Workloads are divided into jobs, where each job type has specific data requirements. In this section we analyze the sites, bandwidth requirements and job types.

7.2.1 Basic assumptions

Before presenting a model of bandwidth usage in the NDGF network, we need to introduce some basic assumptions on placement of jobs, when and which files are transmitted through the network, etc. Currently, NDGF does not have detailed information on which data files each job requires, or on exactly how or when jobs are received. This information, however, is required before bandwidth usage can be calculated. The following assumptions compensate for the lack of information. They are based on experiences of NDGF administrators and should reflect real-life activity well.

- All sites are occupied up to their efficiency, i.e., we assume a sufficient amount of job and data availability in the system at any time.
- Data is randomly and uniformly distributed over all storage sites, i.e., the amount of data available at a specific storage site is proportional to the size of the storage site in relation to the total amount of storage in the system.
- The characteristics of the different job types is known, i.e., we know in advance how much data a job consumes and generates and we know how many compute resources are required for each job type.
- The job mix at a site is known in advance, i.e., we know how many percent a specific job type spends of the available compute time.
- Jobs are spread temporarily and uniformly across the sites. This means that we do not have bursts of a specific job type.
- Traffic flows directly between the storage site and the compute site where a job is executed. No intermediate servers are involved. This is in fact the case for the NDGF setup.
- The caching mechanism in the ARC grid middleware, see Ellert et al. [65], is not taken into account. The ARC middleware employed by NDGF for its ATLAS computations includes a caching mechanism that can significantly reduce network traffic, see Behrmann et al. [34]. Modeling the impact of the caching mechanism is difficult without any empirical evidence on what effect the cache has on different job types.

Later in this paper we discuss how changes to those assumptions impact the model.

7.2.2 Site characteristics

For a site s, we assume to know the amount of installed tape (T^s), the amount of installed disk (D^s), the amount of compute resources for each of the two Tiers (C^{s_1} and C^{s_2}) and the compute efficiency for each Tier in percent ($0 \leq e^{s_1} \leq 1$ and $0 \leq e^{s_2} \leq 1$). Some NDGF sites act as both Tier-1 and Tier-2 sites. This is why we allow a site to have a number of compute resources and efficiencies. We let C, T and D denote the total amount of CPU, tape and disk in the system, respectively.

7.2.3 Job characteristics

NDGF considers a number of different job types. For each job type j we assume to know the amount of CPU required to run the job (R_j), the amount of data read from disk while executing (DI_j), the amount of data read from tape while executing (TI_j), the amount of data written to disk while executing (DO_j), and the amount of data written to tape while executing: (TO_j).

Each site runs a special job mix; a job type j is supposed to occupy a certain fraction of the available compute time. We let J denote the total set of job types in the system. Let $f_j^{s_i}$ denote the fraction for job type j on the Tier-i resources at site s. We assume that $\sum_{j \in J} f_j^{s_i} = 1$, i.e. that all jobs of each type are executed.

We can now calculate the amount of data to be read from and written to disk and tape at a site s. Each job type j needs DI_j data from disk and runs for R_j CPU seconds (this could be any general measurement of CPU performance like *Kilo Specmarks Integer year 2000 (KSI2K)* [182]). Therefore DI_J/R_J denotes the amount of data a job j requires per CPU second. The number of CPU seconds a job type requires on a site is calculated as $C_j^{si} = f_j^{s_i} e^{s_i} C^{s_i}$, i.e., the fraction of the resource that runs jobs of type j multiplied with the efficiency times and with the total amount of computational resources. Now, the amount of required data for all computations on a site is given by:

$$DI_C^s = \sum_{j \in J} \sum_{i \in \{1..2\}} C_j^{si} \frac{DI_j}{R_j}$$

Similarly, we can calculate values for tape read TI^s, disk write DO^s and tape write

TO^s for each site s:

$$\begin{aligned} TI^s_C &= \sum_{j \in J} \sum_{i \in \{1..2\}} C^{si}_j \frac{TI_j}{R_j} \\ DO^s_C &= \sum_{j \in J} \sum_{i \in \{1..2\}} C^{si}_j \frac{DO_j}{R_j} \\ TO^s_C &= \sum_{j \in J} \sum_{i \in \{1..2\}} C^{si}_j \frac{TO_j}{R_j} \end{aligned}$$

7.2.4 Bandwidth requirements

Parts of the data at a site s can be read and written locally. Assuming a uniform distribution of data over the sites, the part that can be read and written locally corresponds to the fraction of storage available at s in relation to the total amount of storage in the system (D^s/D and T^s/T). The amount of data, which cannot be read locally by site s is expressed as:

$$BI^s_C = DI^s_C \frac{D - D^s}{D} + TI^s_C \frac{T - T^s}{T}$$

Similarly, the amount of data site s must write to other sites is formulated as:

$$BO^s_C = DO^s_C \frac{D - D^s}{D} + TO^s_C \frac{T - T^s}{T}$$

Furthermore, other sites will read from and write to the disk and tape resources at site s. Again the amount corresponds to the relation between the installed disk and tape capacity at s and the total installed capacity (D^s/D and T^s/T). The amount of data, which other sites must write to site s is defined as:

$$BI^s_O = \sum_{t \in S \setminus \{s\}} \left(DO^t_C \frac{D^s}{D} + TO^t_C \frac{T^s}{T} \right)$$

Similarly, we can calculate values for reads from site s by all other sites:

$$BO^s_O = \sum_{t \in S \setminus \{s\}} \left(DI^t_C \frac{D^s}{D} + TI^t_C \frac{T^s}{T} \right)$$

Finally, we need to take traffic external to NDGF into account. Again the same arguments as before apply and the amount of external traffic depends on the relation between installed disk and total installed capacity:

$$BI^s_E = BI_E \frac{D^s}{D}$$

The input bandwidth requirements for a site s become:

$$BI^s = BI^s_c + BI^S_O + BI^s_E$$

For bandwidth out of a site, we derive a similar equation:

$$BO^s = BO^s_C + BO^s_O + BO_E \frac{D^s}{D}$$

The bandwidth requirement for a site is then the maximum of BI^s and BO^s.

As output the model gives the average network throughput at a site, assuming that all compute resources are occupied up to their efficiency with a certain mix of jobs. It will not take burst into account; neither will it include any overhead caused by transport protocols.

7.2.5 Job types

Table 7.1 forms the basis for calculating bandwidth requirements in the distributed Nordic Tier-1. The table holds information on how much a job type takes up Tier-1 and Tier-2, respectively, in percent. Furthermore, the table gives information on run times, disk reads and writes, and tape reads. The instance includes no tape writes. It is noted that three job types are present in the distributed Nordic Tier-1: ALICE, ATLAS and CMS. The job types stem from the experiments performed by the Large Hadron Collider (LHC) by the European Organization for Nuclear Research (CERN). Each of the three experiments is expected to generate huge amounts of data, hence both scientific work on the data and data storage itself is distributed world-wide on grids. One of these grids is the NDGF network. For more details on the LHC experiments, we refer to Shiers [176] and the project homepage [41]. For more details on the relationship between NDGF and CERN, we refer to Anderlik et al. [9] and the website of NDGF [154].

7.3 NDGF bandwidth requirements

The calculations of bandwidth usage from Section 7.2 are applied to the current NDGF setup. The outcome can be seen in Table 7.2, Table 7.3 and Table 7.4. Tables 7.2 and 7.3 display job and data distribution, respectively. Then follows Table 7.4 displaying network usage. The placement of jobs and data is given in advance; therefore the interesting part is the resulting network usage. The network traffic seems to reflect real-life activity well. The link between the central NDGF router and Sweden has high

Job name	Tier-1	Tier-2	Run time	Disk in	Disk out	Tape in
ALICE analysis	20%	50%	1	1000	10	0
ALICE recon	40%	0%	5	10	100	1000
ALICE MC	40%	50%	15	10	10000	0
ATLAS analysis	20%	50%	1	100	100	0
ATLAS recon	40%	0%	1	10	100	1000
ATLAS MC	40%	50%	12	100	500	0
CMS analysis	20%	50%	1	100	100	0
CMS recon	40%	0%	2	100	100	2000
CMS MC	40%	50%	12	100	500	0

Table 7.1: Job information used to calculate network usage. First column holds the job name. This is followed by the expected percentage each job takes up Tier-1 and Tier-2. Then comes the expected run time, the estimated amount of disk reads and writes, and finally the estimated amount of tape reads. The instance holds no tape writes, which thus have been omitted from the table.

load compared to the remaining links, which is caused by the many sites in Sweden. Generally, network traffic is distributed according to compute and resource availability in each country. To avoid network bottlenecks it is interesting to investigate if changes to job and data placement could distribute traffic more evenly across the network. This is investigated in the following section.

7.4 Analyzing changes to the distributed Nordic Tier-1 network

The distributed Nordic Tier-1 is formalized into a mathematical formulation to investigate the effects of changing different network or site settings, such as job placement, data distribution, introduction of caches etc. The goal of the model is to reduce the maximal link load in hope of distributing bandwidth requirements more evenly across the system. The constraints ensure that job placement, data distribution, bandwidth limitations and the network topology are satisfied. The constraints are formed by the formalization of bandwidth requirements in Section 7.2. When minimizing the maximal link load we want to impose changes on one or several of the constraints. This is done by introducing variables. An example is when job placement is not fixed. The constraints of the mathematical model would then state that all jobs must be executed and that no compute resources are exceeded.

Considering the introduced notation, the problem of minimizing the maximal network

Tier-1	ALICE	ATLAS	CMS	Total
DK	240	240		480
FI	280			280
NO	410	348		758
SE	727	983		1710
SL				
Sum	1657	1571		3228

Tier-2	ALICE	ATLAS	CMS	Total
DK				0
FI			666	666
NO		325		325
SE	624	925		1549
SL		600		450
Sum	624	1700	666	2990
Total	2281	3271	666	6218

Table 7.2: The given distribution of job execution. The first column holds the name of the compute site. Next follows the amount of compute resources in KSI2K dedicated to each of the three job types. Finally, the sum of compute resources is given.

load ($B \geq 0$) can be formalized. The mathematical model is:

$$\min \quad B \tag{7.1}$$

$$\text{s.t.} \quad B \geq BI^s \qquad \forall s \in S \tag{7.2}$$

$$B \geq BO^s \qquad \forall s \in S \tag{7.3}$$

$$BI^s = BI^s_C + BI^s_O + BI^s_E \qquad \forall s \in S \tag{7.4}$$

$$BO^s = BO^s_C + BO^s_O + BO^s_E \qquad \forall s \in S \tag{7.5}$$

$$BI^s_C = DI^s_C(D - D^s)/D + TI^s_C(T - T^s)/T \qquad \forall s \in S \tag{7.6}$$

$$BO^s_C = DO^s_C(D - D^s)/D + TO^s_C(T - T^s)/T \qquad \forall s \in S \tag{7.7}$$

$$BI^s_O = \sum_{t \neq s} (DO^t_C \cdot D^t/D + TO^t_C \cdot T^t/T) \qquad \forall s \in S \tag{7.8}$$

$$BO^s_O = \sum_{t \neq s} (DI^t_C \cdot D^t/D + TI^t_C \cdot T^t/T) \qquad \forall s \in S \tag{7.9}$$

$$DI^s_C = \sum_j \sum_i C^{si}_j \cdot DI_j/R_j \qquad \forall s \in S \tag{7.10}$$

$$TI^s_C = \sum_j \sum_i C^{si}_j \cdot TI_j/R_j \qquad \forall s \in S \tag{7.11}$$

Disk Tier-1	ALICE	ATLAS	CMS	Total
DK	150	150		300
FI	97			97
NO	251	185		436
SE	310	681		991
SL				0
Sum	808	1016	0	1824

Disk Tier-2	ALICE	ATLAS	CMS	Total
DK				0
FI			205	205
NO		91		91
SE	148	454		602
SL		200		200
Sum	148	745	205	1098

Total	956	1761	205	2922

Tape	ALICE	ATLAS	CMS	Total
DK	150	150		300
FI	127			127
NO	250	128		378
SE	578	517		1095
SL				0
Sum	1105	795	0	1900

Table 7.3: The given distribution of data storage. First, the name of the storage site is given. Then follows the amount of stored data for each job type in terabytes. Finally, the amount of stored data is summed.

Site/Country	Network load
DCSC/KU	1.5 Gbps
Denmark	1.5 Gbps
CSC	1.0 Gbps
Jyv	0.0 Gbps
Finland	1.0 Gbps
UiB	1.3 Gbps
UiO	1.1 Gbps
Norway	2.2 Gbps
HPC2N	1.8 Gbps
LUNARC	0.6 Gbps
PDC	2.3 Gbps
NSC	1.8 Gbps
UPPMAX	0.6 Gbps
Sweden	**4.4 Gbps**
PIKOLIT	0.1 Gbps
Slovenia	0.1 Gbps

Table 7.4: Network load between sites and the NDGF main router. The first column holds the name of the different sites and the second column holds network loads. Note that the total amount of traffic between each country and the NDGF main router is given.

$$DO_C^s = \sum_j \sum_i C_j^{si} \cdot DO_j / R_j \quad \forall s \in S \tag{7.12}$$

$$TO_C^s = \sum_j \sum_i C_j^{si} \cdot TO_j / R_j \quad \forall s \in S \tag{7.13}$$

$$B \geq 0 \tag{7.14}$$

The objective function (7.1) minimizes B, which is the maximal network load. Constraints (7.2) and (7.3) ensure that the maximal network load B is greater or equal to all link loads in the network. The constraints (7.4) and (7.5) calculate the amount of in- and outgoing bandwidth usage at each site. Constraints (7.6) and (7.7) define the amount of data being read from other sites and written to other sites by site s, respectively. The next two constraints (7.8) and (7.9) define the amount of data being written to and read from s by other sites, respectively. Next, we have the total amount of disk data (7.10) and tape data (7.11) to be read by a site s. Correspondingly, the total amount of disk data (7.12) and tape data (7.13) to be written is defined. Finally, the bound (7.14) ensures that the variable indicating the maximal link load is non-negative.

The model can obviously be solved in constant time; given a problem instance, network usage is immediately calculated and the model returns the largest amount of bandwidth travelling on any link. In the following sections, we change the model slightly to represent changes in the problem instance. For example, given the job types to be calculated

and a fixed data placement, it is interesting to find an optimal job placement. In this case, we must introduce some variables in the model to represent job placements.

The resulting models of the following analyses are all solved by CPLEX version 10.2 [102], even though not all problems are $\mathcal{NP}$-hard. The reason for this is that the emphasis of the following analyses is on the resulting bandwidth usage and not on solution techniques. The problem instances are small and consist of eleven sites and three job types, so even $\mathcal{NP}$-hard problems are solved very quickly.

7.4.1 Optimizing job placement

The current job placement from Section 7.3 may not be optimal. Without imposing any changes to resources on each site or to job properties, we try to re-arrange the job placement such that the maximal link load is minimized. This corresponds to transforming $0 \leq f_j^{si} \leq 1$ into a variable, which denotes the percentage a job j takes up compute resources at site s, Tier-i. To ensure that a job is fully executed and that compute resources at each site are not exceeded, we introduce the extra constraints:

$$\sum_{s \in S} C^{si} f_j^{si} = R_j \quad \forall i \in I, \forall j \in J$$

$$\sum_{j \in J} C_j^{si} \leq C^{si} \quad \forall i \in I, \forall s \in S$$

The added variables f_j^{si} and all constraints in the model (7.1)-(7.14) are linear, so the resulting problem is polynomial. Results of optimizing job placement are seen in Table 7.5 for job placement and in Table 7.6 for network usage. The maximal network link load is reduced from 4.4 Gbps to 3.5 Gbps. The link between the central NDGF router and Sweden still carries more data than the other links, which is not surprising because Sweden has more compute resources and thus requires and produces more data. Link loads between the central NDGF router and the remaining countries have generally increased; minimizing the maximal link load causes a more evenly distribution of data transmissions. This is beneficial for NDGF, because they in this way may avoid or decrease the risk of network bottlenecks.

7.4.2 Virtual organizations

NDGF requires all participants of the system to be *Virtual Organizations (VOs)*. The national organizations supplying resources are thus registered as VOs. Similarly each user of the system registers as a VO. The three job types in the system, *ALICE*, *ATLAS* and *CMS* require both Tier-1 and Tier-2 resources. This is translated into having six

Tier-1	ALICE	ATLAS	CMS	Total
DK		480		480
FI		280		280
NO	130	628		758
SE	1527	183		1710
SL				
Sum	1657	1571		3228

Tier-2	ALICE	ATLAS	CMS	Total
DK				0
FI		666		666
NO		325		325
SE	624	259	666	1549
SL		450		450
Sum	624	1700	666	2990
Total	2281	3271	666	6218

Table 7.5: The result of optimizing job placement. The table shows the job placement. Comparing to the initial job placement in Table 7.2 it is noted that all job types are fully executed and that all compute capacities are satisfied.

Site/Country	Network load
DCSC/KU	1.8 Gbps
Denmark	1.8 Gbps
CSC	1.0 Gbps
Jyv	0.0 Gbps
Finland	1.0 Gbps
UiB	1.4 Gbps
UiO	1.2 Gbps
Norway	2.4 Gbps
HPC2N	1.9 Gbps
LUNARC	0.6 Gbps
PDC	2.0 Gbps
NSC	1.9 Gbps
UPPMAX	0.6 Gbps
Sweden	**3.5 Gbps**
PIKOLIT	0.1 Gbps
Slovenia	0.1 Gbps

Table 7.6: The result of optimizing job placement. The table shows network usage.

users (VOs). In this analysis, we wish to assign a VO (job type) to each site. This, however, is not possible with the current workload; hence the goal is modified into minimizing the number of job types per site. The secondary goal is to minimize the maximal network link load. In the mathematical formulation we introduce an integer variable $x_s^i \in \mathbb{Z}_0^+$, which denotes the number of job types using site s, Tier-i. We also introduce the binary variable $x_j^{si} \in \{0, 1\}$ denoting whether or not job j is using site s, Tier-i. The following constraints are added to the model:

$$f_j^{si} \leq x_j^{si} \quad \forall j \in J, \forall s \in S, \forall i \in I$$
$$\sum_{j \in J} x_j^{si} - 1 \leq x_s^i \qquad \forall s \in S, \forall i \in I$$

The first constraint says that if a job type is placed on site s, Tier-i, then the variable x_j^{si} must be set. The next constraint says that x_s^i is set to the number of job types minus one using site s, Tier-i. The first job type using the site and Tier is "free", because we wish to assign exactly one (or as few as possible) VO(s) to each site.

The objective is changed such that a penalty is added for each extra VO (job type) using a site and Tier. Let M be the large penalty. The objective is:

$$\min \quad \sum_{s \in S} \sum_{i \in I} M x_s^i + B$$

Adding these constraints and the new objective function results in an $\mathcal{NP}$-hard problem; this can be seen by reduction from the two-partitioning problem, see Garey and Johnson [89]. The problem of minimizing the number of VOs per site was solved very quickly, though; CPLEX found an optimal solution in less than a second.

Results of optimizing VO distribution are seen in Table 7.7 for job placement and in Table 7.8 for network usage. The maximal network link load is reduced from 4.4 Gbps to 3.7 Gbps. The link between the central NDGF router and Sweden still carries the larger network load, which is as previously described due to Sweden's larger compute and storage capabilities. The maximal network link usage is reduced because the original job placement was not optimal. Comparing with the optimal job placement in Section 7.4.1, the maximal network link load is actually increased from 3.5 Gbps to 3.7 Gbps. This increase is caused by the extra constraint on placing jobs according to VOs.

Many VOs take interest in limiting the number of VOs using each site. The reason for this is partly to get a better overview of activity in the grid; another reason is to avoid competing for resources with other VOs. This analysis shows that if workloads were to be distributed according to VOs, then the VOs must accept a decrease in the utilization of the system.

Tier-1	**ALICE**	**ATLAS**	**CMS**	**Total**
DK		480		480
FI	280			280
NO		758		758
SE	1377	333		1710
SL				0
Sum	1657	1571		

Tier-2	**ALICE**	**ATLAS**	**CMS**	**Total**
DK				0
FI			666	666
NO		325		325
SE	624	925		1549
SL		450		450
Sum	624	1700	666	2990

Total	**ALICE**	**ATLAS**	**CMS**	**Total**
	2281	3271	666	6218

Table 7.7: The result of placing jobs according to Virtual Organizations. The table shows the job placement. Comparing to the initial job placement in Table 7.2 it is noted that all job types are fully executed, but the number of different job types at each site is minimized.

Site/Country	Network load
DCSC/KU	1.8 Gbps
Denmark	1.8 Gbps
CSC	1.0 Gbps
Jyv	0.0 Gbps
Finland	1.0 Gbps
UiB	1.4 Gbps
UiO	1.3 Gbps
Norway	2.6 Gbps
HPC2N	1.9 Gbps
LUNARC	0.6 Gbps
PDC	2.1 Gbps
NSC	1.9 Gbps
UPPMAX	0.6 Gbps
Sweden	**3.7 Gbps**
PIKOLIT	0.1 Gbps
Slovenia	0.1 Gbps

Table 7.8: The result of placing jobs according to Virtual Organizations. The table shows network usage.

7.4.3 Cache considerations

In this section we analyze the effects of adding caches to the network topology. So far time has not been part of the model or the analyses. However, when working with caches the time dimension is important. No information is available on the order of job execution. Hence we consider a discrete time representation and in each time slot, the job mix given in e.g. Section 7.3 is calculated. When deciding what to store on a cache, we assume that the jobs in each time iteration are executed at the same time. Cache replacement strategies such as ”first in first out”, ”most used”, ”least used”, ”last recently used”, etc. are irrelevant with our time representation; we assume that the cache contents never change.

This strategy does not resemble the caching mechanism in the ARC grid middleware, because of the simplified time representation. As mentioned previously in Section 7.2.1, we do not consider ARC caching. However, we hope that this simpler caching approach gives a good picture of any potential effect on network usage.

First we consider how network usage is changed, when we add caches to the current job placement from Section 7.3. This is followed by adding caches to the optimal job placement, see Section 7.4.1.

7.4.3.1 Adding caches to current topology

Assume that the first iteration of job execution in the NDGF system is based on the job placement and data distribution illustrated in Section 7.3. Caches of a certain size are added to each site and data traffic is then re-calculated according to job placement, data distribution and data stored on caches. The contents of the caches are assumed to be such that the maximal network link load is minimized. Specifically, when $\mathfrak{C}_D^s$ and $\mathfrak{C}_T^s$ are the amount of cache on site s for disk and tape, respectively, bandwidth requirements for input and output data are calculated as:

$$BI_C^s = DI_C^s \frac{D - D^s - \mathfrak{C}_D^s}{D} + TI_C^s \frac{T - T^s - \mathfrak{C}_T^s}{T}$$

$$BO_C^s = DO_C^s \frac{D - D^s - \mathfrak{C}_D^s}{D} + TO_C^s \frac{T - T^s - \mathfrak{C}_T^s}{T}$$

How big an impact caches have on network usage depends on the size of the caches. We assume that the caches each can hold 200 units of data. Introducing caches does not impose any new variables to the model, which hence can be solved in $\mathcal{O}(1)$ time.

The results of adding caches are seen in Table 7.9. The maximal network link load is reduced from 4.4 to 3.8 Gbps using cache. Generally, the reduction size depends on the

size of the cache: if all data is replicated and stored on all caches, the network usage would be minimized and would only consist of generated data from job computations.

Site/Country	Network load
DCSC/KU	1.5 Gbps
Denmark	1.5 Gbps
CSC	0.9 Gbps
Jyv	0.0 Gbps
Finland	0.9 Gbps
UiB	1.3 Gbps
UiO	1.0 Gbps
Norway	2.0 Gbps
HPC2N	1.7 Gbps
LUNARC	0.5 Gbps
PDC	2.1 Gbps
NSC	1.7 Gbps
UPPMAX	0.5 Gbps
Sweden	**3.8 Gbps**
PIKOLIT	0.1 Gbps
Slovenia	0.1 Gbps

Table 7.9: The resulting bandwidth usage when adding caches to the initial job placement and data distribution from Table 7.2 and 7.3.

7.4.3.2 Changing job execution according to cache

Assuming that the caches are included in the network, we change job placement such that the maximum network link load is minimized. The modification of bandwidth requirements was illustrated in the previous section. The modification of job placement was illustrated in Section 7.4.1. The resulting problem is polynomial; adding caches to the model changes the calculations of bandwidth usage without imposing new variables and optimizing job placement introduces linear variables and constraints to the model.

The results of adding caches and optimizing job placement can be seen in Table 7.10 for job placement and in Table 7.11 for bandwidth usage. The maximal network link load is 3.0 Gbps between the central NDGF router and Sweden. Comparing with the optimal job placement without cache, the link load is reduced from 3.5 to 3.0 Gbps by using the caching mechanism. Again the reduction of network traffic depends on the size of the caches. With the 200 data unit caches, the reduction of the maximal link load is significant. This indicates that it is important for the model to include the ARC caching mechanism in order to reflect real-life network usage. It also indicates that investing in caches may be a relatively inexpensive way of ensuring that network traffic does not exceed network capacities.

Tier-1	**ALICE**	**ATLAS**	**CMS**	**Total**
DK		480		480
FI		280		280
NO	152	606		758
SE	1505	205		1710
SL				0
Sum	1657	1571		3228
Tier-2	**ALICE**	**ATLAS**	**CMS**	**Total**
DK				0
FI		666		666
NO		325		325
SE	624	709	216	1549
SL			450	450
Sum	624	1700	666	2990
Total	**ALICE**	**ATLAS**	**CMS**	**Total**
	2281	3271	666	6218

Table 7.10: The result of adding caches and then optimizing job placement. Data is distributed as seen in Table 7.3. The table illustrates job placement.

Site/Country	Network load
DCSC/KU	1.7 Gbps
Denmark	1.7 Gbps
CSC	0.9 Gbps
Jyv	0.0 Gbps
Finland	0.9 Gbps
UiB	1.3 Gbps
UiO	1.1 Gbps
Norway	2.3 Gbps
HPC2N	1.8 Gbps
LUNARC	0.5 Gbps
PDC	1.8 Gbps
NSC	1.8 Gbps
UPPMAX	0.5 Gbps
Sweden	**3.0 Gbps**
PIKOLIT	0.1 Gbps
Slovenia	0.1 Gbps

Table 7.11: The result of adding caches and then optimizing job placement. Data is distributed as seen in Table 7.3. The table shows the resulting bandwidth usage.

7.4.4 Changing data distribution and capacity

Changing the storage capacity of sites and re-arranging the data distribution may reduce the maximal network link load. The changes cannot be imposed immediately to the NDGF network because of changes to storage capacities. Even though the total amount of stored data in the system is not changed, it cannot be expected that a site is willing to move parts of its storage to another site. Hence this analysis is performed as a strategic tool to measure optimal disk requirements for the job placement of Section 7.3. This corresponds to transforming $D^s \geq 0$ and $T^s \geq 0$ (data and tape stored at each site s) into variables in the mathematical model, such that all data is distributed:

$$\sum_{s \in S} D^s = D$$
$$\sum_{s \in S} T^s = T$$

Only linear variables and constraints are added to the model; hence the resulting problem is polynomial.

The result of changing the data and storage distribution is seen in Table 7.12 for data distribution and Table 7.13 for network usage. The maximal network link load is reduced from 4.4 Gbps to 3.7 Gbps. The maximal network load can still be found on the link between the central NDGF router and Sweden. The optimal data placement does not distribute data more evenly across sites. Sweden, for example, has a large amount of compute resources but no data stored. In this way, the link from the central NDGF router to Sweden only needs to carry data required and generated by jobs executed on the Swedish sites. Not storing any data in Sweden is not realistic; from the results we can conclude that both compute and storage resources should be evenly distributed - it is not beneficial to only distribute storage evenly.

7.4.5 Changing job placement and compute limits

This analysis is used in a strategic context and considers the effects of removing the limits on the amount of compute resources at each site. The total compute requirement in the system is not changed, only the placement of compute resources is altered. As in the case for moving storage resources, it is not immediately possible to move compute resources from one site to another, hence the strategic nature of the analysis. The placement of jobs is changed such that the maximal network link load is minimized. The data distribution described in Section 7.3 remains unchanged. We transform $0 \leq f_j^{si} \leq 1$ (percentage job j takes up compute resources at site s, Tier-i) into a variable. Furthermore, we ensure that each job is fully executed, but set no upper bound on

Disk Tier-1	ALICE	ATLAS	CMS	Total
DK		473		473
FI	174	543		717
NO	634			624
SE				0
SL				0
Sum	808	1016	0	1824

Disk Tier-2	ALICE	ATLAS	CMS	Total
DK				0
FI	148	745	205	1098
NO				0
SE				0
SL	148	745	205	1098
Sum	956	1761	205	2922

Tape	ALICE	ATLAS	CMS	Total
DK				0
FI				0
NO	1105	795		1900
SE				0
SL				0
Sum	1105	795	0	1900

Total	2281	3271	666	6218

Table 7.12: The result of changing the distribution of data placement and of storage capacities. Jobs are placed as seen in Table 7.2. The table illustrates data distribution.

Site/Country	Network load
DCSC/KU	2.1 Gbps
Denmark	2.1 Gbps
CSC	2.5 Gbps
Jyv	0.0 Gbps
Finland	2.5 Gbps
UiB	3.2 Gbps
UiO	1.5 Gbps
Norway	3.5 Gbps
HPC2N	0.8 Gbps
LUNARC	0.6 Gbps
PDC	1.2 Gbps
NSC	0.8 Gbps
UPPMAX	0.6 Gbps
Sweden	**3.7** Gbps
PIKOLIT	0.1 Gbps
Slovenia	0.1 Gbps

Table 7.13: The result of changing the distribution of data placement and of storage capacities. Jobs are placed as seen in Table 7.2. The table illustrates network usage.

compute resources at each site:

$$\sum_{s \in S} C^{si} f_j^{si} = R_j \quad \forall i \in I, \forall j \in J$$

All added variables and resulting constraints are linear; hence the resulting problem can be solved in polynomial time.

Results of changing job placement and of changing compute capacities can be seen in Table 7.14 for job placement and in Table 7.15 for network usage. The maximal network link load is reduced from 4.4 Gbps to 3.5 Gbps compared to the initial job placement in Section 7.3. Compared to the optimal job placement in Section 7.4.1, the maximal network link load has not been reduced. Hence, changing the distribution of compute resources does not reduce network usage in the NDGF system.

7.5 Conclusion

In this paper, we have formalized network usage in the distributed Nordic Tier-1 operated by the Nordic DataGrid Facility (NDGF) into a mathematical model. Using the model we have calculated network usage subject to job placements, data requirements and network capacities. The model and the results can only be considered as a first approximation of what kind of network load NDGF can expect. Some of the assumptions

Tier-1	ALICE	ATLAS	CMS	Total
DK		474		474
FI				0
NO				0
SE	1657	1095		2754
SL				0
Sum	1657	1571		

Tier-2	ALICE	ATLAS	CMS	Total
DK				0
FI				0
NO				
SE		1700	666	2366
SL	624			624
Sum	624	1700	666	2990
Total	2281	3271	666	6218

Table 7.14: The result of changing the job placements and compute capacities. Data is distributed as seen in Table 7.3. The table illustrates job placement.

Site/Country	Network load
DCSC/KU	1.8 Gbps
Denmark	1.8 Gbps
CSC	0.4 Gbps
Jyv	0.0 Gbps
Finland	0.4 Gbps
UiB	1.1 Gbps
UiO	0.6 Gbps
Norway	1.6 Gbps
HPC2N	2.0 Gbps
LUNARC	0.2 Gbps
PDC	2.1 Gbps
NSC	2.0 Gbps
UPPMAX	0.2 Gbps
Sweden	**3.5 Gbps**
PIKOLIT	1.7 Gbps
Slovenia	1.7 Gbps

Table 7.15: The result of changing the job placements and compute capacities. Data is distributed as seen in Table 7.3. The table illustrates network usage.

behind the model can rightfully be criticized for being too simple. Especially the assumption on the uniform distribution of job types over time is questionable. One way to deal with that assumption would be to only consider the job type that causes the highest network load. This would make the model a better fit for worst case loads. However, it will be more important to take the caching mechanism of ARC into account, as this mechanism has been reported to have a significant impact on how many times popular files are downloaded to a site. In order to extend the model to take caching into account, a more in-depth analysis of the caching mechanism of ARC needs to be performed first.

Though the model may be simplified, it still provides us with the possibility of analyzing the effects of changes to the system. We showed that it is highly beneficial to consider job placement carefully, because network requirements heavily depend on this. The users of the system (denoted Virtual Organizations (VOs)) take interest in dividing workloads on the system according to their job mixes in order for them to more easily get an overview of activity in the NDGF system. We showed that placing jobs according to VOs most likely increases network requirements, thus this placement strategy is not attractive for the overall system. As previously mentioned, the simplified model does not consider the ARC caching mechanism. We tried to compensate for this by introducing a very simple cache strategy, which reduces network traffic significantly. Finally, we used the model to make more strategic analyses of the network, i.e., we investigated the effects of changing storage and compute capacities in the system. Network usage is lowered when data storage is optimized according to a given job placement, while changing the compute capacities does not reduce the maximal link load.

The model has been modified to reflect the mentioned analyses of the NDGF system, and the modifications generally increase the complexity of the problem. In most cases the resulting problem is polynomial, so the problems scale well and can be solved for much larger systems than the NDGF system. Placing jobs according to VOs resulted in an $\mathcal{NP}$-hard problem, which was solved very quickly, though. We believe that the problem is practically tractable even for larger instances. Based on this we conclude that the model can be used as foundation for the development of general strategic tools for grid systems. It is noted that if we wish to find the optimal job placement *and* data distribution, then the model becomes quadratic and possibly more difficult to solve.

In the introduction of this work, we mentioned that parts of the network used by the NDGF system were shared with other users. This was illustrated as the magenta lines in Figure 7.2. Future analyses on the NDGF system could be to decrease the amount of traffic on the public links. If NDGF wants to maintain the right to use these links, NDGF must ensure that bandwidth usage on the links never blocks out other users.

At the time of writing it is not possible to compare the complete model to real world measurements. However, NDGF system administrators have noted some differences between the model and actual grid behaviour:

- Currently, the main bottleneck of the system is actually the lack of bandwidth between the compute element and the network. This issue is solvable by upgrading the hardware at sites.
- The assumption about a uniform mix of jobs is not always correct. We observe that jobs of certain types come in bursts.
- The ARC caching mechanism has a dramatic effect on the amount of data transferred. This is not surprising considering our calculations on including a simplified caching strategy.

NDGF is currently investigating the deployment of monitoring services to measure bandwidth usage. With such measurements we can test the validity of our model. Once the model has been compared to actual network usage and possibly calibrated to reflect this, calculations of optimal job placements and of distribution of storage and CPU capacities will be re-performed in order to ensure an optimal utilization of the distributed Nordic Tier-1.

Part III

The multicommodity *k*-splittable flow problem

CHAPTER 8

Introduction to the multicommodity k-splittable flow problem

The problem of data transmission in a network can be represented as a multicommodity flow problem (MCFP). Many variants of the MCFP exist in order to reflect corresponding real-life telecommunication problems. Examples are that each data transmission can only use a certain number of paths, each path can only consist of a certain number of edges, the amount of data sent through the network must be maximized, the cost of sending data through the network must be minimized, etc. A variety of multicommodity flow problems arising in telecommunication context are presented and discussed in the book by Resende and Pardalos [166].

The multicommodity k-splittable flow problem (MCkFP) represents the Multiprotocol Label Switching problem, which limits the size of routing tables by gathering data packets under a label [67]. The MCkFP is $\mathcal{NP}$-hard and consists of routing all commodities through a network such that each commodity uses at most k paths. Edges in the network are capacitated and all edge capacities must be satisfied. Traditionally two variants of MCFP problems are considered: minimizing the total cost of sending all commodities or maximizing the total amount of flow sent through the network. This is also the variants considered for the MCkFP in this part.

This chapter is organized as follows. Section 8.1 describes the multicommodity k-splittable flow problem. This is followed by motivating the existence of the problem by giving real-life examples in Section 8.2. An overview of work in the literature performed on the multicommodity flow problem and especially on k-splittable flow problems is presented in Section 8.3. The contribution of this thesis is described in Section 8.4 and finally we discuss future directions on work on multicommodity k-splittable flow problems in Section 8.5.

8.1 Problem description

The family of multicommodity flow problems belongs to the group of network flow problems. In graph theory, a flow network is a directed graph consisting of nodes and capacitated edges, where flow travels on edges without exceeding edge capacities and where the amount of flow going into a node equals the amount of outgoing flow. Flow is routed from a start node (source) to an end node (destination). The objective is typically to maximize the amount of flow routed through the graph or to minimize the cost of sending a fixed amount of flow through the graph. In the case of the latter objective, a cost per flow unit is attached to each edge of the graph. The described network flow problems are polynomially solvable.

In multicommodity flow problems (MCFP) several flows (or commodities) must be routed through the network. Each commodity consists of a source and a destination and possibly also a fixed amount of flow to route. It is assumed that at least two commodities do not share both source and destination node, because otherwise the problem reduces to a regular network flow problem. MCFP is polynomial, but adding extra (practically relevant) constraints may make the problem $\mathcal{NP}$-hard. Such variants include the multicommodity unsplittable flow problem (MCuFP), where each commodity must use exactly one path to send its flow from its source to its destination. Another version is the multicommodity k-splittable flow problem (MCkFP), where each commodity can use at most k paths to send its flow from its source to its destination.

A flow network is illustrated in Figure 8.1. Costs per flow unit and capacities are given at each edge. Two commodities are to be sent: 4 flow units from s_1 to t_1 via at most 2 paths and 3 flow units from s_2 to t_2 via at most 3 paths.

Table 8.1 shows all possible paths and their costs for the two commodities. The columns `Comm. 1` and `Comm. 2` report optimal solutions for commodity 1 alone and for commodity 2 alone, respectively. Columns `Comm. 1` $\rightarrow$ `Comm. 2` and `Comm. 2` $\rightarrow$ `Comm. 1` are optimal solutions if commodity 1 has highest priority and if commodity 2 has highest priority, respectively. Finally, in column `Optimal` an overall optimal solution is given. As illustrated there is no connection between the

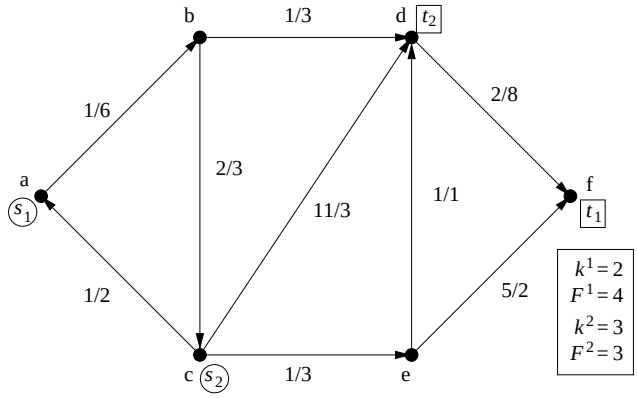


Figure 8.1: Example of a flow network. Two commodities, 1 and 2 are to be sent through the network. The number of flow units to ship for each commodity is denoted F^i, $i \in \{1, 2\}$, and the maximal number of paths to use for each commodity is denoted k^i, $i \in \{1, 2\}$. The cost per flow unit and capacity, respectively, are given at each edge. The objective is to minimize the cost of shipping the commodities.

costs of sending either of the commodities individually and the overall optimal solution. The example shows that both commodities must be taken into account at the same time when finding an optimal solution, which is a consequence of the problem being $\mathcal{NP}$-hard.

		Flow				
Path	Cost	Comm. 1	Comm. 2	Comm. 1 → Comm. 2	Comm. 2 → Comm. 1	Optimal
a-b-d-f	4	3	-	3	1	2
a-b-c-d-f	16	0	-	0	1	0
a-b-c-e-d-f	7	1	-	1	0	0
a-b-c-e-f	9	0	-	0	2	2
c-a-b-d	3	-	2	0	2	1
c-d	11	-	0	3	0	1
c-e-d	2	-	1	0	1	1
	Cost	19	8	52	46	42

Table 8.1: Overview of paths and costs for the commodities in Figure 8.1

8.2 Motivation

Flow problems have wide applications in many logistical problems, where commodities must be routed through a network. This includes traffic modeling in a street or railway network, currency regulation in electrical circuits, distribution of water in pipes, data packets in a network, etc.

An application for the MCkFP is *Multiprotocol Label Switching* (MPLS), which gathers several data packets under a label in order to limit the routing tables and to increase the quality of data transmission. Also, encapsulating packets of different network protocols and only making forwarding decisions based on the labels, eliminates the need for the network to support several data link layer technologies. For more details on MPLS, we refer to the book of Evans and Filsfils [67]. The cost of sending data increases with the number of *Label Switch Paths* (LSP). By limiting the number of used labels (i.e. paths) the total cost can be reduced. However, we must still ensure that all or as much data as possible is transmitted. This corresponds to the MCkFP; given an upper bound on the number of paths to use, we either try to send all data at the lowest possible cost (minimum cost MCkFP) or we try to maximize the total throughput in the network (maximum flow MCkFP).

Another application for the MCkFP is the transportation of goods e.g. via trains, where the number of locomotives is limited. Assuming that each storage has k locomotives available for sending its goods to a destination, then at most k different routes can be used. This corresponds to the MCkFP, where the objective is either to send all ordered goods at the lowest possible cost (minimum cost MCkFP) or to maximize the total amount of goods to send (maximum flow MCkFP).

Though we have not applied the MCkFP in a grid scheduling context, the flow problem is still highly relevant in telecommunications and data transmission problems. The MCkFP is applicable to the MPLS protocol, which could very well be used for data transmission in the grid network. The MCkFP can be used for finding appropriate network routes for data connections.

8.3 Historical overview

The MCFP is polynomial, but to the best of our knowledge no straight-forward combinatorial algorithm is currently known for the problem, see Cormen et al. [52]. Instead, solution methods for the MCFP from the literature include:

- Cost based decomposition
- Resource based decomposition
- Interior point methods

Lagrange relaxation is an example of cost based decomposition, where constraints ensuring capacitated flow transmission are multiplied with a Lagrange multiplicator and moved to the objective function. In this way, MCFP can be divided into separate flow

problems for each commodity; see Wolsey [204]. Other cost based decompositions include Dantzig-Wolfe decomposition, see Dantzig and Wolfe [54]. Barnhart et al. [26] presented a decomposition where the pricing problem generated paths for each commodity and the master problem merged paths into an overall solution.

Resource based decomposition focuses on omitting edge capacities by introducing resources. Resources are attached to each edge and the total resource consumption for each commodity is bounded from above. The total commodity resource consumption on each edge is smaller or equal to the corresponding edge capacity. In this way each commodity can be considered individually as resource constrained flow problems, which again can be solved using a subgradient method, see Ahuja et al. [5].

MCFP can be solved using interior point algorithm. The maximum flow MCFP was formulated as a quadratic problem by Kamath and Palmon [116] who used two interior point algorithms. The algorithms are also capable of solving the minimum cost MCFP.

Baier et al. [19] introduced the Multicommodity k-splittable Flow Problem and proved that the problem is $\mathcal{NP}$-hard in the strong sense for directed graphs, even in the singlecommodity case. They presented approximation algorithms for the single- and multicommodity versions, specifically the maximum budget-constrained k-splittable Flow Problem. The authors noted that if k is greater than or equal to the number of edges, then the k-splittable MCFP degenerates to an ordinary MCFP.

Koch et al. [122] proved that the maximum flow MCkFP is $\mathcal{NP}$-hard in the strong sense for directed and undirected graphs. They also showed that no approximation algorithm exists which is better than $\frac{5}{6}$, unless $\mathcal{P} = \mathcal{NP}$. Koch et al. [123] also presented a two-stage algorithm for the MCkFP. The first stage is routing, i.e., deciding which k paths to use for each commodity, and the second stage consists of packing, i.e., on assigning flow on the paths. They argued that when k is constant, then the packing alternatives can be constructed in polynomial time, and when k is part of the input they present an algorithm with approximation factor $(1 - \epsilon)$, $\epsilon > 0$.

Truffot et al. [190] presented a branch-and-price algorithm for solving the maximum flow MCkFP to optimality. An edge-path model is presented, on which a branch-and-price algorithm is applied. The pricing problem is a shortest path problem, which generates paths for each commodity and which can be solved in polynomial time. The master problem merges the paths into an overall feasible solution.

8.4 Contribution

This part of the thesis considers the MCkFP variant of the multicommodity flow problem, where each commodity can use at most k paths to route its flow. The two contributed papers are:

- Two- and three-index formulations of the multicommodity k-splittable flow problem
- Comparing branch-and-price algorithms for the multi-commodity *k*-splittable maximum flow problem

The first paper considers the minimum cost MCkFP, specifically the three-index model and corresponding branch-and-price algorithm for the MCkFP suggested by Truffot et al. [190]. The three indices indicate a commodity, a path and a path index, i.e., which of the k paths we wish to consider. The paper proposes a heuristic for the three-index branch-and-price algorithm, which tries to merge certain paths when more than k paths are used for a commodity. The heuristic boosts the performance of the algorithm. The main contribution of the paper, however, is a two-index problem formulation and a corresponding branch-and-price algorithm. The two indices indicate a commodity and a path, respectively. The algorithm includes a somewhat complicated branching strategy, which in worst case causes a large search tree due to many branching combinations and due to many branching children. However, the two-index model eliminates large amounts of symmetry in the solution space and hence the branch-and-price algorithm outperforms the three-index algorithm.

The second paper on the MCkFP considers the maximum flow version of the problem. The work from the former paper is applied to the maximum flow version, i.e., the heuristic is added to a three-index branch-and-price algorithm and the two-index branch-and-price algorithm is slightly altered to fit the maximum flow objective function. The two-index branch-and-price algorithm has less impressive performance when maximizing flow, because the objective causes an increase in the number of branching combinations. Hence, we propose a new two-index branch-and-price algorithm, where the branching strategy consists of forcing and forbidding the usage of certain paths. Forcing the use of a path is done by adding cuts to the model. We also forbid the usage of certain paths in order to eliminate symmetry in the solution space. This new branching strategy improves the performance of the two-index algorithm dramatically, hence making it superior to the exact algorithms from the literature.

8.5 Future directions

In this section we focus on future work to be performed on the MCkFP when using Dantzig-Wolfe decomposition.

Our work shows that especially two bottlenecks should be considered in the proposed branch-and-price algorithms for the MCkFP: the branching strategy and how to bound the number of used paths. Future work on branching strategies could focus on reducing the impact on the pricing problem and on producing a smaller search tree. The latter is reached by generating fewer branching children and on providing stronger bounds in each branching child. We believe that adding branching cuts is a strategy worth exploring even further.

The second bottleneck of the branch-and-price algorithms is the way the number of used paths is bounded. Tightening the formulation would improve the performance of the algorithms significantly. Adding cuts to the relaxed master formulation is not trivial, because the variables of the model are linear. However, cuts could be used to strengthening the path bound. We believe it could be interesting to find cuts on the binary variables in the original edge formulation or in the (non-relaxed) master problem formulation. These cuts could then either be transferred into working on the LP-relaxed master problem or some binary variables from the original model could be kept in the LP-relaxed master problem. Desaulniers et al. [57] show how cuts on original formulations can be used in a Dantzig-Wolfe decomposition context.

Another approach is to somehow decompose the problem differently. The decomposition should be based on a strong formulation. Furthermore, it should not cause pricing and branching to be too difficult in order for the decomposition to be beneficial.

CHAPTER 9

Two- and three-index formulations of the minimum cost multicommodity *k*-splittable flow problem

Mette Gamst
DIKU, Department of Computer Science,Copenhagen University, Universitetsparken 1, DK-2100 Copenhagen Ø, Denmark

Peter Neergaard Jensen
DIKU, Department of Computer Science,Copenhagen University, Universitetsparken 1, DK-2100 Copenhagen Ø, Denmark

David Pisinger
DIKU, Department of Computer Science,Copenhagen University, Universitetsparken 1, DK-2100 Copenhagen Ø, Denmark

Published in European Journal of Operational Research 202: 82-89, 2010

Christian Plum
DIKU, Department of Computer Science,Copenhagen University,
Universitetsparken 1, DK-2100 Copenhagen Ø, Denmark

Abstract
The Multicommodity Flow Problem (MCFP) considers the efficient routing of commodities from their origins to their destinations subject to capacity restrictions and edge costs. Baier et al. [19] introduced the Maximum-flow Multicommodity k-*splittable* Flow Problem (MCkFP) where each commodity may use at most *k* paths between its origin and its destination. This paper studies the $\mathcal{NP}$-hard Minimum Cost Multicommodity k-splittable Flow Problem (MCMCkFP) in which a given flow of commodities has to be satisfied at the lowest possible cost. The problem has applications in transportation problems, where a number of commodities must be routed using a limited number of distinct transportation units for each commodity. Based on a three-index formulation by Truffot et al. [191], we present a new two-index formulation for the problem and solve both formulations through branch-and-price. The three-index algorithm by Truffot et al. is improved by introducing a simple heuristic method to reach a feasible solution by eliminating some symmetry. A novel branching strategy for the two-index formulation is presented, forbidding subpaths in the branching children. Though the proposed heuristic for the three-index algorithm improves its performance, the three-index algorithm is still outperformed by the two-index algorithm, both with respect to running time and to the number of solved test instances.

Key words: Network flows; Transportation; Decomposition; Multicommodity flow;

9.1 Introduction

Various variants of the Multicommodity Flow Problem (MCFP) have been considered. In an MCFP we are given a network $G = (V, E)$ where each edge has a certain capacity and possibly an associated cost. Furthermore there is a set of commodities each of which has to be sent from a starting vertex to an ending vertex. Typically the goal is one of the following:

1. (*Minimum cost*) A given flow for each commodity is to be routed through the network. The goal is to minimize the cost of sending all commodities through the network.
2. (*Maximum flow*) The goal is to maximize the flow through the network, i.e., there is no fixed flow demand for the commodities. Edge costs are unimportant because the cost of sending flow is not taken into account.

The MCFP can be solved in polynomial time, see e.g. [52]. Often, however, there are extra conditions that have to be satisfied, making the problem $\mathcal{NP}$-hard. An example of such a condition is an upper bound on the length of the paths used to ship the flow. This is relevant in telecommunication networks where the path must not be too long as this may lead to delays. The length-bounded flow problem is $\mathcal{NP}$-hard even for a single commodity [99]. Another condition could be that all flow for each commodity must be sent via just one path. This type of problem, which is often denoted the *Unsplittable* MCFP, is introduced and proven $\mathcal{NP}$-hard by Kleinberg [121]. Yet another condition is an upper bound on the number of paths used by a commodity. This is called the Multicommodity k-splittable Flow Problem (MCkFP). We consider the Minimum Cost MCkFP (MCMCkFP), which for instance appears in the transportation sector where a number of different commodities have to be dispatched to various destinations at the lowest possible cost. For safety reasons, it is not desirable to divide the commodities into more than k routes.

Robacker [168] considered the flow maximization version of the MCFP. He describes a decomposition of the problem which he hopes may lead to combinatorial methods for solving the multicommodity problem. Ford and Fulkerson [73] suggested a variant of the simplex method, based on column generation, where each simplex step consists in finding a shortest path. This is the forerunner of the general Dantzig-Wolfe decomposition procedure [54]. Kamath and Palmon [116] formulated the Maximum MCFP as a quadratic problem. They solved the problem in polynomial time using an interior point algorithm. Their results also apply to the Minimum Cost MCFP.

Barnhart et al. [26] considered the Minimum Cost Unsplittable MCFP. They presented a branch-and-cut-and-price algorithm in which they used column generation to find bounds in the branch-and-bound tree, and they proposed a new branching rule allowing new columns to be generated effectively. They concluded that their cuts only work for problems where the commodity flow is large compared to the edge capacities.

The Multicommodity k-splittable Flow Problem (MCkFP) was introduced by Baier et al. [19] who presented approximation algorithms for the Single- and Multicommodity k-splittable Flow Problems, specifically, variants of the maximum flow problem including the maximum budget-constrained k-splittable Flow Problem. The authors proved that the Maximum Single-commodity k-splittable Flow Problem is $\mathcal{NP}$-hard in the strong sense for directed graphs. Finally, they note that for $k \geq |E|$, a k-splittable (s,t) flow problem degenerates to an ordinary (s,t) flow problem.

Koch et al. [123] proved that the Maximum MCkFP is $\mathcal{NP}$-hard in the strong sense for directed as well as undirected graphs. They also showed that, unless $\mathcal{P} = \mathcal{NP}$, no approximation algorithm exists which is better than $\frac{5}{6}$. In a later paper, Koch et al. [122] consider the Maximum MCkFP as a two-stage problem consisting of the decision on the k paths (routing) and on the amount of flow on the paths (packing). If

k is a constant, they show that it suffices to consider a polynomial number of packing alternatives, which can be constructed in polynomial time. If k is part of the input, Koch et al. propose an approximation algorithm with approximation factor $(1 - \epsilon)$, $\epsilon > 0$.

Truffot et al. [192] used branch-and-price to solve the Maximum MCkFP. An edge-path model was presented to which a branch-and-price algorithm was applied. The subproblem for the column generation is a shortest path problem solvable in polynomial time. More recent work on this problem is seen in [190], where Truffot and Duhamel also presented a two-index model for the Maximum MCkFP, but they concluded that the two-index model cannot be used in an efficient way in a branch-and-bound scheme. Truffot et al. [191] also introduced the minimum cost MCkFP. A three-index model for the problem was solved using a branch-and-price algorithm. The algorithm is closely related to the one presented in [192] and [190].

The Minimum Cost MCkFP can be represented by a directed graph $G = (V, E)$, where V is the set of vertices and E the set of edges. Each edge $e \in E$ has a nonnegative cost c_e and a positive capacity u_e attached. The edge capacities are positive since any edge with zero capacity can be removed from the graph. The set of commodities is denoted L and each commodity $l \in L$ has a source s_l, a destination t_l, an amount to be shipped F^l, and a maximal number of routes the commodity may use k^l.

Baier et al. [19] and Koch et al. [123] showed that the maximum single-commodity k-splittable flow problem is $\mathcal{NP}$-hard in directed and undirected networks, respectively. As a consequence, it is $\mathcal{NP}$-hard even to decide whether an instance of the minimum cost single-commodity k-splittable flow problem has a feasible solution.

In this paper, we compare various formulations of the Minimum Cost MCkFP when solved through branch-and-price. A two-index formulation is presented and it is compared to the three-index model by Truffot et al. [192]. The two-index model is based on the work of Barnhart et al. [26] where the formulation is changed from unsplittable to k-splittable. The model was introduced for the Minimum Cost MCkFP by Gamst et al. [87] and for the Maximum MCkFP by Truffot and Duhamel [190].

The main contribution of this paper is the branch-and-price algorithm for the two-index model. The algorithm consists of a sophisticated branching strategy and a pricing problem which handles restrictions imposed by branching. Furthermore, we introduce a heuristic for the three-index model of Truffot et al. [191] which improves the performance of the three-index algorithm. Despite the improvement, however, the two-index algorithm outperforms the three-index model. The three-index algorithm is capable of solving instances with up to 1085 commodities, 400 nodes, and 1520 edges. The two-index algorithm solves instances with up to 2239 commodities, 400 nodes, and 1520

edges.

The paper is organized as follows: Section 9.2 contains the three-index mathematical formulation of Truffot et al. [191] and the corresponding branch-and-price solution approach. We present a heuristic to speed up the solution process in this section. In Section 9.3 we introduce the two-index mathematical formulation and solve it through branch-and-price. Both algorithms are tested and compared in Section 9.4 showing that the three-index algorithm is outperformed by the two-index algorithm, both with respect to running time and to the number of solved test instances. Section 9.5 concludes the paper.

9.2 Three-index model

The three-index model for the MCMCkFP was introduced by Truffot et al. [191]. Let P^l be the set of possible paths for commodity l. The variable x_p^{hl} denotes the amount of flow on path p for the h'th path of commodity l. The binary variable y_p^{hl} decides whether path p for the h'th path of commodity l is to be used or not. The model *(MIP3)* is:

$$\min \quad \sum_{l \in L} \sum_{h=1}^{k^l} \sum_{p \in P^l} c_p x_p^{hl}$$

$$\text{s.t.} \quad \sum_{l \in L} \sum_{h=1}^{k^l} \sum_{p \in P^l} \delta_e^p x_p^{hl} \leq u_e \qquad \forall e \in E \tag{9.1}$$

$$x_p^{hl} - u_p y_p^{hl} \leq 0 \qquad \forall l \in L, h = 1, \ldots, k^l, \forall p \in P^l \tag{9.2}$$

$$\sum_{p \in P^l} y_p^{hl} \leq 1 \qquad \forall l \in L, h = 1, \ldots, k^l \tag{9.3}$$

$$\sum_{h=1}^{k^l} \sum_{p \in P^l} x_p^{hl} \geq F^l \qquad \forall l \in L \tag{9.4}$$

$$x_p^{hl} \geq 0 \qquad \forall l \in L, h = 1, \ldots, k^l, \forall p \in P^l$$

$$y_p^{hl} \in \{0, 1\} \qquad \forall l \in L, h = 1, \ldots, k^l, \forall p \in P^l$$

The objective function minimizes the total cost. The cost c_p of a path $p \in P^l$ is defined as the sum of edge costs c_e on the path. Constraint (9.1) is a capacity constraint, in which δ_e^p indicates whether or not edge e is used by path p. In (9.2), u_p denotes the capacity constraint on path p which is defined as $u_p = min\{u_e \,|\, e \in p\}$, hence (9.2) forces the decision variable y_p^{hl} to be set if there is flow on the corresponding path x_p^{hl}.

Constraint (9.3) ensures that at most one path is used as the h'th path of a commodity l, and finally (9.4) ensures that all commodities are shipped.

The model is relaxed into an LP-model: first the binary variables y_p^{hl} are LP-relaxed to $0 \leq y_p^{hl} \leq 1$. From (9.2) and (9.3) we are given, that $x_p^{hl}/u_p \leq y_p^{hl} \leq 1,\ u_p > 0$. Setting $y_p^{hl} = x_p^{hl}/u_p$, does thus not violate any constraints, instead the formulation is simplified to only consisting of one type of variables and constraint (9.2) is eliminated. The model *(LP3)* is:

$$
\begin{aligned}
\min \quad & \sum_{l \in L} \sum_{h=1}^{k^l} \sum_{p \in P^l} c_p x_p^{hl} \\
\text{s.t.} \quad & \sum_{l \in L} \sum_{h=1}^{k^l} \sum_{p \in P^l} \delta_e^p x_p^{hl} \leq u_e && \forall e \in E \\
& \sum_{p \in P^l} \frac{x_p^{hl}}{u_p} \leq 1 && \forall l \in L, h = 1, \ldots, k^l \\
& \sum_{h=1}^{k^l} \sum_{p \in P^l} x_p^{hl} \geq F^l && \forall l \in L \\
& x_p^{hl} \geq 0 && \forall l \in L, h = 1, \ldots, k^l, \forall p \in P^l
\end{aligned}
$$

Model *(MIP3)*, and thus also *(LP3)*, cause symmetry in the solution space as the h-index may result in equivalent solutions being treated as different solutions. For example, consider a commodity l which uses two paths p_1 and p_2. Now, the two solutions $x_{p_1}^{1l} = 1,\ x_{p_2}^{2l} = 2$ and $x_{p_2}^{1l} = 2,\ x_{p_1}^{2l} = 1$ are treated as different solutions though they use the same paths. To eliminate some of this symmetry, Truffot et al. use *variable ordering* by adding constraint (9.5) to the models *(MIP3)* and *(LP3)*:

$$
\sum_{p \in P^l} x_p^{(h+1)l} - \sum_{p \in P^l} x_p^{hl} \leq 0, \quad \forall l \in L, h = 1, \ldots, k^l - 1 \tag{9.5}
$$

However, (9.5) does not eliminate symmetry introduced by flow variables having the same amount of flow.

Pricing Problem

The pricing problem can be recognized as a shortest path problem. Let $\pi_e \leq 0$ correspond to the first constraint of the primal model, $\lambda^{hl} \leq 0$ to the second, $\sigma^l \geq 0$ to the third and $\omega^{hl} \leq 0$ to the symmetry constraint (9.5). Even though the primal model only consists of one variable type, the dual formulation has three constraints because

of the symmetry constraint (9.5). The reduced costs are:

$$\sum_{e\in E}\delta_e^p(c_e-\pi_e)-\frac{\lambda^{hl}}{u_p}+\sigma^l+\omega^{hl} \qquad \forall l\in L, h=1, \forall p\in P^l$$
$$\sum_{e\in E}\delta_e^p(c_e-\pi_e)-\frac{\lambda^{hl}}{u_p}+\sigma^l+\omega^{hl}-\omega^{(h-1)l} \qquad \forall l\in L, h=2,\dots k^l-1, \forall p\in P^l$$
$$\sum_{e\in E}\delta_e^p(c_e-\pi_e)-\frac{\lambda^{hl}}{u_p}+\sigma^l-\omega^{(h-1)l} \qquad \forall l\in L, h=k^l, \forall p\in P^l$$

For each pair of values (h,l) the task is to find a path $p\in P^l$ which has negative reduced cost. If the value for u_p is known in advance, the problem is a shortest path problem defined in costs $(c_e-\pi_e)\geq 0$, which can be solved in polynomial time using e.g. Dijkstra's algorithm [5]. Recall, that $u_p=\min\{u_e|e\in p\}$. That is, u_p can take on $\mathcal{O}(|E|)$ values; for each of the $\mathcal{O}(|E|)$ values of u_p the shortest path problem is solved on a graph, where edges with $u_e<u_p$ are removed.

Branching Strategy

The chosen branching scheme is closely related to that proposed by Barnhart et al. [26]. For the h'th path of commodity l, the strategy is based on dividing all edges $\phi^+(d_{hl})$ going out from *the first divergence* node d_{hl}, into two subsets. The first divergence node d_{hl} of a commodity l and path h is defined as the node to which all flow of the l'th commodity is following the same path and from which the flow is using two or more paths. The two resulting subsets of outgoing edges $\phi_1^+(d_{hl})$ and $\phi_2^+(d_{hl})$ are disjoint and balanced. Now, we use the dichotomic branching rule adding one of the following two constraints:

$$\left(\sum_{e\in\phi_1^+(d_{hl})}\delta_e^p x_p^{hl}=0\right) \qquad \left(\sum_{e\in\phi_2^+(d_{hl})}\delta_e^p x_p^{hl}=0\right)$$

Heuristic

To decrease the running time of the branch-and-price algorithm we suggest a simple heuristic method to reach a feasible solution by eliminating some symmetry. The model *(LP3)* can cause problems, as the constraint $x_p^{hl}/u_p\leq 1$ will not always be tight and hence may allow several paths to be used as the h'th path of commodity l. Also, the mathematical formulation *(LP3)* does not prevent the same path for a commodity from taking on several values of h. For these reasons, any of the two following situations may occur:

1: For a commodity, several identical paths are used but with different values of h.

2: More than one path is used for a single value of h for a commodity.

In the first case, we merge the paths into one. In the second case, each path is assigned a unique value of h, if possible. In this way a feasible solution may be reached faster.

9.3 Two-index model

In order to investigate how the h-indices affect the behavior of the branch-and-price algorithm, we have studied another path formulation of the MCMCkFP without the use of h-indices. The model *(MIP2)* is:

$$\begin{aligned}
\min \quad & \sum_{l \in L} \sum_{p \in P^l} c_p x_p^l & & \\
\text{s.t.} \quad & \sum_{l \in L} \sum_{p \in P^l} \delta_e^p x_p^l \leq u_e & \forall e \in E & (9.6)\\
& x_p^l - u_p y_p^l \leq 0 & \forall l \in L, \forall p \in P^l & (9.7)\\
& \sum_{p \in P^l} y_p^l \leq k^l & \forall l \in L & (9.8)\\
& \sum_{p \in P^l} x_p^l \geq F^l & \forall l \in L & (9.9)\\
& x_p^l \geq 0 & \forall l \in L, \forall p \in P^l & \\
& y_p^l \in \{0,1\} & \forall l \in L, \forall p \in P^l &
\end{aligned}$$

Here x_p^l is the total flow of commodity l on path p, and the corresponding variable y_p^l is set if and only if commodity l has flow on path p. The remaining variables have the same meaning as in the three-index model. The objective function minimizes the total cost of routing the commodities. Constraint (9.6) ensures edge capacities are never violated and constraint (9.7) forces the decision variable to take on value 1, whenever the amount of flow on the corresponding path is positive. Constraint (9.8) limits the number of used paths for commodity l to at most k^l and finally constraint (9.9) ensures that every commodity is shipped.

The problem is relaxed in the same manner as the three-index model, i.e., we replace

y_p^l with x_p^l/u_p getting *(LP2)*:

$$\begin{aligned}
\min \quad & \sum_{l\in L}\sum_{p\in P^l} c_p x_p^l \\
\text{s.t.} \quad & \sum_{l\in L}\sum_{p\in P^l} \delta_e^p x_p^l \le u_e \quad \forall e \in E && (9.10)\\
& \sum_{p\in P^l} \frac{x_p^l}{u_p} \le k^l \quad \forall l \in L && (9.11)\\
& \sum_{p\in P^l} x_p^l \ge F^l \quad \forall l \in L && (9.12)\\
& x_p^l \ge 0 \quad \forall l \in L, \forall p \in P^l
\end{aligned}$$

Constraint (9.7) becomes redundant and is removed from the formulation.

Pricing Problem

Let π_e, λ^l and σ^l be the dual variables for equations (9.10), (9.11) and (9.12) in *(LP2)*. The reduced cost for a commodity $l \in L$ and for a path $p \in P^l$ is given by:

$$\sum_{e\in E} \delta_e^p (c_e - \pi_e) - \frac{\lambda^l}{u_p} + \sigma^l \tag{9.13}$$

We have that $c_p \ge 0$, $\sigma^l \le 0$ and the terms $-\sum_{e\in E}\delta_e^p\pi_e$ and $-\lambda^l/u_p$ are nonnegative since $\pi_e \le 0$ and $\lambda^l \le 0$. The problem (9.13) is thus equivalent to the pricing problem for the three-index algorithm: a shortest path problem defined in costs $(c_e - \pi_e) \ge 0$ which must be solved for each possible value of u_p for each commodity l.

Branching Strategy

The branching strategy from Section 9.2 unfortunately does not work for the two-index algorithm due to the lacking h-indices in the formulation. Nor can we use the original formulation from Barnhart et al. [26] since we are allowed to use k^l paths for each commodity. Thus, we have developed a novel branching strategy for the *(LP2)* formulation of the problem.

The branching strategy for the two-index algorithm consists of forbidding sequences of edges. In the general case it does not suffice to forbid the use of a single edge or node. Consider a divergence node for some commodity. The number of paths emanating from this node may be larger than the number of outgoing edges. Thus, forbidding an edge can result in forbidding several paths. This is not desirable, as an optimal solution becomes unreachable when it uses all edges going out of a divergence node. A similar

situation can occur when forbidding nodes. Instead, paths emanating from a divergence node are considered.

Let γ_{vl} be the set of paths for commodity $l \in L$ emanating from divergence node v, and let the number of elements in γ_{vl} be greater than k^l. In this case, branching is necessary. A feasible solution includes at most k^l of the paths in γ_{vl}. Thus, the paths in γ_{vl} are divided into $k^l + 1$ branching children, and when branching, the paths in the corresponding branching child are forbidden.

The branching strategy is feasible since any subset of k^l paths from γ_{vl} can be used in a solution in at least one of the branching children. Consider any subset of k^l paths from γ_{vl}. Each of the paths in the subset is forbidden in exactly one branching child, i.e., the total number of branching children, including at least one of the paths, is at most k^l. Since $k^l + 1$ branching children are generated, at least one branching child holds none of the k^l paths.

When branching, the resulting solution space in each branching child is reduced according to the forbidden paths. The solution spaces of branching siblings, however, are not necessarily disjoint; a solution using less than k^l paths from γ_{vl} is feasible in several branching children. The branching strategy may thus impose degeneracy problems.

The number of columns in the master problem is possibly exponential. Thus, the branching strategy may cause a large search tree, because the number of paths to forbid can be very large.

To limit degeneracy problems and to limit the size of the search tree, the branching strategy is changed into forbidding certain sequences of edges rather than forbidding entire paths. A path consists of a sequence of edges. The $k^l+\alpha, \alpha \geq 1$ paths in γ_{vl} may share several edges, but two paths never share all edges. When generating the branching children it thus suffices to find $k^l + 1$ different edge sequences used by the paths in γ_{vl}. Each edge sequence must be consecutive, i.e., it forms a connected subpath, and no two subpaths share all edges. Let Γ_{vl} be the set of the $k^l + 1$ different edge sequences derived from the paths in γ_{vl}. Let the edge sequences be derived such, that each path in γ_{vl} uses exactly one of the edge sequences, and each edge sequence consists of as few edges as possible. This can be done by a breadth first search of all edges used by the paths in γ_{vl}. When branching, the edge sequences of the corresponding branching child are forbidden. This is feasible by the same argument for the strategy forbidding entire paths.

The reason for forbidding edge sequences rather than entire paths, is that a forbidden edge sequence may cut off more of the solution space, because more than one path is possibly forbidden. This may lead to less degeneracy in the branching children, and to a smaller search tree size.

An illustration of the branching strategy is seen in Figure 9.1. In the figure, a graph with four nodes is seen. A commodity with source s and target t is to be routed using at most two paths. In the current solution three paths are used: $p_1 = \{e_A, e_D, e_E\}, p_2 = \{e_A, e_C, e_E\}$ and $p_3 = \{e_B, e_C, e_E\}$. Assume that the optimal solution consists of path p_1 and p_3. When branching on the current solution it is thus not feasible to forbid the use of any single path or node. Instead, $k^l + 1$ subpaths are found: $\{e_A, e_C\}$, $\{e_A, e_D\}$ and $\{e_B\}$. Now, the optimal solution is found in the branching child which forbids the use of edge sequence $\{e_A, e_C\}$.

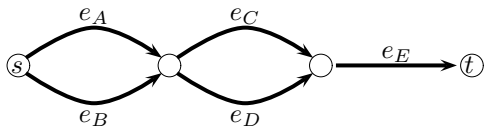


Figure 9.1: A graph used to illustrate the branching strategy. The graph consists of four nodes, the leftmost node is denoted s, and the rightmost node, t. Edges are e_A, e_B, e_C, e_D and e_E.

The branching strategy necessitates some changes to the pricing problem. When solving the shortest path problem, we need to ensure that we do not use the forbidden edge sequences. The shortest path problem with forbidden paths is a polynomial problem and can be solved using a modified k-shortest path algorithm [199].

9.4 Computational results

The described branch-and-price algorithms for the two models were tested on a 2.66 GHz Intel Xeon machine with 8 Gb RAM. Note, that CPU times in the following stem from using one core. The algorithms have been implemented using the framework `COIN` [140] with `ILOG CPLEX 10.2` as LP-solver. Computations regarding selection of branching candidate and branching child are handled by `COIN`.

When reporting the running times of the three-index model, we refer to our own implementation of the three-index algorithm. All tests have been performed with uniform values of k, i.e., $k^l = k$ for all commodities $l \in L$.

In both algorithms we have through preliminary results [87] decided to use strong branching [14]. We investigate all possible branching candidates. A best-first search strategy is used in the branch-and-bound tree. Also, based on [87], we set the number of paths priced in per iteration to $0.5 \cdot |L| \cdot k$ for the three-index algorithm and to $0.5 \cdot |L|$ for the two-index algorithm. For the three-index model we multiply the number

of paths priced in per iteration with k because of the extra h-index in the model. For both algorithms we never price more than one path into the restricted master problem for each pair of values (h, l) or for each commodity l, respectively, per iteration. This is to keep column generation simple.

The algorithms are tested on four types of problems: The Carbin instances [7], also denoted `bl`, `bs`, and the `grid` and `planar` instances [134]. The Carbin instances are randomly generated problems. The `grid` instances are formed as grids, and the `planar` are designed to simulate problems arising in telecommunication. Note, that we have not performed tests on all of the instances. We have not solved the Carbin instances with variable edge weights, because the algorithms cannot handle this. For the `grid` and the `planar` instances this is due to the algorithms being unable to solve the larger instances in reasonable time.

First, we test the branch-and-price algorithm for the three-index model with and without the proposed heuristic. Results can be seen in Table 9.1. Overall, the running time is improved significantly for the solved instances when the heuristic is included. As can be seen in the table, this is due to achieving a smaller search tree when using the heuristic; the impact of the heuristic is that less branching is required to reach a feasible solution. Furthermore, the table shows that very little time is spent on running the heuristic. For several instances, the optimal solution is found in the root node when using the heuristic. This, however, is not the case for all instances. For the unsolved instances, using the heuristic either leads to better bounds or it has no effect on the performance. Throughout the remaining of this section, the heuristic is thus included in the branch-and-price algorithm for the three-index model, and the heuristic is run in every node of the search tree.

Next, we compare the two branch-and-price algorithms with each other. A summary of the results can be seen in Table 9.2. Table 9.3 and 9.4 show detailed test data for the Carbin instances, Table 9.5 shows detailed test data for the `planar` instances and Table 9.6 shows detailed test data for the `grid` instances.

For $k = 2$, the three-index algorithm solves only three of the `bs` instances and six of the `bl` instances, while the two-index algorithm is capable of solving nine out of the eleven `bs` instances and all the `bl` instances. The average running time for the two-index algorithm is considerably better than for the three-index algorithm. For $k = 3$, the three-index algorithm is unable to solve three of the `bs` instances and one `bl` instance, where the two-index algorithm solves all to optimality. Again the two-index algorithm shows a better average running time than that of the three-index algorithm. Both algorithms are capable of solving all the Carbin instances for $k = 10$, however, the two-index algorithm averagely spends less time on doing so than the three-index algorithm.

The running times reflect the complexity of the corresponding problem instances and

used algorithms. Whenever the value of k exceeds some threshold value, the running time for solving the instance decreases. The reason for this is that at some point, k does not impose a constraint on the problem, i.e., the instance corresponds to the linear MCFP. The value of k has greater impact on the three-index algorithm. When k takes on a value greater than the mentioned threshold, the running time of the three-index algorithm increases, because columns are generated for each $h = 1, \ldots, k$, and are priced into the master problem. Generating columns and solving a larger master problem is time consuming. Also, even if the value of k is greater than the threshold, the three-index algorithm may generate solutions using more then one path as the h'th path, hence causing the algorithm to branch. The same is obviously not the case for the two-index algorithm.

The three-index algorithm fails to solve the largest `planar` instance for $k = 2$ and $k = 3$. The two-index algorithm solves all the `planar` instances. The average running time for the algorithms shows, that the two-index algorithm performs significantly better than the three-index algorithm for $k = 2$ and $k = 10$, but the three-index algorithm has smaller running time for $k = 3$.

For larger `grid` instances with $k = 2$, both algorithms experience problems. The three-index algorithm solves four, and the two-index algorithm solves five out of seven instances. For $k = 3$, the three-index algorithm manages to solve five out of seven instances, and the two-index algorithm solves all instances. For $k = 10$ all instances are solved. Again, the two-index algorithm shows a better average running time than the three-index algorithm for $k = 2$ and $k = 10$, while the opposite holds for $k = 3$. Larsson and Yuan [134] are capable of solving all `grid` instances as the linear MCFP. Neither of the two algorithms here presented are capable of solving instances as large as Larsson and Yuan, which is due to our algorithms not being specialized for the linear MCFP.

The three-index algorithm is capable of solving instances with up to 2239 commodities, 850 edges and 150 nodes (planar_{150}), and 400 commodities, 1520 edges and 400 nodes ($\text{grid}_{400:1520:400}$) for $k = 10$, and instances with up to 532 commodities, 1085 edges and 100 nodes (planar_{100}) for $k = 2$. The two-index algorithm solves instances with up to 2239 commodities, 850 edges and 150 nodes (planar_{150}) and 400 commodities, 1520 edges and 400 nodes ($\text{grid}_{400:1520:400}$) for $k = 10$, and instances with up to 2239 commodities, 850 edges and 150 nodes (planar_{150}) for $k = 2$. Also, the three-index algorithm is capable of solving about 76% of the test instances to optimality, while the two-index has solved just over 96% of the test instances to optimality. Hence, for the far majority of the problem instances, the two-index algorithm outperforms the three-index algorithm both with respect to time spent and to the number of instances solved to optimality. We conclude that this is partly due to the extra h-index in the three-index model causing symmetry in the solution space, and partly due to the three-index algorithm having k times as many variables as the two-index algorithm.

9.5 Conclusions

In this paper we have presented a branch-and-price algorithm for the MCMCkFP which outperforms existing methods. The new branch-and-price algorithm is based on a two-index formulation, which unlike previous formulations omits a symmetry inducing index for each of the k paths per commodity. The two-index model was independently suggested for the Maximum Flow MCkFP by Truffot et al. [191], but the authors discarded the model since it complicates branching. We have presented a branching strategy for the model which ensures that the pricing problem can be solved efficiently. The branching strategy and the algorithm for the resulting pricing problem can also be used for the Maximum Flow problem. Thus, our branch-and-price algorithm can be viewed as a general framework applicable for various variants of the MCkFP.

Furthermore, we have introduced a rounding heuristic for the three-index branch-and-price algorithm which transforms certain fractional solutions into feasible solutions. Though the heuristic boosts the performance of the three-index algorithm, it is still outperformed by the two-index algorithm. The three-index algorithm including the proposed heuristic has solved 76% of the problem instances to optimality within the available time and space, where the two-index has solved 96% of the problem instances to optimality. Further comparison of the algorithms shows, that the two-index branch-and-price algorithm also outperforms the three-index algorithm with respect to running time.

The solution times for the Minimum Cost MCkFP are larger than those of Barnhart et al. [26] for the unsplittable MCFP. This indicates that the k-splittable constraints are harder to maintain than the unsplittable constraints, probably because the k-splittable constraints increase the size of the solution space and introduce symmetry. In order to improve the performance it could be interesting to tighten the formulations through various cuts, as done in e.g. Jepsen et al. [111]. Also, adding constraints which break the symmetry might improve the solution times. The introduction of a good initial heuristic will only marginally improve the running times, since the current algorithm generally quickly finds a good upper bound.

Problem, k	Heur.	Time	H. Time	Tree size	Depth	Col.	Gap	UB
bl01, 2	no	176.28	-	>48000	54	353	0.04	1549555.0
bl01, 2	yes	177.74	-	>48000	54	356	0.04	1549555.0
bl01, 3	no	0.80	-	101	45	525	0.00	1548873.0
bl01, 3	yes	0.33	<0.01	35	17	525	0.00	1548873.0
bl01, 10	no	0.91	-	31	15	1740	0.00	1548873.0
bl01, 10	yes	0.12	<0.01	1	0	1740	0.00	1548873.0
bl03, 2	no	225.06	-	>34000	75	422	0.23	15836.0
bl03, 2	yes	225.38	-	>34000	75	417	0.23	15836.0
bl03, 3	no	2.98	-	317	49	591	0.00	15799.0
bl03, 3	yes	0.44	<0.01	1	0	591	0.00	15799.0
bl03, 10	no	2.17	-	63	31	2020	0.00	15799.0
bl03, 10	yes	0.13	<0.01	1	0	2020	0.00	15799.0
bs01, 2	no	212.65	-	>43000	73	407	0.23	1536558.0
bs01, 2	yes	213.30	-	>43000	73	408	0.23	1536558.0
bs01, 3	no	73.25	-	6295	65	579	0.00	1533606.0
bs01, 3	yes	66.15	0.32	5531	45	579	0.00	1533606.0
bs01, 10	no	5.34	-	171	45	1870	0.00	1533095.0
bs01, 10	yes	0.11	<0.01	1	0	1870	0.00	1533095.0
bs03, 2	no	0.59	-	125	28	325	0.00	16488.0
bs03, 2	yes	0.47	<0.01	97	25	325	0.00	16488.0
bs03, 3	no	0.17	-	29	14	438	0.00	16488.0
bs03, 3	yes	0.02	<0.01	1	0	438	0.00	16488.0
bs03, 10	no	1.31	-	61	25	1470	0.00	16488.0
bs03, 10	yes	0.08	0.04	1	0	1470	0.00	16488.0
bs13, 2	no	581.78	-	>11000	234	1397	0.17	3259617.5
bs13, 2	yes	569.69	-	>11000	231	1393	0.17	3259606.0
bs13, 3	no	492.96	-	>10000	208	2110	0.01	3254481.25
bs13, 3	yes	529.69	-	>10000	168	2103	<0.01	3254331.5
bs13, 10	no	222.87	-	857	176	7039	0.00	3254081.06
bs13, 10	yes	1.42	0.01	1	0	7030	0.00	3254081.06

Table 9.1: Results for the three-index algorithm with and without the proposed heuristic. The second column **Heur.**, indicates whether the heuristic is included. Next follows total time usage (**Time**), and time spent on running the heuristic (**H. Time**). The Table gives information about the tree size and depth (**Tree size** and **Depth**), as well as the number of columns added to the master problem (**Col.**). **Time** is measured in seconds and **Gap** in percent between upper and lower bound. An optimal solution is found whenever Gap=0.00. A non-zero gap, indicate that the testrun ran out of memory. Upper bounds have been rounded to two decimal precision.

Name	k	# instances	3-index		2-index	
			A.Mean	**Opt.**	**A.Mean**	**Opt.**
bl	2	11	5.06	6/11	1.90	11/11
bl	3	11	0.43	10/11	0.21	11/11
bl	10	11	0.87	11/11	0.22	11/11
bs	2	11	41.66	3/11	0.32	9/11
bs	3	11	37.95	8/11	0.32	11/11
bs	10	11	1.08	11/11	0.27	11/11
planar	2	5	117.92	4/5	3.09	5/5
planar	3	5	2.58	4/5	2.75	5/5
planar	10	5	267.40	5/5	15.13	5/5
grid	2	7	1.40	4/7	0.24	5/7
grid	3	7	0.09	5/7	0.73	7/7
grid	10	7	7.00	7/7	1.31	7/7

Table 9.2: The number of test instances solved to optimality with the 3-index and 2-index algorithms, for various k values. **A.Mean** is the average mean time in seconds calculated over those instances solved to optimality by both algorithms.

Name	k	3-index			2-index		
		Time	**Gap**	**UB**	**Time**	**Gap**	**UB**
bl_{01}	2	177.45	0.04	1549555.00	1.79	0.00	1549555.00
bl_{01}	3	0.33	0.00	1548873.00	0.05	0.00	1548873.00
bl_{01}	10	0.11	0.00	1548873.00	0.02	0.00	1548873.00
bl_{03}	2	224.00	0.23	15836.00	5.28	0.00	15836.00
bl_{03}	3	0.04	0.00	15799.00	0.04	0.00	15799.00
bl_{03}	10	0.13	0.00	15799.00	0.04	0.00	15799.00
bl_{05}	2	2.72	0.00	460698.00	9.55	0.00	460698.00
bl_{05}	3	0.73	0.00	460041.00	0.08	0.00	460041.00
bl_{05}	10	0.09	0.00	460037.00	0.02	0.00	460037.00
bl_{07}	2	0.04	0.00	5588.00	0.04	0.00	5588.00
bl_{07}	3	0.04	0.00	5588.00	0.03	0.00	5588.00
bl_{07}	10	0.15	0.00	5588.00	0.04	0.00	5588.00
bl_{09}	2	11.97	0.00	6106441.00	0.99	0.00	6106441.00
bl_{09}	3	0.20	0.00	6106255.00	0.18	0.00	6106255.00
bl_{09}	10	0.75	0.00	6106255.00	0.17	0.00	6106255.00
bl_{11}	2	2.70	0.00	68088.50	0.21	0.00	68088.50
bl_{11}	3	0.11	0.00	68086.00	0.13	0.00	68086.00
bl_{11}	10	0.48	0.00	68086.00	0.13	0.00	68086.00
bl_{15}	2	345.16	0.05	32237.00	77.76	0.00	32235.00
bl_{15}	3	0.24	0.00	32220.00	0.24	0.00	32220.00
bl_{15}	10	0.82	0.00	32220.00	0.26	0.00	32220.00
bl_{17}	2	5.33	0.00	13086437.00	0.58	0.00	13086437.00
bl_{17}	3	0.30	0.00	13086437.00	0.32	0.00	13086437.00
bl_{17}	10	1.33	0.00	13086437.00	0.28	0.00	13086437.00
bl_{19}	2	7.59	0.00	108027.00	0.62	0.00	108027.00
bl_{19}	3	0.41	0.00	108027.00	0.42	0.00	108027.00
bl_{19}	10	1.68	0.00	108027.00	0.35	0.00	108027.00
bl_{21}	2	463.15	0.02	5571253.00	19.27	0.00	5571239.00
bl_{21}	3	0.55	0.00	5570292.00	0.58	0.00	5570292.00
bl_{21}	10	2.20	0.00	5570292.00	0.55	0.00	5570292.00
bl_{23}	2	472.65	0.02	54414.50	58.84	0.00	54414.50
bl_{23}	3	429.00	$<$0.01	54402.00	2.14	0.00	54402.00
bl_{23}	10	1.80	0.00	54401.00	0.51	0.00	54401.00

Table 9.3: Results for the three-index and the two-index algorithms on the Carbin instances called `bl`. **Time** is measured in seconds and **Gap** in percent between upper and lower bound. An optimal solution is found whenever Gap=0.00. The maximal running time is set to 1800 seconds. Results with Time < 1800, and a non-zero gap, indicate that the testrun ran out of memory. Upper bounds have been rounded to two decimal precision.

Name	k	3-index			2-index		
		Time	**Gap**	**UB**	**Time**	**Gap**	**UB**
bs_{01}	2	213.71	0.23	1536558.00	5.21	0.00	1536558.00
bs_{01}	3	66.15	0.00	1533606.00	0.10	0.00	1533606.00
bs_{01}	10	0.11	0.00	1533095.00	0.04	0.00	1533095.00
bs_{03}	2	0.55	0.00	16488.00	0.10	0.00	16488.00
bs_{03}	3	0.03	0.00	16488.00	0.02	0.00	16488.00
bs_{03}	10	0.08	0.00	16488.00	0.04	0.00	16488.00
bs_{05}	2	331.11	0.59	410502.00	9.57	0.00	410417.00
bs_{05}	3	380.74	0.09	408496.00	0.43	0.00	408496.00
bs_{05}	10	0.19	0.00	408114.00	0.07	0.00	408114.00
bs_{07}	2	273.61	0.28	5816.00	4.65	0.00	5816.00
bs_{07}	3	182.37	0.00	5801.00	0.16	0.00	5801.00
bs_{07}	10	0.14	0.00	5800.00	0.06	0.00	5800.00
bs_{11}	2	403.03	$<$0.01	63381.83	0.84	0.00	63381.83
bs_{11}	3	0.32	0.00	63380.33	0.34	0.00	63380.33
bs_{11}	10	1.19	0.00	63380.33	0.21	0.00	63380.33
bs_{13}	2	506.64	0.17	3259573.50	523.72	0.02	3258178.50
bs_{13}	3	610.59	$<$0.01	3254299.00	5.77	0.00	3254192.72
bs_{13}	10	1.28	0.00	3254081.06	0.35	0.00	3254081.06
bs_{15}	2	440.36	0.09	35392.00	62.74	$<$0.01	35390.00
bs_{15}	3	499.73	$<$0.01	35362.00	0.77	0.00	35362.00
bs_{15}	10	0.78	0.00	35360.00	0.28	0.00	35360.00
bs_{17}	2	62.52	0.00	11323466.00	0.53	0.00	11323466.00
bs_{17}	3	0.33	0.00	11323427.00	0.34	0.00	11323427.00
bs_{17}	10	1.32	0.00	11323427.00	0.33	0.00	11323427.00
bs_{19}	2	58.90	0.00	105449.50	0.34	0.00	105449.50
bs_{19}	3	0.29	0.00	105449.50	0.36	0.00	105449.50
bs_{19}	10	1.26	0.00	105449.50	0.31	0.00	105449.50
bs_{21}	2	658.58	0.03	5194721.00	27.37	0.00	5194297.00
bs_{21}	3	53.54	0.00	5193164.50	0.62	0.00	5193164.50
bs_{21}	10	2.93	0.00	5193164.50	0.64	0.00	5193164.50
bs_{23}	2	616.91	0.05	53994.50	652.64	0.00	53987.00
bs_{23}	3	0.56	0.00	53968.63	0.62	0.00	53968.63
bs_{23}	10	2.65	0.00	53968.63	0.68	0.00	53968.63

Table 9.4: Results for the three-index and the two-index algorithms on the Carbin instances called bs. **Time** is measured in seconds and **Gap** in percent between upper and lower bound. An optimal solution is found whenever Gap=0.00. The maximal running time is set to 1800 seconds. Results with Time < 1800, and a non-zero gap, indicate that the testrun ran out of memory. Upper bounds have been rounded to two decimal precision.

Name	k	3-index			2-index		
		Time	**Gap**	**UB**	**Time**	**Gap**	**UB**
planar$_{30}$	2	0.04	0.00	44350624.00	0.06	0.00	44350624.00
planar$_{30}$	3	0.06	0.00	44350624.00	0.06	0.00	44350624.00
planar$_{30}$	10	0.20	0.00	44350624.00	0.07	0.00	44350624.00
planar$_{50}$	2	0.29	0.00	122199689.00	0.87	0.00	122199689.00
planar$_{50}$	3	0.39	0.00	122199689.00	0.55	0.00	122199689.00
planar$_{50}$	10	1.79	0.00	122199689.00	0.50	0.00	122199689.00
planar$_{80}$	2	243.72	0.00	182438134.00	6.70	0.00	182438134.00
planar$_{80}$	3	2.90	0.00	182438134.00	2.42	0.00	182438134.00
planar$_{80}$	10	15.93	0.00	182438134.00	2.46	0.00	182438134.00
planar$_{100}$	2	227.61	0.00	231339582.00	10.81	0.00	231339582.00
planar$_{100}$	3	6.97	0.00	231339582.00	7.95	0.00	231339582.00
planar$_{100}$	10	37.12	0.00	231339582.00	7.64	0.00	231339582.00
planar$_{150}$	2	1503.54	$>$1000	545566045720.00	248.59	0.00	548087089.00
planar$_{150}$	3	1302.07	$>$1000	545566045720.00	83.24	0.00	548087089.00
planar$_{150}$	10	1281.94	0.00	548087089.00	64.99	0.00	548087089.00

Table 9.5: Results for the three-index and the two-index algorithms on `planar` instances. **Time** is measured in seconds and **Gap** in percent between upper and lower bound. An optimal solution is found whenever Gap=0.00. The maximal running time is set to 1800 seconds. Results with Time < 1800, and a non-zero gap, indicate that the testrun ran out of memory. Upper bounds have been rounded to two decimal precision.

Name	k	3-index			2-index		
		Time	**Gap**	**UB**	**Time**	**Gap**	**UB**
$\text{grid}_{25:80:50}$	2	0.03	0.00	827319.00	0.05	0.00	827319.00
$\text{grid}_{25:80:50}$	3	0.05	0.00	827319.00	0.02	0.00	827319.00
$\text{grid}_{25:80:50}$	10	0.07	0.00	827319.00	0.03	0.00	827319.00
$\text{grid}_{25:80:100}$	2	0.42	0.00	1705378.00	0.08	0.00	1705378.00
$\text{grid}_{25:80:100}$	3	0.08	0.00	1705378.00	0.08	0.00	1705378.00
$\text{grid}_{25:80:100}$	10	0.24	0.00	1705378.00	0.06	0.00	1705378.00
$\text{grid}_{100:360:50}$	2	1.06	0.00	1524657.00	0.21	0.00	1524657.00
$\text{grid}_{100:360:50}$	3	0.06	0.00	1524642.00	0.05	0.00	1524642.00
$\text{grid}_{100:360:50}$	10	0.18	0.00	1524642.00	0.06	0.00	1524642.00
$\text{grid}_{100:360:100}$	2	4.07	0.00	3031717.00	0.61	0.00	3031717.00
$\text{grid}_{100:360:100}$	3	0.18	0.00	3031695.00	0.20	0.00	3031695.00
$\text{grid}_{100:360:100}$	10	0.61	0.00	3031695.00	0.18	0.00	3031695.00
$\text{grid}_{225:840:100}$	2	133.91	$<$0.01	5049776.50	70.71	0.00	5049759.50
$\text{grid}_{225:840:100}$	3	14.52	0.00	5049688.50	3.28	0.00	5049688.50
$\text{grid}_{225:840:100}$	10	1.84	0.00	5049688.50	0.54	0.00	5049688.50
$\text{grid}_{225:840:200}$	2	275.21	$<$0.01	10402290.80	212.23	$<$0.01	10402154.75
$\text{grid}_{225:840:200}$	3	309.44	$<$0.01	10401819.87	7.18	0.00	10401782.00
$\text{grid}_{225:840:200}$	10	12.19	0.00	10401782.00	1.92	0.00	10401782.00
$\text{grid}_{400:1520:400}$	2	252.51	$>$1000	15281128750.00	614.19	$<$0.01	25864060.50
$\text{grid}_{400:1520:400}$	3	251.16	$>$1000	15281128750.00	28.07	0.00	25864036.57
$\text{grid}_{400:1520:400}$	10	33.85	0.00	25864036.57	6.39	0.00	25864036.57

Table 9.6: Results for the three-index and the two-index algorithms on the `grid` instances. **Time** is measured in seconds and **Gap** in percent between upper and lower bound. An optimal solution is found whenever Gap=0.00. The maximal running time is set to 1800 seconds. Results with Time < 1800, and a non-zero gap, indicate that the testrun ran out of memory. Upper bounds have been rounded to two decimal precision.

CHAPTER 10

Comparing branch-and-price algorithms for the multi-commodity k-splittable maximum flow problem

Mette Gamst
DTU Management Engineering, Technical University of Denmark

Bjørn Petersen
DTU Management Engineering, Technical University of Denmark

The Multi-Commodity k-splittable Maximum Flow Problem consists of routing as much flow as possible through a capacitated network such that each commodity uses at most k paths and the capacities are satisfied. The problem has previously been solved to optimality through branch-and-price. In this paper we propose two new exact solution methods both based on an alternative decomposition. The two methods differ

In submission 2009

in their branching strategy. The first method, which branches on forbidden edge sequences, shows some performance difficulty due to large search trees. The second method, which branches on forbidden and forced edge sequences, demonstrates much better performance. The latter also outperforms a leading exact solution method from the literature. Furthermore, a heuristic algorithm is presented. The heuristic is fast and yields good solution values.

Key words: Multi-Commodity flow, k-splittable, branch-and-price, Dantzig-Wolfe decomposition

10.1 Introduction

The Multi-Commodity k-splittable Maximum Flow Problem (MCkMFP) consists of maximizing the amount of routed flow through a capacitated network such that each commodity uses at most k paths and the capacities are satisfied. The MCkMFP appears in the transportation sector when a number of commodities must be routed using only a limited number of transportation units, and in telecommunication for limiting the number of used network connections.

The Multi-Commodity k-splittable Flow Problem (MCkFP) was presented by Baier et al. [19], who solved the Maximum Budget-Constrained Single- and Multi-Commodity k-splittable Flow Problems using approximation algorithms. The authors proved that the Maximum Single-Commodity k-splittable Flow Problem is $\mathcal{NP}$-hard in the strong sense for directed graphs. Finally, they noted that for $k \geq |E|$, a k-splittable (s,t) flow problem degenerates to an ordinary (s,t) flow problem.

Koch et al. [123] proved that the MCkMFP is $\mathcal{NP}$-hard in the strong sense for directed as well as undirected graphs, and showed that when $\mathcal{P} \neq \mathcal{NP}$, the best possible approximation factor is $\frac{5}{6}$. Koch et al. [122] considered the MCkMFP as a two-stage problem, where the first stage consists of the decision on the k paths (routing) and the second of the amount of flow on the paths (packing). If k is a constant then it suffices to consider a polynomial number of packing alternatives, which can be constructed in polynomial time. If k is part of the input, they proposed an approximation algorithm having approximation factor $(1-\epsilon)$, $\epsilon > 0$.

Truffot and Duhamel [190] used branch-and-price to solve the Single-Commodity k-splittable Maximum Flow Problem (SCkMFP). A 3-index edge-path model was presented to which a branch-and-price algorithm was applied. The pricing problem for the column generation is a shortest path problem solvable in polynomial time. Furthermore, Truffot et al. [192] have applied their 3-index branch-and-price algorithm to the

MCkMFP.

Gamst et al. [83] used branch-and-price to solve the Minimum Cost Multi-Commodity k-splittable Flow Problem (MCMCkFP). They applied the algorithm of Truffot et al. [192] to the MCMCkFP. Furthermore, they proposed a new branch-and-price algorithm based on a 2-index model. The latter showed very good performance and outperformed the existing branch-and-price algorithm.

The MCkMFP can be represented by a directed graph $G = (V, E)$, where V is the set of vertices and E the set of edges. A positive capacity u_e is associated with every edge $e \in E$. Edge capacities are positive since any edge $e \in E$ with non-positive capacity can be removed from the graph. The set of commodities is denoted L and each commodity $l \in L$ has a source $s_l \in E$ and a destination $t_l \in E$. The maximal number of routes each commodity may use is denoted k.

In this paper three exact solution methods are applied to the MCkMFP and compared. The 3-index branch-and-price algorithm (3BP) by Truffot et al. [192] is extended with a heuristic proposed by Gamst et al. [83] to reach feasible solutions faster. The extended 3BP is compared to two algorithms based on a 2-index formulation by Truffot and Duhamel [190] which was never investigated further. Both algorithms are based on the 2-index branch-and-price algorithm of Gamst et al. [83] applied to the MCkMFP. The two algorithms only differ in their branching scheme. The first algorithm (2BP) uses the same branching strategy as in the literature where certain subpaths are forbidden and the second algorithm (2BP') uses a new branching strategy where the use of certain paths is either forced or forbidden.

The main contribution of this paper is to apply the 2BP algorithm to the MCkMFP and especially to introduce the branching scheme of the 2BP' algorithm. Furthermore, a heuristic use of the 2BP and 2BP' algorithms is presented, denoted 2HEUR.

The paper is organized as follows. First, the MCkMFP is formally introduced in Section 10.2. The 2BP algorithm is presented in Section 10.3, which is followed by the 2BP' algorithm in Section 10.4. All algorithms are tested and compared in Section 10.5. Section 10.6 concludes the paper.

10.2 The multi-commodity *k*-splittable maximum flow problem

The MCkMFP can be formulated as an edge-based model on the graph G. The model contains two types of variables: the flow variables x_e^{hl} representing the amount of flow

on edge e for the h'th path of commodity l and the decision variables y_e^{hl} indicating whether or not edge e is used by the h'th path of commodity l. A backward edge e_l with unlimited capacity and with flow $x_{e_l}^h$ is added for each commodity l to ease flow conservation constraints. Edge (t, s) in Figure 10.1 is a backward edge.

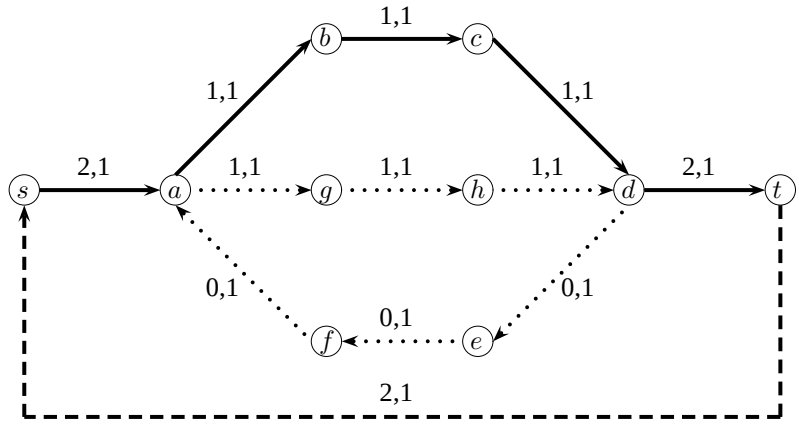


Figure 10.1: The example illustrates an infeasible path. The path $P : s \to a \to b \to c \to d \to e \to f \to a \to g \to h \to d \to t$ contains a subtour and the amount of flow on used edges differs. The illustration is taken from [192].

To model flow conservation, let the set of incoming edges at vertex v be denoted $\phi^-(v)$ and the set of outgoing edges at vertex v be denoted $\phi^+(v)$. For each commodity, the sum of ingoing edges must equal the sum of outgoing edges at each vertex. Similarly for each commodity, the total amount of incoming flow must equal the total amount of outgoing flow at each vertex. Subtours may occur, as shown in Figure 10.1. The vertices s and t denote the source and destination of a given commodity consisting of 2 units of flow. At each edge e the pair (x_e^{hl}, y_e^{hl}) is given. Consider the path: $s \to a \to b \to c \to d \to e \to f \to a \to g \to h \to d \to t \to s$. The path is not feasible because of the subtour, but flow conservation is satisfied. The subtour can be eliminated by adding a constraint saying that for the h'th path of commodity l, each vertex can have at most one incoming (and thus outgoing) edge. The edge-based model

now becomes:

$$\max \quad \sum_{l\in L}\sum_{h=1}^{k}\sum_{e\in\phi^+(s_l)} x_e^{hl} \tag{10.1}$$

$$\text{s.t.} \quad \sum_{e\in\phi^-(v)} x_e^{hl} = \sum_{e\in\phi^+(v)} x_e^{hl} \quad \forall v\in V, \forall l\in L, \forall h\in\{1,\dots,k\} \tag{10.2}$$

$$\sum_{e\in\phi^-(v)} y_e^{hl} = \sum_{e\in\phi^+(v)} y_e^{hl} \quad \forall v\in V, \forall l\in L, \forall h\in\{1,\dots,k\} \tag{10.3}$$

$$\sum_{l\in L}\sum_{h=1}^{k} x_e^{hl} \le u_e \qquad \forall e\in E \tag{10.4}$$

$$x_e^{hl} - u_e y_e^{hl} \le 0 \qquad \begin{array}{l}\forall l\in L, \forall h\in\{1,\dots,k\},\\ \forall e\in E\cup\{e_{e_l}^h\}\end{array} \tag{10.5}$$

$$\sum_{e\in\phi^-(v)} y_e^{hl} \le 1 \qquad \forall l\in L, \forall h\in\{1,\dots,k\}, \forall v\in V \tag{10.6}$$

$$x_e^{hl} \ge 0 \qquad \forall l\in L, \forall h\in\{1,\dots,k\}, \forall e\in E$$

$$y_e^{hl} \in \{0,1\} \qquad \forall l\in L, \forall h\in\{1,\dots,k\}, \forall e\in E$$

The objective function (10.1) maximizes the total amount of routed flow. Constraints (10.2) and (10.3) are flow conservation constraints, (10.4) ensure that the capacity constraint on each edge is not violated and (10.5) force each decision variable y_e to take on value 1 whenever the amount of flow on edge e is positive. Constraints (10.6) prevent subtours.

Truffot et al. [192] solved the MCkMFP by applying Dantzig-Wolfe decomposition to the edge-based model [54]. We denote their branch-and-price algorithm 3BP. The pricing problem finds the h'th path of commodity l and the master problem merges paths into an overall feasible solution. Here, we present the master problem to motivate our work on the MCkMFP. In the master problem, the variable $x_p^{hl} \ge 0$ denotes the amount of flow on path p for the h'th path of commodity l and the binary variable y_p^{hl} denotes whether or not path p is used as the h'th path for commodity l. The 3BP problem is:

$$\max \quad \sum_{l\in L}\sum_{h=1}^{k}\sum_{p\in P^l} x_p^{hl}$$

$$\text{s.t.} \quad \sum_{l\in L}\sum_{h=1}^{k}\sum_{p\in P^l} \delta_e^p x_p^{hl} \le u_e \quad \forall e\in E \tag{10.7}$$

$$x_p^{hl} - u_p y_p^{hl} \le 0 \qquad \forall l\in L, h\in\{1,\dots,k\}, \forall p\in P^l \tag{10.8}$$

$$\sum_{p \in P^l} y_p^{hl} \leq 1 \qquad \forall l \in L, h \in \{1, \ldots, k\} \tag{10.9}$$
$$x_p^{hl} \geq 0 \qquad \forall l \in L, h \in \{1, \ldots, k\}, \forall p \in P^l$$
$$y_p^{hl} \in \{0, 1\} \qquad \forall l \in L, h \in \{1, \ldots, k\}, \forall p \in P^l$$

The objective function maximizes the total amount of routed flow. The set P^l contains paths p for commodity l. In capacity constraints (10.7), δ_e^p indicates whether or not edge e is used by path p. The constant u_p denotes the capacity constraint on path p, which is defined as $u_p = min\{u_e \,|\, e \in p\}$. Hence, constraints (10.8) force the decision variable y_p^{hl} to be set if there is flow on the corresponding path x_p^{hl}. Constraints (10.9) ensure that at most one path is used as the h'th path of a commodity l.

Gamst et al. [83] applied the 3BP algorithm to The Minimum Cost k-splittable Flow Problem and argued that the path index $h \in \{1, \ldots, k\}$ causes symmetry in the solution space as well as a large number of columns in the master problem. To overcome these problems they presented a master problem without the path index and a corresponding branch-and-price algorithm (2BP). In the following sections we show that the 2BP algorithm can be applied to the MCkMFP, we introduce a heuristic to possibly find feasible solutions faster, and we present a branch-and-price algorithm (2BP') based on the same master problem as in the 2BP algorithm, but with a new branching strategy.

10.3 The 2-index branch-and-price algorithm (2BP)

Applying Dantzig-Wolfe decomposition to the edge-based model without using the h-index gives a pricing problem, which generates a path for each commodity, and a master problem, which merges the paths into an overall feasible solution. Let $x_p^l \geq 0$ denote the amount of flow on path p for commodity l and let $y_p^l \in \{0, 1\}$ denote whether or not path p is used by commodity l. The master problem is:

$$\max \quad \sum_{l \in L} \sum_{p \in P^l} x_p^l$$
$$\text{s.t.} \quad \sum_{l \in L} \sum_{p \in P^l} \delta_e^p x_p^l \leq u_e \qquad \forall e \in E \tag{10.10}$$
$$x_p^l - u_p y_p^l \leq 0 \qquad \forall l \in L, \forall p \in P^l \tag{10.11}$$
$$\sum_{p \in P^l} y_p^l \leq k \qquad \forall l \in L \tag{10.12}$$
$$x_p^l \geq 0 \qquad \forall l \in L, \forall p \in P^l$$
$$y_p^l \in \{0, 1\} \qquad \forall l \in L, \forall p \in P^l$$

The objective function maximizes the total amount of routed flow. Constraints (10.10) ensure edge capacities are never violated and constraints (10.11) force the decision variables to take on value 1, whenever the amount of flow on the corresponding path is positive. Constraints (10.12) limit the number of used paths for commodity l to at most k.

By LP-relaxing the binary variables y_p^l to $0 \leq y_p^l \leq 1$ the model is turned into an LP-model. Setting $y_p^l = x_p^l/u_p$ satisfies constraints (10.11) and (10.12) and simplifies the formulation to only consisting of one type of variable. Constraints (10.11) are now redundant and can be removed from the formulation. The relaxed master problem becomes:

$$\max \quad \sum_{l \in L} \sum_{p \in P^l} x_p^l \tag{10.13}$$

$$\text{s.t.} \quad \sum_{l \in L} \sum_{p \in P^l} \delta_e^p x_p^l \leq u_e \quad \forall e \in E \tag{10.14}$$

$$\sum_{p \in P^l} \frac{x_p^l}{u_p} \leq k \qquad \forall l \in L \tag{10.15}$$

$$x_p^l \geq 0 \qquad \forall l \in L, \forall p \in P^l \tag{10.16}$$

10.3.1 Pricing problem

Let $\pi \geq \mathbf{0}$ and $\lambda \geq \mathbf{0}$ be the dual variables for constraints (10.14) and (10.15). The reduced cost for a path $p \in P^l$ for a commodity $l \in L$ is:

$$c_P^l = 1 - \sum_{e \in E} \delta_e^p \pi_e - \frac{\lambda^l}{u_p} \tag{10.17}$$

The pricing problems generate columns with positive reduced cost for each commodity l. Now, λ^l is a constant when l is fixed so finding a column with positive reduced cost (if any exists) is equivalent to solving the shortest path problems:

$$\sum_{e \in E} \delta_e^p \pi_e \leq 1 - \frac{\lambda^l}{u_p}, \quad \forall l \in L, \, \forall p \in P^l$$

The path capacity u_p is not known until the path has been generated. Hence, we set fixed bounds on u_p. We know that the capacity can be set to at most $|E|$ different values (one for each different $u_e : e \in E$), hence the pricing problems can be solved by considering at most $|E|$ shortest path problems. The pricing problems can now be defined as solving the shortest path problem defined on costs $\pi \geq \mathbf{0}$ for edges with $u_e \geq u_p$ for each different $u_e : e \in E$. This can be done in polynomial time by using, e.g., Dijkstra's algorithm.

10.3.2 Heuristic solution

We may reach solutions where more than k paths are used for each commodity. In this case we try to move the flow between the paths in order to find a feasible solution using at most k paths for each commodity. The feasible solution may route less flow through the network, but it can possibly improve the current upper bound in the branch-and-bound scheme and hence help prune parts of the search tree.

For each commodity the heuristic investigates all paths in the current fractional solution and greedily assigns flow to the path having the highest capacity. The steps of the heuristic are:

1: **for** (each commodity) **do**
2: Sort all the paths in the current fractional solution according to decreasing capacity
3: **for** (each path in the sorted list, until flow is assigned to k paths) **do**
4: Assign as much flow as possible to the path
5: Subtract the assigned flow from the capacity of each edge on the path
6: **end for**
7: **end for**

In the case where commodities do not share (many) edges, the heuristic may result in good solutions and hence good upper bounds.

10.3.3 Branching scheme – forbidding edge sequences

The branching scheme consists of forbidding edge sequences. Let the divergence vertex for a commodity be defined as the first vertex with one incoming path and several outgoing paths. If the number of paths emanating from the divergence vertex for some commodity l is greater than k then branching can be applied. For each emanating path p we find the first edges of p, which make p different from the remaining emanating paths. This is denoted the unique edge sequence for p. Each unique edge sequence is forbidden in a branching child. If more than $k + 1$ paths emanate from the divergence vertex, then we let some branching children consist of more than one unique edge sequence such that the number of branching children is always equal to $k + 1$. The reason for this is to reduce the width of the search tree. It is legal to let a branching child forbid several unique edge sequences, because all combinations of k paths from the emanating paths are available in at least one other branching child.

An illustration of the branching strategy is seen in Figure 10.2. A graph with four vertices is given and a commodity with source s and destination t is to be routed using at most two paths. In the current solution three paths are used: $p_1 = \{e_1, e_4, e_5\}, p_2 =$

$\{e_1, e_3, e_5\}$, and $p_3 = \{e_2, e_3, e_5\}$. Assume that the optimal solution consists of path p_1 and p_3. Now $k + 1$ subpaths are found: $\{e_1, e_3\}$, $\{e_1, e_4\}$ and $\{e_2\}$. The optimal solution is found in the branching child, which forbids the use of edge sequence $\{e_1, e_3\}$.

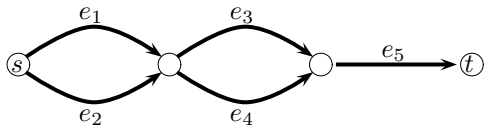


Figure 10.2: A graph used to illustrate the branching scheme. The graph consists of four vertices, the source vertex is denoted s, and the destination vertex t. Edges are e_1, e_2, e_3, e_4, and e_5. The illustration is taken from [83].

The branching scheme changes the pricing problem. When solving the shortest path problem we need to ensure that we do not use the forbidden edge sequences. The shortest path problem with forbidden paths is a polynomial problem and can be solved by applying a shortest path algorithm to an extended graph [199].

10.4 A new 2-index branch-and-price algorithm (2BP')

The 2BP' algorithm only differs from the 2BP algorithm in the branching scheme. The master problem (10.13)–(10.16) is the same and the reduced cost is given by (10.17). The heuristic described in Section 10.3.2 is also applicable.

10.4.1 Branching

The new branching scheme resembles the branching strategy of Cook et al. [51] and is based on the idea of forbidding or forcing the use of a certain path p' for a fixed commodity $l \in L$. This corresponds to setting $y^l_{p'} = 0$ or $y^l_{p'} = 1$, respectively, in the non-relaxed master problem. In the remainder of this section a fixed commodity $l \in L$ is assumed.

The effect of the branching scheme on the non-relaxed master problem, specifically constraint (10.12) is considered:

$$\sum_{p \in P} y^l_p \leq k$$

In both the case that $y^l_{p'} = 0$ or $y^l_{p'} = 1$ the variable can be left out of the constraint. If $y^l_{p'} = 1$ then the constraint is rewritten as

$$\sum_{p \in P \setminus \{p'\}} y^l_p \leq k - 1$$

Now, the effect of the branching scheme on the relaxed master problem, specifically constraint (10.15) is considered:

$$\sum_{p \in P^l} \frac{x^l_p}{u_p} \leq k$$

When path p' is forbidden for commodity l then $x^l_{p'} = 0$. When use of path p' is forced then we set $x^l_{p'} > 0$ and constraint (10.15) is rewritten as

$$\sum_{p \in P^l \setminus \{p'\}} \frac{x^l_p}{u_p} \leq k - 1 \tag{10.18}$$

This is stronger than the original constraint when $x^l_{p'} < u_{p'}$, hence the bound of the branching child is strengthened in this case.

The number of branching children varies according to the current fractional solution. Assume that the current solution consists of $k + \alpha, \alpha > 0$ paths for commodity l. If a path in the current solution carries as much flow as possible, i.e., $x^l_p = u_p$, then forcing the use of path p has no effect because (10.18) is not violated.

Since the current fractional solution is a feasible solution to the relaxed master problem constraints (10.15) are satisfied. Hence, at least $\alpha+1$ paths have $x^l_p < u_p$ (otherwise the sum $\sum_{p \in P} x^l_p / u_p$ would exceed k). An optimal solution may consist of paths not part of the current fractional solution. Thus, we cannot generate $\alpha + 1$ branching children, where the use of exactly one path is forced in each child. Rather, $\alpha + 2$ children should be generated: Each of the first $\alpha + 1$ branching children forces the use of exactly one path p with $x^l_p < u_p$, and the last branching child forbids the use of all $\alpha + 1$ paths.

The first $\alpha + 1$ children cause symmetry in the solution space; several solutions in one branching child can also be found in the other children, especially when several of the $\alpha + 1$ paths are part of the solutions. The first $\alpha + 1$ children are thus changed into forcing *and* forbidding the use of certain paths. Consider the $\alpha + 1 = 3$ branching children b_1, b_2, and b_3, forcing the use of path p_1, p_2, and p_3, respectively. Child b_1 is unaltered and forces the use of p_1. Child b_2 forces the use of p_2 and forbids the use of p_1. In this way, the solution using p_1 and p_2 is only available in the subtree of b_1. Similarly, child b_3 forces the use of p_3 and forbids the use of p_1 and p_2.

In practice we would rather add a cut than rewrite constraints (10.15) when the use of a path is forced. Recall inequality (10.18) when forcing the use of path p'. This inequality is now denoted the branch cut. Let $\omega_{bl} \geq 0$ be the dual of branch cut b for commodity l. The resulting reduced cost for path $p \in P^l$ for commodity $l \in L$ is

$$c_p^l = 1 - \sum_{e \in E} \delta_p^e \pi_e - \frac{\lambda_l}{u_p} - \sum_{b \in B} \frac{\delta_p^b \omega_{bl}}{u_p} \tag{10.19}$$

The extra dual cost ω_{bl} is subtracted from the reduced costs for all new paths for commodity l; this is similar to how λ_l is handled. Hence, the branch cut does not affect edge weights or path properties in the graph of the pricing problem. The pricing problem must, however, be able to avoid using forbidden paths as before.

10.5 Computational results

A computational evaluation is performed on a dual 2.66 GHz Intel® Xeon® X5355 machine with 16 GB of RAM. Note that CPU times in the following stem from using one core only.

We have tested three algorithms; the 3BP extended with a heuristic to reach feasible solutions faster [83, 192], the 2BP, and the 2BP'. We implemented all three algorithms using the COIN framework [140] with `ILOG CPLEX 10.2` as LP-solver. Computations concerning the selection of branching candidates and branching children are handled by COIN.

The three solution methods are tested on benchmark instances from the literature [190]: The `Random` instances are randomly generated and the `tg` instances are generated by the Transit Grid generator [189] using topologies from transportation networks. See Table 10.1 for details.

Three different types of tests have been performed. First the impact of using the heuristic from Section 10.3.2 in the 2BP and the 2BP' algorithms is tested. Then the three exact algorithms are computationally evaluated on the proposed instances and results are compared. Finally, we examine if the 3BP and either of the 2BP and 2BP' algorithms give good heuristic solutions by terminating each test run once the root node has been computed (when omitting branching the 2BP and the 2BP' algorithms are identical).

Name	$\|V\|$	$\|E\|$	$\|L\|$
Random5-35	5	35	1
Random10-45	10	45	1
Random15-60	15	60	1
Random20-140	20	140	1
tg10-2	12	40	1
tg20-2	22	80	1
tg40-1	42	154	1
tg40-5	42	154	1
tg80-1	82	322	1
tg100-2	102	400	1
Random10-40	10	40	3
Random11-42	11	42	11
Random20-80	20	80	20
Random22-56	22	56	22

Table 10.1: Sizes of test instances. First column denotes the instance name, then follows the number of vertices, the number of edges, and finally the number of commodities.

10.5.1 Heuristic added to the 2BP and the 2BP' algorithms

The 2BP and the 2BP' algorithms are tested with and without the heuristic from Section 10.3.2. The test results are included in the Appendix in Table 10.8-10.10 for the 2BP and Table 10.11-10.13 for the 2BP', since they show that the heuristic has very little to no impact on the performance of the results. The size of the search tree and the running times are neither worsened nor improved. We include the heuristic as it does not negatively affect the performance and as it may help improve lower bounds.

10.5.2 Optimal approach

The three algorithms are computationally evaluated on the proposed instances. Results for the single-commodity `Random` instances are summarized in Table 10.2 and results for the single-commodity `tg` instances are summarized in Table 10.3. The multi-commodity instances are all of the `Random` type and results are summarized in Table 10.4.

In the tables the first column holds the name of the problem instance, the second column holds the value of k and the third column holds the optimal value. Then follows the size and depth of the search tree, the number of generated variables, the gap in percent between the upper and lower bound, and the time in seconds spent on solving the instance for the 3BP, the 2BP, and the 2BP' algorithms, respectively. If a test run is

marked with "-" then it has run out of memory. If the gap is also marked with "-" then no lower bound was found. The total number of times each algorithm has best performance, is found at the bottom of each table. Also, for each instance the best performance is written in **bold**.

The 2BP algorithm performs much better than the 3BP algorithm for the Minimum Cost MCkFP [83]; however, this is generally not the case for the MCkMFP. Although the number of times the algorithm has best performance is larger for the 2BP, the 3BP algorithm is capable of solving more instances. The change of objective function has a great impact on the problem; the algorithms always try to push as much flow through the network as possible, thus potentially exploiting the somewhat weakly formulated bound on the number of used paths. The formulation has less impact on the minimum cost problem because it may not always be beneficial to increase the number of used paths. The 2BP algorithm suffers from large search trees because of the existence of potentially many solutions using more than k paths per commodity and because the branching scheme allows much symmetry in the branching children. The 2BP algorithm, however, performs somewhat better than the 3BP for the multi-commodity `Random` instances with respect to running times.

The 2BP' algorithm generally performs much better than the 3BP algorithm. Exceptions are `tg40-5`, $k = 4$ and `Random20-80`, $k = 5$, which the 2BP' algorithm spends more time on solving. Furthermore, 2BP' is unable to find an optimal solution for `Random20-80`, $k = 4$. For the far majority of test instances, however, the 2BP' algorithm is capable of finding an optimal solution in little time, even when the 3BP algorithm shows great difficulty. The 2BP' algorithm generally also generates smaller gaps for instances, which are not solved to optimality. Reasons are that the search tree sizes are generally smaller for the 2BP', the number of variables in the master problem is smaller, and much symmetry is eliminated because of the lacking h-indices.

The 2BP' algorithm generally also performs much better than the 2BP algorithm. Exceptions are `Random20-80`, $k = 4, 5,$ and 6 where the 2BP has overall best performance. The reason for the generally superior performance of the 2BP' algorithm is that the branching scheme gives better bounds in the branching children: forcing the use of a path is much stronger than forbidding a path. Also forbidding the use of all paths with positive flow is stronger than forbidding a subset of the paths.

All three algorithms suffer from the same weakness in the formulation, specifically the bounding of the number of used paths per commodity: constraints (10.9) for the 3BP and (10.15) for the 2BP and the 2BP' algorithms. Because the objective is to maximize the total amount of flow, the algorithms are very likely to exceed k paths per commodity whenever the mentioned constraints are not tight. The constraints will rarely be tight, especially when several paths share the same edges causing the corresponding x_p^l/u_p to become much smaller than one. The 2BP' reduces this problem to some extent with the branching cut (10.18).

			3BP					2BP					2BP'				
Problem	**k**	**z***	**size**	**depth**	**vars**	**gap**	**time**	**size**	**depth**	**vars**	**gap**	**time**	**size**	**depth**	**vars**	**gap**	**time**
Random5-35	1	66	1	0	5	0.00%	**0.00**	1	0	5	0.00%	**0.00**	1	0	5	0.00%	**0.00**
	2	128	1	0	14	0.00%	**0.00**	1	0	7	0.00%	**0.00**	1	0	7	0.00%	**0.00**
	3	182	1	0	27	0.00%	0.01	1	0	9	0.00%	**0.00**	1	0	9	0.00%	**0.00**
	4	223	13	6	48	0.00%	0.01	1	0	12	0.00%	**0.00**	1	0	12	0.00%	**0.00**
	5	262	19	9	60	0.00%	0.03	1	0	12	0.00%	**0.00**	1	0	12	0.00%	**0.00**
	6	297	21	10	78	0.00%	0.03	1	0	14	0.00%	0.01	1	0	14	0.00%	**0.00**
	7	326	67	12	98	0.00%	0.10	1	0	14	0.00%	**0.00**	1	0	14	0.00%	**0.00**
	8	326	1	0	104	0.00%	**0.00**	1	0	11	0.00%	0.01	1	0	11	0.00%	**0.00**
Random10-45	1	73	1	0	6	0.00%	**0.00**	1	0	5	0.00%	**0.00**	1	0	5	0.00%	**0.00**
	2	142	5	2	15	0.00%	**0.01**	4	1	9	0.00%	**0.01**	8	2	9	0.00%	**0.01**
	3	209	9	3	33	0.00%	**0.02**	21	3	15	0.00%	0.03	20	3	12	0.00%	**0.02**
	4	260	45	17	68	0.00%	0.08	411	12	24	0.00%	0.56	34	4	20	0.00%	**0.03**
	5	306	369	22	102	0.00%	0.80	23599	18	34	0.00%	44.96	40	4	20	0.00%	**0.07**
	6	345	973	26	137	0.00%	2.90	>427099	>26	39	2.36%	-	135	6	26	0.00%	**0.22**
	7	381	4281	36	219	0.00%	16.55	>354551	>22	46	-%	-	313	8	34	0.00%	**0.64**
	8	413	22985	43	265	0.00%	102.51	>431299	>29	46	2.93%	-	606	9	40	0.00%	**1.31**
	9	429	>110199	>58	380	6.43%	-	>388228	>26	60	-%	-	2507	11	46	0.00%	**5.97**
	10	451	>104999	>57	448	5.74%	-	>456699	>41	74	6.57%	-	2355	12	46	0.00%	**5.91**
Random15-60	1	86	1	0	5	0.00%	**0.00**	1	0	6	0.00%	**0.00**	1	0	6	0.00%	**0.00**
	2	163	1	0	16	0.00%	**0.00**	1	0	8	0.00%	**0.00**	1	0	8	0.00%	**0.00**
	3	221	9	3	34	0.00%	**0.02**	41	6	15	0.00%	0.06	12	2	12	0.00%	**0.02**
	4	248	111	10	70	0.00%	0.32	>100454	>26	50	-%	-	111	6	20	0.00%	**0.22**
	5	268	557	18	101	0.00%	551.83	>176599	>29	52	2.86%	-	322	7	29	0.00%	**0.76**
	6	287	419	21	135	0.00%	1.59	>277801	>31	45	2.74%	-	354	9	30	0.00%	**0.79**
	7	295	19097	35	194	0.00%	72.91	>387565	>23	49	-%	-	836	10	27	0.00%	**1.74**
	8	301	>88799	>47	231	2.90%	-	>413343	>33	55	2.90%	-	4995	11	30	0.00%	**11.32**
	9	306	>153099	>51	229	1.29%	-	>547079	>28	48	-%	-	2263	11	19	0.00%	**4.42**
Random20-140	1	81	1	0	4	0.00%	**0.00**	1	0	5	0.00%	**0.00**	1	0	5	0.00%	**0.00**
	2	158	1	0	14	0.00%	**0.00**	1	0	7	0.00%	**0.00**	1	0	7	0.00%	**0.00**
	3	228	1	0	30	0.00%	0.02	1	0	11	0.00%	**0.00**	1	0	11	0.00%	**0.00**

4	253	9935	31	103	0.00%	75.25	>41444	>42	68	-%	-	90	18	67	0.00%	**1.04**
5	274	>39999	>41	146	1.86%	-	>68299	>66	87	1.86%	-	819	22	51	0.00%	**12.65**
6	294	>30199	>61	184	1.78%	-	>60299	>86	107	1.78%	-	>14106	>32	113	1.78%	-
7	-	>28999	>70	227	1.81%	-	>75894	>46	91	-%	-	>14299	>32	109	1.69%	-
8	319	>30599	>80	267	1.91%	-	>94699	>101	120	1.91%	-	4028	22	29	0.00%	**52.95**
9	325	>39599	>93	315	0.84%	-	>108990	>63	105	-%	-	130	9	25	0.00%	**0.32**
10	327	2907	109	326	0.00%	19.15	>272685	>49	68	0.61%	-	17	3	22	0.00%	**0.02**
11	327	1325	86	301	0.00%	8.75	49	3	22	0.00%	**0.03**	20	5	20	0.00%	**0.03**
Best						**11**					**14**					**36**

Table 10.2: Results from solving the single-commodity `Random` instances exactly.

			3BP					2BP					2BP'				
Problem	**k**	**z***	**size**	**depth**	**vars**	**gap**	**time**	**size**	**depth**	**vars**	**gap**	**time**	**size**	**depth**	**vars**	**gap**	**time**
tg10-2	1	389	1	0	5	0.00%	**0.00**	1	0	4	0.00%	**0.00**	1	0	4	0.00%	**0.00**
	2	557	215	13	26	0.00%	0.21	355	14	10	0.00%	0.21	41	5	11	0.00%	**0.04**
	3	716	553	19	58	0.00%	0.70	39505	20	28	0.00%	32.49	53	5	15	0.00%	**0.06**
	4	815	83	17	52	0.00%	0.10	6	1	8	0.00%	**0.00**	5	1	8	0.00%	**0.00**
	5	815	1	0	40	0.00%	**0.00**	1	0	8	0.00%	**0.00**	1	0	8	0.00%	**0.00**
tg20-2	1	385	1	0	4	0.00%	**0.00**	1	0	4	0.00%	**0.00**	1	0	4	0.00%	**0.00**
	2	643	1	0	10	0.00%	**0.00**	1	0	6	0.00%	**0.00**	1	0	6	0.00%	**0.00**
	3	832	5	2	33	0.00%	0.04	1	0	10	0.00%	**0.00**	1	0	10	0.00%	**0.00**
	4	853	1	0	40	0.00%	0.01	1	0	10	0.00%	**0.00**	1	0	10	0.00%	**0.00**
tg40-1	1	517	1	0	5	0.00%	**0.00**	1	0	5	0.00%	0.01	1	0	5	0.00%	**0.00**
	2	750	5	2	16	0.00%	0.07	4	1	10	0.00%	**0.02**	10	3	12	0.00%	0.07
	3	908	>9999	>40	96	2.61%	-	>83282	>61	94	-%	-	231	11	21	0.00%	**3.32**
	4	994	>7799	>57	143	1.00%	-	>82770	>45	64	-%	-	893	18	33	0.00%	**25.15**
	5	1004	15	7	65	0.00%	0.09	703	27	20	0.00%	1.41	11	2	18	0.00%	**0.03**
	6	1004	1	0	96	0.00%	0.03	29	3	13	0.00%	**0.02**	43	6	13	0.00%	0.07
tg40-5	1	487	1	0	8	0.00%	0.01	1	0	6	0.00%	**0.00**	1	0	6	0.00%	**0.00**
	2	828	>20599	>46	80	4.11%	-	>64248	>45	57	5.70%	-	144	9	23	0.00%	**1.49**
	3	1062	>17299	>59	139	0.28%	-	>77103	>44	65	-%	-	276	8	22	0.00%	**4.20**
	4	1078	181	47	68	0.00%	**0.61**	>148934	>22	50	-%	-	1520	21	22	0.00%	26.53
	5	1078	3	1	90	0.00%	**0.03**	61	4	16	0.00%	0.04	76	20	16	0.00%	1.72
tg80-1	1	549	1	0	7	0.00%	0.04	1	0	6	0.00%	**0.02**	1	0	6	0.00%	**0.02**
	2	984	1591	29	80	0.00%	65.22	2308	22	25	0.00%	59.16	288	11	39	0.00%	**8.72**
	3	1411	>2199	>36	162	3.85%	-	>51476	>49	107	-%	-	1914	10	38	0.00%	**110.38**
tg100-2	1	530	1	0	7	0.00%	0.07	1	0	6	0.00%	0.03	1	0	6	0.00%	**0.02**
	2	1007	3	1	16	0.00%	0.20	1	0	8	0.00%	**0.04**	1	0	8	0.00%	**0.04**
	3	1407	>1099	>31	115	0.39%	-	>29087	>60	113	-%	-	229	6	51	0.00%	**29.14**
	4	1768	>1499	>72	234	1.51%	-	>56256	>40	167	-%	-	2118	9	82	0.00%	**284.41**
Best							**7**					**12**					**23**

Table 10.3: Results from solving the `tg` instances exactly.

			3BP					2BP					2BP'				
Problem	**k**	**z***	**size**	**depth**	**vars**	**gap**	**time**	**size**	**depth**	**vars**	**gap**	**time**	**size**	**depth**	**vars**	**gap**	**time**
Random10-40	1	110	5	2	18	0.00%	0.02	5	2	15	0.00%	**0.00**	4	1	14	0.00%	0.01
	2	194	1	0	26	0.00%	**0.00**	34	5	21	0.00%	0.04	4	1	18	0.00%	0.01
	3	258	183	18	80	0.00%	0.39	213	12	24	0.00%	0.18	50	6	23	0.00%	**0.06**
	4	293	695	36	129	0.00%	2.23	2956	16	41	0.00%	4.25	112	7	32	0.00%	**0.20**
	5	309	1989	30	176	0.00%	7.91	>253716	>25	56	1.24%	**1.06**	561	12	39	0.00%	**1.06**
	6	318	15905	36	253	0.00%	73.84	>610006	>24	59	1.35%	-	1294	13	60	0.00%	**2.63**
	7	321	>153199	>56	286	0.01%	-	>335959	>26	54	-%	-	26182	18	47	0.00%	**57.10**
	8	323	113	37	299	0.00%	**0.52**	2008	14	37	0.00%	1.23	2051	15	36	0.00%	2.43
	9	323	333	49	318	0.00%	1.38	11	1	32	0.00%	**0.01**	18	5	32	0.00%	0.02
Random11-42	1	291	5	2	29	0.00%	0.02	7	3	28	0.00%	**0.01**	7	2	27	0.00%	0.02
	2	343	1	0	50	0.00%	**0.01**	7	2	27	0.00%	**0.01**	6	1	27	0.00%	**0.01**
	3	344	1	0	75	0.00%	**0.00**	1	0	26	0.00%	**0.00**	1	0	26	0.00%	**0.00**
	4	344	1	0	100	0.00%	**0.00**	1	0	26	0.00%	**0.00**	1	0	26	0.00%	**0.00**
Random20-80	1	347	7	3	55	0.00%	0.06	3	1	51	0.00%	**0.02**	7	2	53	0.00%	0.04
	2	553	3	1	100	0.00%	0.03	4	1	50	0.00%	0.02	4	1	51	0.00%	**0.01**
	3	584	1063	24	188	0.00%	6.14	57	7	59	0.00%	**0.16**	1020	16	62	0.00%	3.45
	4	601	5599	33	277	0.00%	40.05	1041	10	60	0.00%	**2.02**	>81550	>548	601	2.01%	-
	5	617	13291	44	340	0.00%	117.96	4363	14	66	0.00%	**7.35**	49695	34	67	0.00%	198.61
	6	621	>48999	>40	380	0.01%	-	3998	11	63	0.00%	**6.42**	32552	29	58	0.00%	100.08
	7	626	413	37	412	0.00%	3.48	17	2	57	0.00%	**0.02**	116	14	57	0.00%	0.22
	8	626	1	0	440	0.00%	0.03	1	0	57	0.00%	**0.01**	1	0	57	0.00%	**0.01**
Random22-56	1	365	9	3	52	0.00%	**0.02**	7	3	42	0.00%	**0.02**	7	2	44	0.00%	**0.02**
	2	389	11	4	81	0.00%	0.02	10	3	42	0.00%	0.02	9	3	42	0.00%	**0.01**
	3	407	1	0	108	0.00%	**0.01**	1	0	41	0.00%	**0.01**	1	0	41	0.00%	**0.01**
	4	407	1	0	144	0.00%	0.01	1	0	41	0.00%	**0.00**	1	0	41	0.00%	**0.00**
Best							**7**					**17**					**14**

Table 10.4: Results from solving the multi-commodity instances exactly.

10.5.3 Heuristic approach

The three exact algorithms presented can be used as heuristics by only computing the root node. This approach does not guarantee a polynomial running time, since an exponential number of columns potentially needs to be added in the root. In practice, however, we expect low running times.

The heuristic usage of the 3BP algorithm is denoted 3HEUR. Because no branching occurs the heuristic usage of the 2BP and the 2BP' algorithms is identical and is denoted 2HEUR. Including the heuristic from Section 10.3.2 in the latter gives the final heuristic denoted 2HEUR'.

All three heuristics are evaluated on the previously proposed instances. Test results for the heuristic use of the exact algorithms are summarized in tables 10.5, 10.6, and 10.7.

The first column of each table holds the name of the problem instance, the second column holds the value of k, and the third column holds the optimal value. Then, follows for each of the algorithms 3HEUR, 2HEUR, and 2HEUR'; the number of iterations, the gap between the heuristic and the optimal value, and the time in seconds spent on solving the instance. An entry marked with "-" indicates that no feasible solution was found. The average number of iterations, gap, and time usage are given at the bottom of each table.

The results show that the 3HEUR algorithm often gives poor heuristic solutions with gaps of up to 94%. For three multi-commodity `Random` instances the 3BP algorithm is even unable to find a feasible solution in the root node. The 2HEUR algorithm generally finds much better solution values than the 3HEUR algorithm. The 2HEUR', however, shows superior performance by solving the majority of the instances to optimality and with the largest gap of those not solved being 20%. All heuristics have very low running times and terminate in less than a second.

			3HEUR			2HEUR			2HEUR'		
Problem	**k**	**z***	**iter.**	**gap**	**time**	**iter.**	**gap**	**time**	**iter.**	**gap**	**time**
Random5-35	1	66	5	0.00	0.00	3	0.00	0.00	3	0.00	0.01
	2	128	7	0.00	0.00	5	0.00	0.00	5	0.00	0.00
	3	182	8	0.00	0.00	7	0.00	0.00	7	0.00	0.00
	4	223	12	16.60	0.00	10	0.00	0.00	10	0.00	0.00
	5	262	12	54.96	0.00	10	0.00	0.00	10	0.00	0.00
	6	297	13	80.37	0.00	12	0.00	0.00	12	0.00	0.00
	7	326	14	54.29	0.00	12	0.00	0.00	12	0.00	0.00
	8	326	13	0.00	0.01	11	0.00	0.00	11	0.00	0.00
Random10-45	1	73	6	0.00	0.00	4	0.00	0.00	4	0.00	0.00
	2	142	7	12.68	0.00	6	12.68	0.00	6	0.00	0.00
	3	209	10	22.00	0.01	10	15.31	0.00	10	0.00	0.00
	4	260	13	65.38	0.01	13	18.08	0.00	13	0.00	0.00
	5	306	15	75.82	0.01	17	46.73	0.00	17	0.00	0.00
	6	345	17	73.91	0.02	19	43.48	0.00	19	0.00	0.00
	7	381	23	76.38	0.02	21	47.77	0.00	21	8.40	0.01
	8	413	24	78.21	0.02	23	48.18	0.00	23	6.78	0.01
	9	429	30	79.02	0.04	30	37.06	0.00	30	1.40	0.00
	10	451	35	80.04	0.05	38	36.59	0.01	38	0.00	0.01
Random15-60	1	86	5	0.00	0.00	6	0.00	0.00	6	0.00	0.00
	2	163	8	0.00	0.01	8	0.00	0.00	8	0.00	0.00
	3	221	10	55.24	0.01	11	85.52	0.00	11	16.74	0.00
	4	248	13	87.50	0.01	18	56.04	0.01	18	10.01	0.00
	5	268	16	57.49	0.02	20	51.49	0.00	20	5.97	0.00
	6	287	18	93.38	0.02	22	50.52	0.01	22	6.62	0.00
	7	295	20	85.05	0.02	23	46.44	0.01	23	13.90	0.00
	8	301	19	59.47	0.01	21	46.84	0.00	21	17.94	0.00
	9	306	18	60.13	0.01	18	39.87	0.00	18	19.93	0.00
Random20-140	1	81	4	0.00	0.00	4	0.00	0.00	4	0.00	0.00
	2	158	7	0.00	0.02	6	0.00	0.00	6	0.00	0.00
	3	228	10	0.00	0.02	10	0.00	0.00	10	0.00	0.00
	4	253	12	82.48	0.03	13	0.00	0.00	13	0.00	0.01
	5	274	16	84.69	0.04	16	1.46	0.01	16	0.00	0.01
	6	294	18	84.69	0.04	24	69.05	0.01	24	3.40	0.01
	9	325	22	86.15	0.04	24	44.92	0.01	24	0.31	0.01
	10	327	21	86.24	0.01	19	3.67	0.00	19	3.67	0.00
	11	327	20	86.24	0.01	19	0.61	0.00	19	0.61	0.00
Average			14	49.40	0.01	15	22.29	<0.01	15	3.21	<0.01

Table 10.5: Results from solving the single-commodity `Random` instances heuristically, where each algorithm terminates after having evaluated the root node only.

			3HEUR			2HEUR			2HEUR'		
Problem	**k**	**z***	**iter.**	**gap**	**time**	**iter.**	**gap**	**time**	**iter.**	**gap**	**time**
tg10-2	1	389	5	0.00	0.00	4	0.00	0.00	4	0.00	0.00
	2	557	6	0.00	0.00	6	0.00	0.00	6	0.00	0.00
	3	716	9	0.00	0.00	10	15.22	0.00	10	0.00	0.00
	4	815	8	0.00	0.00	8	15.83	0.01	8	0.00	0.00
	5	815	8	0.00	0.00	8	0.00	0.00	8	0.00	0.00
tg20-2	1	385	4	0.00	0.00	4	0.00	0.00	4	0.00	0.00
	2	643	5	0.00	0.00	6	0.00	0.00	6	0.00	0.00
	3	832	11	18.03	0.00	10	0.01	0.00	10	0.00	0.01
	4	853	10	0.00	0.01	10	0.00	0.00	10	0.00	0.00
tg40-1	1	517	5	0.00	0.01	5	0.00	0.01	5	0.00	0.01
	2	750	7	72.13	0.01	9	61.33	0.01	9	0.00	0.01
tg40-5	1	487	8	0.00	0.00	6	0.00	0.00	6	0.00	0.01
tg80-1	1	549	7	0.00	0.04	6	0.00	0.02	6	0.00	0.02
	2	984	11	52.74	0.14	14	14.63	0.06	14	0.00	0.06
tg100-2	1	530	7	0.00	0.07	6	0.00	0.02	6	0.00	0.02
	2	1007	8	28.10	0.15	8	0.00	0.04	8	0.00	0.03
Average			7	10.69	0.03	8	6.69	0.01	8	0.00	0.01

Table 10.6: Results from solving the `tg` instances heuristically, where each algorithm terminates after having evaluated the root node only.

			3HEUR			2HEUR			2HEUR'		
Problem	**k**	**z***	**iter.**	**gap**	**time**	**iter.**	**gap**	**time**	**iter.**	**gap**	**time**
Random10-40	1	110	6	31.82	0.00	5	20.00	0.01	5	17.27	0.00
	2	194	6	0.00	0.00	9	60.82	0.00	9	0.00	0.00
	3	258	11	80.62	0.01	11	69.38	0.00	11	6.59	0.00
	4	293	14	66.21	0.02	15	36.52	0.01	15	7.17	0.01
	5	309	16	67.96	0.02	19	34.95	0.00	19	8.41	0.01
	6	318	21	68.89	0.03	25	33.02	0.00	25	5.97	0.01
	7	321	17	84.42	0.02	21	24.61	0.00	21	1.56	0.01
	8	323	21	84.52	0.01	20	22.29	0.00	20	4.34	0.00
	9	323	20	84.52	0.01	21	9.29	0.00	21	0.00	0.00
Random11-42	1	291	6	6.19	0.01	6	6.19	0.00	6	0.00	0.01
	2	343	4	0.00	0.00	5	19.53	0.00	5	4.08	0.00
	3	344	4	0.00	0.00	5	0.00	0.01	5	0.00	0.00
	4	344	4	0.00	0.00	5	0.00	0.00	5	0.00	0.00
Random20-80	1	347	6	25.36	0.01	6	16.14	0.01	6	0.00	0.01
	2	553	7	35.80	0.02	7	15.91	0.01	7	0.00	0.01
	3	584	9	-	0.02	9	7.53	0.00	9	0.00	0.01
	4	601	12	-	0.03	12	7.65	0.01	12	0.00	0.01
	5	617	14	-	0.04	16	4.05	0.02	16	2.27	0.00
	6	621	12	58.29	0.03	14	0.64	0.01	14	0.00	0.01
	7	626	12	58.63	0.03	14	0.96	0.01	14	0.80	0.01
	8	626	12	0.00	0.03	14	0.00	0.01	14	0.00	0.01
Random22-56	1	365	6	0.00	0.01	5	0.00	0.00	5	0.00	0.00
	2	389	6	1.54	0.00	5	1.54	0.00	5	0.00	0.00
	3	407	6	0.00	0.01	5	0.00	0.00	5	0.00	0.01
	4	407	6	0.00	0.00	5	0.00	0.01	5	0.00	0.01
Average*			9	34.31	0.01	11	15.64	<0.01	11	2.34	<0.01

Table 10.7: Results from solving the multi-commodity `Random` instances heuristically, where each algorithm terminates after having evaluated the root node only. Average$^{*)}$ is only over the instances where all heuristics found a feasible solution.

10.6 Conclusion

Two exact solution methods for the MCkMFP problem have been introduced. They are both based on Dantzig-Wolfe decomposition, where the master problem is a 2-index formulation merging paths for commodities into an overall solution. The first solution method was inspired by previous work on The Minimum Cost Multi-Commodity k-splittable Flow Problem [83]. The two methods differ in their branching schemes: the first method forbids subpaths (2BP), while the second forces or forbids the use of certain paths (2BP'). The latter also adds branching cuts to the master problem.

The 2BP and 2BP' algorithms have been implemented and compared with a leading exact algorithm from the literature denoted 3BP. Results showed that the 2BP' algorithm performs significantly better than the 2BP and the 3BP algorithms both with respect to the number of solved instances and with respect to the time usage. The main reason is that using the 2BP' algorithm gives smaller search trees, reduces the number of variables in the master problem, and eliminates some of the symmetry in the solution space.

Terminating the computations after having evaluated the root node transforms the 3BP and the 2BP/2BP' algorithms into heuristics denoted 3HEUR and 2HEUR, respectively. Because no branching occurs in this heuristic use, the 2BP and the 2BP' algorithms become identical. Test results for this approach showed that the 3HEUR does not perform well, with the majority of the solution values having gaps of up to 94%. The 2HEUR algorithm, however, showed very promising performance when including a greedy heuristic, which transforms some fractional solutions into feasible solutions. In most cases optimal solutions were found and the average solution gaps never exceeded 4%. Both heuristics terminate in less than a second for all tested instances.

All algorithms suffer from weak formulations for bounding the number of used paths per commodity. We believe that future work should concentrate on tightening these constraints. This could be done by somehow reformulating the problem or by adding cuts. We believe that the focus should be on cuts violated in the edge-based model or the original master problem. Future work could also concentrate on finding better branching strategies for the 2-index formulation in order to further reduce the size of the search tree.

Acknowledgement
We would like to thank GlobalConnect A/S for their support of this work.

Appendix

10.7 2BP without and with pruning heuristic

			2BP					2BP+heur				
Problem	**k**	**z***	**size**	**depth**	**vars**	**gap**	**time**	**size**	**depth**	**vars**	**gap**	**time**
Random5-35	1	66	1	0	5	0.00%	0.00	1	0	5	0.00%	0.00
	2	128	1	0	7	0.00%	0.00	1	0	7	0.00%	0.00
	3	182	1	0	9	0.00%	0.01	1	0	9	0.00%	0.00
	4	223	1	0	12	0.00%	0.00	1	0	12	0.00%	0.00
	5	262	1	0	12	0.00%	0.00	1	0	12	0.00%	0.00
	6	297	1	0	14	0.00%	0.00	1	0	14	0.00%	0.01
	7	326	1	0	14	0.00%	0.01	1	0	14	0.00%	0.00
	8	326	1	0	13	0.00%	0.00	1	0	13	0.00%	0.01
Random10-45	1	73	1	0	5	0.00%	0.00	1	0	5	0.00%	0.00
	2	142	4	1	9	0.00%	0.00	4	1	9	0.00%	0.01
	3	209	21	3	15	0.00%	0.03	21	3	15	0.00%	0.03
	4	260	411	12	24	0.00%	0.56	411	12	24	0.00%	0.56
	5	306	23599	18	34	0.00%	45.86	23599	18	34	0.00%	44.96
	6	345	>427099	>26	39	2.36%	-	>427099	>26	39	2.36%	-
	7	381	>349959	>21	46	-%	-	>354551	>22	46	-%	-
	8	413	>427699	>29	46	2.93%	-	>431299	>29	46	2.93%	-
	9	429	>388228	>26	60	-%	-	>388228	>26	60	-%	-
	10	451	>456699	>41	74	6.56%-		>456699	>41	74	6.57%	-
Random15-60	1	86	1	0	6	0.00%	0.00	1	0	6	0.00%	0.00
	2	163	1	0	8	0.00%	0.00	1	0	8	0.00%	0.00
	3	221	41	6	15	0.00%	0.06	41	6	15	0.00%	0.06
	4	248	>101109	>27	50	-%	-	>100454	>26	50	-%	-
	5	268	>176814	>29	52	2.86%	-	>176599	>29	52	2.86%	-
	6	287	>277515	>31	46	2.74%	-	>277801	>31	45	2.74%	-
	7	295	>387565	>23	49	-%	-	>387565	>23	49	-%	-
	8	301	>406168	>35	59	2.90%	-	>413343	>33	55	2.90%	-
	9	306	>568629	>24	45	-%	-	>547079	>28	48	-%	-
Random20-140	1	81	1	0	5	0.00%	0.00	1	0	5	0.00%	0.00
	2	158	1	0	7	0.00%	0.00	1	0	7	0.00%	0.00
	3	228	1	0	11	0.00%	0.00	1	0	11	0.00%	0.00
	4	253	>41444	>42	68	-%	-	>41444	>42	68	-%	-
	5	274	>68299	>66	87	1.86%	-	>68299	>66	87	1.86%	-
	6	294	>60299	>86	107	1.78%	-	>60299	>86	107	1.78%	-
	7	-	>76094	>46	90	-%	-	>75894	>46	91	-%	-
	8	319	>94699	>101	120	1.91%	-	>94699	>101	120	1.91%	-
	9	325	>114289	>47	94	-%	-	>108990	>63	105	-%	-
	10	327	>271899	>48	65	0.61%	-	>272685	>49	68	0.61%	-
	11	327	49	3	22	0.00%	0.03	49	3	22	0.00%	0.03

Table 10.8: Results from solving the single commodity `Random` instances without and with the pruning heuristic.

			2BP					2BP+heur				
Problem	**k**	**z***	**size**	**depth**	**vars**	**gap**	**time**	**size**	**depth**	**vars**	**gap**	**time**
tg10-2	1	389	1	0	4	0.00%	0.00	1	0	4	0.00%	0.00
	2	557	355	14	10	0.00%	0.22	355	14	10	0.00%	0.21
	3	716	40981	19	20	0.00%	33.29	39505	20	28	0.00%	32.49
	4	815	6	1	8	0.00%	0.00	6	1	8	0.00%	0.00
	5	815	1	0	8	0.00%	0.00	1	0	8	0.00%	0.00
tg20-2	1	385	1	0	4	0.00%	0.01	1	0	4	0.00%	0.00
	2	643	1	0	6	0.00%	0.00	1	0	6	0.00%	0.00
	3	832	1	0	10	0.00%	0.01	1	0	10	0.00%	0.00
	4	853	1	0	10	0.00%	0.01	1	0	10	0.00%	0.00
tg40-1	1	517	1	0	5	0.00%	0.00	1	0	5	0.00%	0.01
	2	750	4	1	10	0.00%	0.03	4	1	10	0.00%	0.02
	3	908	>80455	>50	83	-%	-	>83282	>61	94	-%	-
	4	994	>74285	>43	63	-%	-	>82770	>45	64	-%	-
	5	1004	619	24	23	0.00%	1.14	703	27	20	0.00%	1.41
	6	1004	29	3	13	0.00%	0.02	29	3	13	0.00%	0.02
tg40-5	1	487	1	0	6	0.00%	0.00	1	0	6	0.00%	0.00
	2	828	>63948	>48	60	5.70%	-	>64248	>45	57	5.70%	-
	3	1062	>81691	>48	66	-%	-	>77103	>44	65	-%	-
	4	1078	>150030	>22	49	-%	-	>148934	>22	50	-%	-
	5	1078	61	4	16	0.00%	0.04	61	4	16	0.00%	0.04
tg80-1	1	549	1	0	6	0.00%	0.02	1	0	6	0.00%	0.02
	2	984	2215	22	27	0.00%	56.57	2308	22	25	0.00%	59.16
	3	1411	>52274	>44	102	-%	-	>51476	>49	107	-%	-
tg100-2	1	530	1	0	6	0.00%	0.02	1	0	6	0.00%	0.03
	2	1007	1	0	8	0.00%	0.04	1	0	8	0.00%	0.04
	3	1407	>31616	>64	108	-%	-	>29087	>60	113	-%	-
	4	1768	>58467	>26	168	-%	-	>56256	>40	167	-%	-

Table 10.9: Results from solving the `tg` instances without and with the pruning heuristic.

			2BP					2BP+heur				
Problem	**k**	**z***	**size**	**depth**	**vars**	**gap**	**time**	**size**	**depth**	**vars**	**gap**	**time**
Random10-40	1	110	5	2	15	0.00%	0.01	5	2	15	0.00%	0.01
	2	194	34	5	21	0.00%	0.03	34	5	21	0.00%	0.04
	3	258	213	12	24	0.00%	0.15	213	6	12	0.00%	0.18
	4	293	2956	16	41	0.00%	4.24	2956	16	41	0.00%	4.25
	5	309	>232916	>25	56	1.24%	-	>253716	>25	56	1.24%	-
	6	318	>610056	>24	59	1.35%		>610005	>24	59	1.35%	-
	7	321	>329360	>27	54	-%	-	>335959	>26	54	-%	-
	8	323	2008	14	37	0.00%	1.20	2008	14	37	0.00%	1.23
	9	323	11	1	32	0.00%	0.01	11	1	32	0.00%	0.01
Random11-42	1	291	7	3	28	0.00%	0.02	7	3	28	0.00%	0.01
	2	343	7	2	27	0.00%	0.01	7	2	27	0.00%	0.01
	3	344	1	0	26	0.00%	0.00	1	0	26	0.00%	0.00
	4	344	1	0	26	0.00%	0.00	1	0	26	0.00%	0.00
Random20-80	1	347	3	1	51	0.00%	0.03	3	1	51	0.00%	0.02
	2	553	4	1	50	0.00%	0.02	4	1	50	0.00%	0.02
	3	584	57	7	59	0.00%	0.16	57	7	59	0.00%	0.16
	4	601	1041	10	60	0.00%	2.02	1041	10	60	0.00%	2.02
	5	617	4363	14	66	0.00%	7.21	4363	14	66	0.00%	7.35
	6	621	3998	11	63	0.00%	6.40	3998	11	63	0.00%	6.42
	7	626	17	2	57	0.00%	0.03	17	2	57	0.00%	0.02
	8	626	1	0	57	0.00%	0.01	1	0	57	0.00%	0.01
Random22-56	1	365	7	3	42	0.00%	0.02	7	3	42	0.00%	0.02
	2	389	10	3	42	0.00%	0.02	10	3	42	0.00%	0.02
	3	407	1	0	41	0.00%	0.00	1	0	41	0.00%	0.01
	4	407	1	0	41	0.00%	0.00	1	0	41	0.00%	0.00

Table 10.10: Results from solving the multicommodity instances without and with the pruning heuristic.

10.8 2BP' without and with pruning heuristic

			2BP'					2BP'+heur				
Problem	**k**	**z***	**size**	**depth**	**vars**	**gap**	**time**	**size**	**depth**	**vars**	**gap**	**time**
Random5-35	1	66	1	0	5	0.00%	0.00	1	0	5	0.00%	0.00
	2	128	1	0	7	0.00%	0.01	1	0	7	0.00%	0.00
	3	182	1	0	9	0.00%	0.00	1	0	9	0.00%	0.00
	4	223	1	0	12	0.00%	0.00	1	0	12	0.00%	0.00
	5	262	1	0	12	0.00%	0.00	1	0	12	0.00%	0.00
	6	297	1	0	14	0.00%	0.00	1	0	14	0.00%	0.00
	7	326	1	0	14	0.00%	0.00	1	0	14	0.00%	0.00
	8	326	1	0	13	0.00%	0.00	1	0	11	0.00%	0.00
Random10-45	1	73	1	0	5	0.00%	0.00	1	0	5	0.00%	0.00
	2	142	8	2	9	0.00%	0.01	8	2	9	0.00%	0.01
	3	209	20	3	12	0.00%	0.02	20	3	12	0.00%	0.02
	4	260	34	4	20	0.00%	0.04	34	4	20	0.00%	0.03
	5	306	40	4	20	0.00%	0.06	40	4	20	0.00%	0.07
	6	345	98	5	26	0.00%	0.18	135	6	26	0.00%	0.22
	7	381	272	7	31	0.00%	0.57	313	8	34	0.00%	0.64
	8	413	602	8	39	0.00%	1.28	606	9	40	0.00%	1.31
	9	429	2549	12	45	0.00%	6.11	2507	11	46	0.00%	5.97
	10	451	2364	10	54	0.00%	6.04	2355	12	46	0.00%	5.91
Random15-60	1	86	1	0	6	0.00%	0.00	1	0	6	0.00%	0.00
	2	163	1	0	8	0.00%	0.01	1	0	8	0.00%	0.00
	3	221	12	2	14	0.00%	0.02	12	2	12	0.00%	0.02
	4	248	102	5	20	0.00%	0.20	111	6	20	0.00%	0.22
	5	268	316	8	22	0.00%	0.71	322	7	29	0.00%	0.76
	6	287	446	10	30	0.00%	1.04	354	9	30	0.00%	0.79
	7	295	1375	11	35	0.00%	2.88	836	10	27	0.00%	1.74
	8	301	4859	11	29	0.00%	10.73	4995	11	30	0.00%	11.32
	9	306	1868	10	19	0.00%	3.67	2263	11	19	0.00%	4.42
Random20-140	1	81	1	0	5	0.00%	0.00	1	0	5	0.00%	0.00
	2	158	1	0	7	0.00%	0.00	1	0	7	0.00%	0.00
	3	228	1	0	11	0.00%	0.01	1	0	11	0.00%	0.00
	4	253	84	17	18	0.00%	0.89	90	18	67	0.00%	1.04
	5	274	894	22	70	0.00%	14.30	819	22	51	0.00%	12.65
	6	294	>13588	>30	115	1.78%	-	>14106	>32	113	1.78%	-
	7	-	>13442	>33	116	1.69%	-	>14299	>32	109	1.69%	-
	8	319	2100	21	24	0.00%	27.73	4028	22	29	0.00%	52.95
	9	325	121	10	25	0.00%	0.24	130	9	25	0.00%	0.32
	10	327	17	3	22	0.00%	0.02	17	3	22	0.00%	0.02
	11	327	20	5	20	0.00%	0.03	20	5	20	0.00%	0.03

Table 10.11: Results from solving the single commodity `Random` instances without and with the pruning heuristic.

			2BP'					2BP'+heur				
Problem	**k**	**z***	**size**	**depth**	**vars**	**gap**	**time**	**size**	**depth**	**vars**	**gap**	**time**
tg10-2	1	389	1	0	4	0.00%	0.00	1	0	4	0.00%	0.00
	2	557	37	5	12	0.00%	0.04	41	5	11	0.00%	0.04
	3	716	58	5	14	0.00%	0.06	53	5	15	0.00%	0.06
	4	815	5	1	8	0.00%	0.00	5	1	8	0.00%	0.00
	5	815	1	0	8	0.00%	0.00	1	0	8	0.00%	0.00
tg20-2	1	385	1	0	4	0.00%	0.00	1	0	4	0.00%	0.00
	2	643	1	0	6	0.00%	0.00	1	0	6	0.00%	0.00
	3	832	1	0	10	0.00%	0.00	1	0	10	0.00%	0.00
	4	853	1	0	10	0.00%	0.00	1	0	10	0.00%	0.00
tg40-1	1	517	1	0	5	0.00%	0.00	1	0	5	0.00%	0.00
	2	750	10	3	12	0.00%	0.07	10	3	12	0.00%	0.07
	3	908	175	9	21	0.00%	2.26	231	11	21	0.00%	3.32
	4	994	776	16	31	0.00%	19.96	893	18	33	0.00%	25.15
	5	1004	26	5	17	0.00%	0.08	11	2	18	0.00%	0.03
	6	1004	43	6	13	0.00%	0.08	43	6	13	0.00%	0.07
tg40-5	1	487	1	0	6	0.00%	0.00	1	0	6	0.00%	0.00
	2	828	166	9	25	0.00%	1.76	144	9	23	0.00%	1.49
	3	1062	308	8	32	0.00%	5.41	276	8	22	0.00%	4.20
	4	1078	2292	22	22	0.00%	45.58	1520	21	22	0.00%	26.53
	5	1078	72	19	16	0.00%	1.41	76	20	16	0.00%	1.72
tg80-1	1	549	1	0	6	0.00%	0.01	1	0	6	0.00%	0.02
	2	984	351	11	39	0.00%	11.06	288	11	39	0.00%	8.72
	3	1411	1905	9	38	0.00%	113.39	1914	10	38	0.00%	110.38
tg100-2	1	530	1	0	6	0.00%	0.03	1	0	6	0.00%	0.02
	2	1007	1	0	8	0.00%	0.04	1	0	8	0.00%	0.04
	3	1407	240	7	10	0.00%	31.22	229	6	51	0.00%	29.14
	4	1768	2367	11	80	0.00%	293.28	2118	9	82	0.00%	284.41

Table 10.12: Results from solving the `tg` instances without and with the pruning heuristic.

			2BP'					2BP'+heur				
Problem	**k**	**z***	**size**	**depth**	**vars**	**gap**	**time**	**size**	**depth**	**vars**	**gap**	**time**
Random10-40	1	110	4	1	14	0.00%	0.00	4	1	14	0.00%	0.01
	2	194	4	1	18	0.00%	0.01	4	1	18	0.00%	0.01
	3	258	51	6	23	0.00%	0.05	50	6	23	0.00%	0.06
	4	293	150	7	31	0.00%	0.28	112	7	32	0.00%	0.20
	5	309	513	12	34	0.00%	1.00	561	12	39	0.00%	1.06
	6	318	3209	15	48	0.00%	6.16	1294	13	60	0.00%	2.63
	7	321	25691	19	50	0.00%	56.47	26182	18	47	0.00%	57.10
	8	323	2046	16	34	0.00%	2.44	2051	15	36	0.00%	2.43
	9	323	18	5	32	0.00%	0.02	18	5	32	0.00%	0.02
Random11-42	1	291	7	2	27	0.00%	0.01	7	2	27	0.00%	0.02
	2	343	6	1	27	0.00%	0.01	6	1	27	0.00%	0.01
	3	344	1	0	26	0.00%	0.00	1	0	26	0.00%	0.00
	4	344	1	0	26	0.00%	0.00	1	0	26	0.00%	0.00
Random20-80	1	347	7	2	54	0.00%	0.04	7	2	53	0.00%	0.04
	2	553	4	1	52	0.00%	0.02	4	1	51	0.00%	0.01
	3	584	637	14	64	0.00%	2.44	1020	16	62	0.00%	3.45
	4	601	>82274	>547		2.01%	-	>81550	>548	601	2.01%	-
	5	617	>83907	>35	87	0.95%	-	49695	34	67	0.00%	198.61
	6	621	25788	31	57	0.00%	74.55	32552	29	58	0.00%	100.08
	7	626	104	14	57	0.00%	0.19	116	14	57	0.00%	0.22
	8	626	1	0	57	0.00%	0.01	1	0	57	0.00%	0.01
Random22-56	1	365	4	1	41	0.00%	0.01	7	2	44	0.00%	0.02
	2	389	9	3	41	0.00%	0.02	9	3	42	0.00%	0.01
	3	407	1	0	41	0.00%	0.00	1	0	41	0.00%	0.01
	4	407	1	0	41	0.00%	0.00	1	0	41	0.00%	0.00

Table 10.13: Results from solving the multicommodity instances without and with the pruning heuristic.

Part IV

Conclusion

CHAPTER 11

Conclusion

11.1 Summary

This thesis has investigated the scheduling problem in grid computing where network constraints are taken into account. The thesis also considered the data routing problem in multi-protocol label switching, which can be applied in grid computing context. The scheduling problem in grid computing has been solved where components are connected through a packet switched network (e.g. the internet) and where components are connected through an optical network. The main conclusions of the thesis are:

- The scheduling problem in grid computing using a packet switched network can be solved to optimality in little time for problems with up to a 1000 jobs and resources.
- The scheduling problem in grid computing using an optical network should be solved heuristically. The data transmission problem becomes $\mathcal{NP}$-hard, which complicates exact solution procedures. Proposed heuristics have small running times and a low solution value gap of 3% on average.
- Operations research can successfully be used in real-life grid systems to reduce network traffic. The Nordic DataGrid Facility can reduce their maximal link load

with 20% by using an optimal job placement. Introducing cache would reduce the maximal link load with another 15%.

- The proposed algorithms for the multi-protocol label switching problem (also denoted the multi-commodity k-splittable flow problem) outperform previous work in the literature. The algorithms reveal that the two main bottlenecks of the problem is symmetry in the solution space and a somewhat poor bound on the number of used paths per commodity.

Details on the results in this thesis are described in the following, where each chapter is briefly summarized.

The offline scheduling problem where components are connected through a packet switched network was solved in Chapter 4. The exact solution approach solved all tested instances within minutes. The algorithm was based on branch-and-price, where the pricing problem assigned a job to a resource and found a way of sending data. The master problem merged these "sub-schedules" into an overall solution. Only violated constraints were included in the master problem. Furthermore, stabilized column generation reduced the number of generated columns significantly.

The offline grid scheduling problem was extended into having an underlying optical network. This complicated routing, because sending data through the network became the $\mathcal{NP}$-hard Routing and Wavelength Assignment Problem (RWA). The RWA consists of routing data through a network using a wavelength on each link, such that no two data transmissions use the same wavelength on the same link. Much work has been conducted on the RWA problem in the literature. Hence a survey of the most common solution algorithms was presented in Chapter 5. The survey included a discussion of theoretical running times and of practical experiments of the proposed solution methods.

The offline grid scheduling where components are connected through an optical network was solved in Chapter 6. An exact branch-and-price algorithm was proposed, but because the pricing problem became the $\mathcal{NP}$-hard RWA-problem, the exact solution approach suffered from large time usage. A number of greedy heuristics were also presented. The heuristics consisted of combining grid heuristics with RWA heuristics. The grid heuristics concentrated on placing jobs on resources according to criteria, such as arrival time, execution time, finish time, etc. The RWA-heuristics found ways of sending data through the optical network. The proposed heuristics all solved the tested instances within minutes and the best heuristic setting had a 3% solution value gap on average.

The final paper on grid scheduling concerned the network topology from the Nordic DataGrid Facility (NDGF). In Chapter 7 we analyzed and formalized the grid network,

grid sites and the jobs to execute. A number of scenarios relevant in this real-life grid context were analyzed and incorporated into the mathematical formulation. The goal was to minimize the maximal link load. Results showed that using an optimal job placement, the maximal link load was reduced from 4.4 Gbps to 3.5 Gbps. Introducing caches reduced the link load further to 3.0 Gbps. The results give a good indication of where and when bottlenecks occur in the NDGF network. Thus the results are not only used for deciding how to distribute jobs, but also as a strategic tool for future expansions.

Part III concentrated on solving the multi-protocol label switching problem, which in operations research is also denoted the multi-commodity k-splittable flow problem. The problem consists of sending a number of commodities through a network, where each commodity cannot use more then k paths. In Chapter 9 the minimum cost version of the problem was considered. A new decomposition was presented in parallel with Truffot and Duhamel [190]. We also presented a corresponding new branch-and-price algorithm, where the pricing problem generated a path for each commodity. Branching forbade certain edge sequences, thus the pricing problem became the polynomial shortest path problem with forbidden sub-paths. The algorithm outperformed the exact algorithms from the literature.

Applying this branch-and-price algorithm to the maximum flow version of the problem, however, did not yield good results. In Chapter 10 we showed that number of paths to forbid during branching explodes for this problem. Hence we proposed a new branching strategy, which either forced or forbade usage of certain paths. The new algorithm outperformed our branch-and-price algorithm from Chapter 9 and the exact algorithms from the literature.

11.2 Directions for future research

This thesis considered a number of different problems in the context of grid scheduling. Each of these problems can be investigated further, both independently and in grid computing context. The grid scheduling problem supports advance reservations, queue emptying and provides an alternative to the online scheduling approach. Future research on the offline problem where grid components are connected through a packet switched network, could focus either on even better exact solution approaches or on better heuristics. The proposed branch-and-price algorithm could be extended with cutting planes. Recent research indicates that cutting planes can boost the performance of exact solvers [57] and this could very well also be the case for the offline scheduling problem. New heuristics could focus more on taking data placement into account when placing jobs on resources. When changing the underlying network topology into being optical, the data transmission problem is complicated significantly and becomes

$\mathcal{NP}$-hard. The proposed exact algorithm suffers from this, because its pricing problem is $\mathcal{NP}$-hard and difficult to solve to optimality. Future work could concentrate on different decompositions, where the pricing problem would be more tractable.

In our work, we assumed that the optical network was dedicated for the grid computing system. The grid administrators have thus made some decisions on which fibers to rent where and when. To support these difficult decisions, network design could be taken into account when solving the scheduling problem. In this way, a compromise between a good schedule and the cost of setting up the optical network could be reached.

The multi-commodity k-splittable flow problem has successfully been solved to optimality in this thesis. However, the proposed algorithms still have room for improvement. The symmetry problems and cumbersome branching could possibly be reduced by adding cuts to the master problem. Another possibility is to consider new decompositions of the problem with special emphasis on tightening the bounds on the number of used paths per commodity.

CHAPTER 12

Summary (in Danish)

Denne afhandling omhandler planlægning af data transmission i grid computing. Grid computing består af en række computere, som arbejder sammen om at løse et stort problem. Computerne kan være placeret over hele verden og er forbundet via et netværk som for eksempel internettet. Et grid har en eller flere centrale servere, som bestemmer hvilke computere, der skal køre hvilke programmer (delproblemer) og hvornår. Serverne beregner planer for aktiviteten, men som det er nu, medtages data transmission enten slet ikke eller kun i ringe grad i beregningerne. Det er et problem, fordi nogle programmer kræver meget data og programmet kan ikke køres før alt data er ankommet til computeren. Programkørslen kan derfor blive forsinket, hvilket ødelægger de planer, som grid serverne har beregnet. Resultatet er, at grid systemet bliver ustabilt.

Denne afhandling foreslår at medtage data transmission i beregning af planer. Planlægningsproblemet er $\mathcal{NP}$-hårdt og derfor svært at løse. I afhandlingen foreslås flere måder at gribe planlægningsproblemet an. Desuden foreslås løsningsmetoder til data transmissionsproblemet, hvor større klumper af data sendes af sted ad gangen. Fremgangsmåden til at løse planlægningsproblemet og data transmissionsproblemet er anvendt matematik, også benævnt operationsanalyse. De optimale løsningsmetoder i afhandlingen er baseret på Dantzig-Wolfe dekomposition, som opdeler problemet i mindre bidder. De heuristiske metoder i afhandlingen er alle grådige. Det videnskabelige bidrag er fordelt over 5 artikler. De første 3 behandler planlægning af data transmission i grid computing sammenhæng. De sidste 2 artikler behandler data transmission problemet, hvor større mængder af data sendes af sted ad gangen.

I kapitel 4 foreslås en metode til at finde en optimal plan for kørsel af programmer i grid computing med hensyn til data transmission. Det er antaget at computerne i grid systemet er forbundet gennem et netværk med samme funktionalitet som internettet. Løsningsmetoden er baseret på Dantzig-Wolfe dekomposition, hvor planlægningsproblemet opdeles i en række delproblemer. Hvert delproblem sørger for at planlægge kørsel af et givent program på en given computer og kan løses i polynomiel tid. En række forbedringer mindsker størrelsen på hovedproblemet og sørger for at delproblemerne giver bedre løsninger. Derved kan metoden løse alle probleminstanser hurtigt - de fleste i løbet af få sekunder og de sværeste på under tre minutter.

I kapitel 6 behandles planlægningsproblemet, når grid computere er forbundet via lyslederfibre (eller optiske fibre). I et optisk netværk er der særlige betingelser til data transmissioner, fordi der er visse hardware begrænsninger. Hver data transmission bliver sendt af sted på en lysfrekvens - hver frekvens i en fiber må højest anvendes af én data transmission. En analyse og oversigt over arbejde på det $\mathcal{NP}$-hårde data transmissionsproblem i optiske netværk - kaldet *The Routing and Wavelength Assignment Problem* - er givet i kapitel 5. Det tilhørende planlægningsproblem i grid computing er som skrevet behandlet i kapitel 6. To forskellige fremgangsmåder analyseres. Den første metode er optimal og baseret på Dantzig-Wolfe dekomposition. Det $\mathcal{NP}$-hårde data transmissionsproblem gør dog, at den optimale løsningsmetode er meget tidskrævende. Derfor foreslår kapitlet også heuristikker til hurtigt at finde en løsning, uden at give nogen garantier for hvor god løsningen er. Den bedste heuristiske indstilling giver gode løsninger, der gennemsnitligt ligger 3% fra de optimale løsninger.

I kapitel 7 løses planlægningsproblemet i Nordic DataGrid Facility (NDGF). Grid systemet og tilhørende netværk anvendt af NDGF analyseres og formaliseres til en matematisk model. Forskellige scenarier analyseres, formaliseres og løses med henblik på at mindske det maksimale netværksforbrug. Det maksimale netværksforbrug nedsættes med cirka 20% (dvs. 900 Mbps) ved at ændre placeringer af programmer. Desuden viser kapitlet at ved tilførsel af flere ressourcer til grid systemet, så kan det maksimale netværksforbrug reduceres med yderligere 15% (500 Mbps).

De sidste to videnskabelige bidrag i denne afhandling løser data transmissionsproblemet, når *Multi-Protocol Label Switching* anvendes. Problemet går ud på at sende data fra en række startpunkter til en række slutpunkter, således at kantkapaciteter overholdes og således at hver transmission bruger højest k ruter. Problemet er $\mathcal{NP}$-hårdt. I kapitel 9 løses problemet, hvor omkostningerne for at sende al data gennem netværket minimeres. Arbejde i litteraturen viser, at der er problemer med meget symmetri i løsningsrummet. Vi foreslår en Dantzig-Wolfe dekomposition, som eliminerer meget af den symmetri, og en dertilhørende branch-and-price algoritme. Pricing problemet er et polynomielt korteste vej problem med forbudte delstier og branching strategien

forbyder brug af visse delstier. Den nye algoritme yder bedre end arbejde fra litteraturen. I kapitel 10 løses data transmissionsproblemet, hvor mængden af transmitteret data søges maksimeret. Branch-and-price algoritmen fra forrige kapitel fungerer ikke tilfredsstillende, når mængden af data ønskes maksimeret, fordi antallet af delstier, som kan forbydes i branching, eksploderer. Derfor foreslår vi en ny branch-and-price algoritme, hvor branching strategien enten tvinger eller forbyder brug af stier. Den nye algoritme yder rigtigt godt og udkonkurrerer tilsvarende arbejde fra litteraturen.

Bibliography

[1] J. H. Abawajy. Adaptive hierarchical scheduling policy for enterprise grid computing systems. *Journal of Network and Computer Applications*, 32(3):770–779, 2009.

[2] ABC@home. http://abcathome.com/, 2009.

[3] V. Agarwal, G. Dasgupta, K. Dasgupta, A. Purohit, and B. Viswanathan. Deco: Data replication and execution co-scheduling for utility grids. In *ICSOC 2006, LNCS 4294*, pages 52–65, 2006.

[4] A. Aggarwal, A. Bar-Noy, R. Ramaswami, B. Schieber, and M. Sudan. Efficient routing and scheduling algorithms for optical networks. In *Proceedings of SIAM symp. On Discrete Algorithms 93*, pages 412 – 423, 1994.

[5] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Network Flows: Theory, Algorithms, and Applications*. Prentice-Hall Inc., 1993.

[6] M. Ali, B. Ramamurthy, and J. S. Deogun. Routing and wavelength assignment (rwa) with power considerations in all-optical wavelength-routed networks. In *Global Telecommunications Conference - Globecom'99*, 1999.

[7] F. Alvelos. *Branch-and-Price and Multicommodity Flows*. PhD thesis, Universidade do Minho, 2005.

[8] F. Alvelos and J. M. V. de Carvalho. Comparing branch-and-price algorithms for the unsplittable multicommodity flow problem. In *International Network Optimization Conference*, pages 7–12, 2003.

[9] C. Anderlik, A. R. Gregersen, J. Kleist, A. Peters, and P. Saiz. Alice - arc integration. In *International Conference on Computing in High Energy and Nuclear Physics (CHEP'07)*, Journal of Physics: Conference Series (JPCS) (2008), 119(6), pages 1–6, 2007.

[10] R. Andersen and B. Vinter. Direct application access to grid storage. *Concurrency and Computation: Practice and Experience*, 19(9):1287–1298, 2007.

[11] D. P. Anderson. BOINC: A system for public-resource computing and storage. In *5th IEEE/ACM International Workshop on Grid Computing*, 2004.

[12] D. P. Anderson, J. Cobb, E. Korpela, M. Lebofsky, and D. Werthimer. Seti@home: An experiment in public-resource computing. *Communications of the ACM*, 45(11):56–61, 2002.

[13] A. Anjum. *Data Intensive and Network Aware (DIANA) Grid Scheduling*. PhD thesis, The Faculty of Computing, Engineering and Mathematical Sciences, University of the West of England, 2006.

[14] D. Applegate, R. Bixby, V. Chvatal, and B. Cook. Finding cuts in the TSP. Technical Report 95-05, Center for Discrete Mathematics and Theoretical Computer Science (DIMACS), Rutgers University, Piscatawy, NJ, 1995.

[15] D. Applegate, R. Bixby, V. Chvatal, and B. Cook. Implementing the Dantzig-Fulkersen-Johnson algorithm for large traveling salesman problems. *Math. Program.*, 97:91–153, 2003.

[16] A. Arteta, B. Barán, and D. Pinto. Routing and wavelength assignment over WDM optical networks: a comparison between MOACOs and classical approaches. In *Applications, Technologies, Architectures, and Protocols for Computer Communication, Proceedings of the 4th international IFIP/ACM Latin American conference on Networking*, 2007.

[17] G. Avellino, S. Beco, B. Cantalupo, A. Maraschini, F. Pacini, M. Sottilaro, A. Terracina, D. Colling, F. Giacomini, E. Ronchieri, A. Gianelle, M. Mazzucato, R. Peluso, M. Sgaravatto, A. Guarise, R. Piro, A.Werbrouck, D. Kouril, A. Krenek, L. Matyska, M. Mulac, J. Posisiil, M. Ruda, Z. Salvet, J. Sitera, J. Skrabal, M. Vocu, M. Mezzadri, F. Prelz, S. Monforte, and M. Pappalardo. The datagrid workload management system: Challenges and results. *Journal of Grid Computing*, 2:353–367, 2004.

[18] B. Awerbuch and T. Leighton. *Multicommodity flows: A survey of recent research*, volume 762 of *Lecture Notes in Computer Science*, pages 297–302. Springer Berlin / Heidelberg, 1993.

[19] G. Baier, E. Köhler, and M. Skutella. On the k-splittable flow problem. In *10th Annual European Symposium on Algorithms*, pages 101–113, 2002.

[20] M. Baker, R. Buyya, and D. Laforenza. Grids and grid technologies for wide-area distributed computing. *Software -practice and experience*, 32(15):1437–1466, 2002.

[21] D. Banerjee and B. Mukherjee. Wavelength-routed optical networks: Linear formulation, resource budgeting tradeoffs, and a reconfiguration study. *IEEE/ACM Transactions on Networking*, 8:598–607, 2000.

[22] N. Banerjee, V. Metha, and S. Pandey. A genetic algorithm approach for solving the routing and wavelength assignment problem in WDM network. In *3rd IEEE/IEE International Conference on Networking, ICN 2004, Paris*, pages 70–78, 2004.

[23] R. Baraglia, R. Ferrini, N. Tonellotto, L. Ricci, and R. Yahyapour. A launch-time scheduling heuristics for parallel applications on wide area grids. *Journal of Grid Computing*, 6(2):159–175, 2008.

[24] A. Baranovski, G. Garzoglio, I. Terekhov, A. Roy, and T. Tannenbaum. Management of grid jobs and data within samgrid. In *IEEE International Conference on Cluster Computing*, 2004.

[25] B. Barán and M. Schaerer. A multiobjective ant colony system for vehicle routing problem with time windows. In *Proc. Twenty first IASTED International Conference on Applied Informatics, Austria*, pages 97–102, 2003.

[26] C. Barnhart, C. A. Hane, and P. H. Vance. Using branch-and-price-and-cut to solve origin-destination integer multicommodity flow problems. *Operations Research*, 48(2):318–326, 2000.

[27] C. Barnhart, E. L. Johnson, G. L. Nemhauser, M. W. P. Savelsbergh, and P. H. Vance. Branch-and-price: Column generation for solving huge integer programs. *Operations Research*, 46:316–329, 1998.

[28] R. Barry and S. Subramaniam. The MAX SUM wavelength assignment algorithm for WDM ring networks. In *Conference on Optical Fiber Communication. OFC 97*, pages 121–122, 1997.

[29] R. A. Barry and P. A. Humblet. Models of blocking probability in all-optical networks with and without wavelength changers. *IEEE Journal on Selected Areas in Communications*, 14:858–867, 1996.

[30] R. J. Bates. *Optical Switching and Networking Handbook*. McGraw-Hill Professional Publishing, 2001.

[31] B. Beauquier, J.-C. Bermond, L. Gargano, P. Hell, S. Pérennes, and U. Vaccaro. Graph problems arising from wavelength-routing in all-optical networks. In *Proc. of 2nd Workshop on Optics and Computer Science, IPPS'97*, 1997.

[32] A. L. Beberg, D. L. Ensign, G. Jayachandran, S. Khaliq, and V. S. Pande. Folding@home: Lessons from eight years of volunteer distributed computing. In *International Parallel and Distributed Processing Symposium*, pages 1–8. IEEE Computer Society, 2009.

[33] G. Behrmann, L. Fischer, M. Gamst, M. Grønager, and J. Kleist. Analysis of internal network requirements for the distributed nordic tier-1. In *17th International Conference on Computing in High Energy and Nuclear Physics (CHEP'09)*, volume 219 of *Journal of Physics: Conference Series (JPCS)*, 2010.

[34] G. Behrmann, P. Fuhrmann, M. Grønager, and J. Kleist. A distributed storage system with dcache. In *International Conference on Computing in High Energy and Nuclear Physics (CHEP'07)*, Journal of Physics: Conference Series (JPCS) (2008), pages 1–11, 2007.

[35] R. Bellman. *Dynamic Programming*. Princeton University Press, 1957.

[36] J. F. Benders. Partitioning procedures for solving mixed-variables programming problems. *Numerische Mathematik*, 4:238–252, 1962.

[37] B. Bergeron. *Bioinformatics Computing*. Prentice Hall, 2001.

[38] A. Birman and A. Kershenbaum. Routing and wavelength assignment methods in single-hop all-optical networks with blocking. In *IEEE Infocom'95*, 1995.

[39] H. K. F. Bjerke. Grid survey. Technical report, CERN openlab, 2004.

[40] D. G. Cameron, A. P. Millar, C. Nicholson, R. Carvajal-Schiaffino, K. Stockinger, and F. Zini. Analysis of scheduling and replica optimisation strategies for data grids using optorsim. *Journal of Grid Computing*, 2:57–69, 2004.

[41] CERN. Worldwide LHC computing grid, http://lcg.web.cern.ch/lcg/.

[42] A. Chakrabarti, R. Dheepak, and S. Sengupta. Integration of scheduling and replication in data grids. In *High Performance Computing - HiPC*, 2004.

[43] K.-M. Chan and T. Yum. Analysis of least congested path routing in WDM lightwave networks. In *INFOCOM '94. Networking for Global Communications., 13th Proceedings IEEE*, 1994.

[44] M. W. Chang, W. Lindstrom, A. J. Olson, and R. K. Belew. Analysis of hiv wild-type and mutant structures via in silico docking against diverse ligand libraries. *J. Chem. Inf. Model.*, 47(3):1258–1262, 2007.

[45] R.-S. Chang, J.-S. Chang, and S.-Y. Lin. Job scheduling and data replication on data grids. *Future Generation Computer Systems*, 23:846–860, 2007.

[46] M. Chiarandini. Bibliography on graph-vertex coloring. http://www.imada.sdu.dk/∼marco/gcp/, 2008.

[47] I. Chlamtac, A. Ganz, and G. Karmi. Lightpath communications: an approach to high bandwidth opticalwan's. *IEEE Transactions on Communications*, 40(7):1171–1182, 1992.

[48] J. S. Choi, N. Golmie, F. Lapeyrere, F. Mouveaux, and D. Su. Classification of routing and wavelength assignment schemes in DWDM networks. In *Proceedings of the 7th International Conference on Optical Communications and Networks, OPNET*, 2000.

[49] W. Chrabakh and R. Wolski. GridSAT: Design and implementation of a computational grid application. *Journal of Grid Computing*, 4(2):177–193, June 2006.

[50] V. Chvatal. *Linear Programming*. W. H. Freeman and Company, 1983.

[51] W. J. Cook, W. H. Cunningham, W. R. Pulleyblank, and A. Schrijver. *Combinatorial optimization*. John Wiley & Sons, Inc, 1998.

[52] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to algorithms*. The MIT Press and McGraw-Hill, second edition, 2001.

[53] N. N. Dang and S. B. Lim. Combination of replication and scheduling in data grids. *International Journal of Computer Science and Network Security*, 7(3):304–308, 2007.

[54] G. B. Dantzig and P. Wolfe. Decomposition principle for linear programs. *Operations Research*, 8:101–111, 1960.

[55] M. P. de Aragão and E. Uchoa. Integer program reformulation for robust branch-and-cut-and-price algorithms. Technical report, Departamento de Informática, PUC-Rio and Dep. de Engenharia de Producão, Universidade Federal Fluminense, 2003.

[56] G. Desaulniers, J. Desrosiers, and M. M. Solomon, editors. *Column Generation*. Springer Science+Media Business, Inc., 2005.

[57] G. Desaulniers, J. Desrosiers, and S. Spoorendonk. Cutting planes for branch-and-price algorithms. Technical Report G-2009-52, GERAD Montreal, 2009.

[58] K. Doerner, W. J. Gutjahr, R. F. Hartl, C. Strauss, and C. Stummer. Pareto ant colony optimization: A metaheuristic approach to multiobjective portfolio selection. *Annals of Operations Research*, 131:79–99, 2004.

[59] K. Doerner, R. F. Hartl, and M. Reimann. Are competants more competent for problem solving? - the case of a multiple objective transportation problem. *Central European Journal of Operations Research,*, 11(2):115–141, 2003.

[60] O. du Merle, D. Villeneuve, J. Desrosiers, and P. Hansen. Stabilisation dans le cadre de la génération de colonnes. Technical report, GERAD, 1997. Publication G-97-08.

[61] O. du Merle, D. Villeneuve, J. Desrosiers, and P. Hansen. Stabilized column generation. *Discrete Math*, 194:229 – 237, 1997.

[62] R. Dutta and G. N. Rouskas. A survey of virtual topology design algorithms for wavelength routed optical networks. *Optical Networks*, 1:73–89, 2000.

[63] P. Eerola, B. Kónya, O. Smirnova, T. Ekelöf, M. Ellert, J. R. Hansen, J. L. Nielsen, A. Wäänänen, A. Konstantinov, and F. Ould-Saada. Building a production grid in scandinavia. *IEEE Internet Computing*, 7(4):27–35, 2003.

[64] A. Elghirani, R. Subrata, and A. Y. Zomaya. Intelligent scheduling and replication in datagrids: a synergistic approach. In *Seventh IEEE International Symposium on Cluster Computing and the Grid (CCGrid'07)*, 2007.

[65] M. Ellert, M. Grønager, A. Konstantinov, B. Kónya, J. Lindemann, I. Livenson, J. Nielsen, M. Niinimäki, O. Smirnova, and A. Wäñänen. Advanced resource connector middleware for lightweight computational grids. *Future Generation Computer Systems*, 23(2):219–240, 2007.

[66] D. Eppstein. Finding the k shortest paths. *SIAM Journal on Computing*, 28(2):652–673, 1998.

[67] J. W. Evans and C. Filsfils. *Deploying IP and MPLS QoS for Multiservice Networks: Theory and Practice*. Morgan Kaufmann, 2007.

[68] T. Ferrandiz and V. Marangozova. *Managing Scheduling and Replication in the LHC Grid*, chapter 1, pages 66–77. Grid Middleware and Services: Challenges and Solutions. Springer US, 2008.

[69] L. Field, M. Grønager, D. Johansson, and J. Kleist. Towards sustainability: An interoperability outline for a regional ARC based infrastructure in the WLCG and EGEE infrastructures. In *Proceedings of CHEP'09*, 2009.

[70] L. Fischer, M. Grønager, J. Kleist, and O. Smirnova. A distributed tier-1. In *Conference on Computing in High Energy and Nuclear Physics (CHEP'07)*, page 1, 2007.

[71] L. Fischer, M. Grønager, J. Kleist, and O. Smirnova. A distributed tier-1. In *International Conference on Computing in High Energy and Nuclear Physics (CHEP'07)*, Journal of Physics: Conference Series (JPCS) (2008), pages 1–12, 2007.

[72] L. R. Ford and D. R. Fulkerson. Constructing maximal dynamic flows from static flows. *Operations Research*, 6(3):419–433, 1958.

[73] L. R. Ford and D. R. Fulkerson. A suggested computation for maximal multi-commodity network flows. *Management Science*, 5:97–101, 1958.

[74] L. R. Ford and D. R. Fulkerson. *Flows in Networks*. Princeton University Press, Princeton, New Jersey, 1962.

[75] I. Foster and C. Kesselman. The globus project: A status report. In *Proc. IPPS/SPDP '98 Heterogeneous Computing Workshop*, pages 4–18, 1998.

[76] I. Foster and C. Kesselman, editors. *The Grid 2: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann, 2 edition, 2003.

[77] M. Gamst. Tidsmæssig planlægning af eksekvering af jobs i grid computing. In *Forskningsnet konferencen*, 2007.

[78] M. Gamst. A survey of the routing and wavelength assignment problem. Technical Report 10.2009, DTU Management Engineering, 2009.

[79] M. Gamst. Greedy and meta-heuristics for the offline scheduling problem in grid computing. Technical Report 2.2010, DTU Management Engineering, 2010.

[80] M. Gamst. Integrating job scheduling and constrained network routing. In *International Symposium on Combinatorial Optimization (ISCO)*, Electronic Notes on Discrete Mathematics (ENDM), 2010.

[81] M. Gamst. Solving the integrated job scheduling and constrained network flow problem. In submission, 2010.

[82] M. Gamst, P. N. Jensen, D. Pisinger, and C. Plum. Two- and three-index formulations of the minimum cost multicommodity k-splittable flow problem. In *International Network Optimization Conference (INOC), Pisa, Italy*, 2009.

[83] M. Gamst, P. N. Jensen, D. Pisinger, and C. Plum. Two- and three-index formulations of the minimum cost multicommodity k-splittable flow problem. *European Journal of Operations Research (EJOR)*, 202(1):82–89, 2010.

[84] M. Gamst and B. Petersen. Comparing branch-and-price algorithms for the multicommodity k-splittable maximum flow problem. In submission, 2009.

[85] M. Gamst and B. Petersen. An exact solution approach for the maximum multicommodity k-splittable flow problem. In *the 20th International Symposium on Mathematical Programming 2009 (ISMP'09)*, 2009.

[86] M. Gamst and D. Pisinger. Integrated job scheduling and network routing. In submission, September 2009.

[87] M. Gamst, C. Plum, and P. N. Jensen. Multicommodity k-splittable flow problemet løst med branch & price. Report 06-02, Department of Computer Science, Copenhagen University, 2006.

[88] P. Gardel, B. Barán, U. Fernández, H. Estigarribia, and S. Duarte. Multiobjective reactive power compensation with an ant colony optimisation algorithm. In *Proc. IEE 8TH International Conference AC and DC Power Transmission, ACDC 2006, London, England*, 2006.

[89] M. R. Garey and D. S. Johnson. *Computers and Intractability. A Guide to the Theory of NP-Completeness.* W. H. Freeman and Company, New York, 1979.

[90] M. R. Garey, D. S. Johnson, G. L. Miller, and C. H. Papadimitriou. The complexity of coloring circular arcs and chords. *SIAM J. Alg. Disc. Meth*, 1(2):216–227, 1980.

[91] O. Gerstel, G. Sasaki, S. Kutten, and R. Ramaswami. Worst-case analysis of dynamic wavelength allocation in optical networks. *IEEE/ACM Transactions on Networking*, 7(6):833–846, 1999.

[92] F. W. Glover and M. Laguna. *Tabu Search*. Springer, 1997.

[93] D. E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley Professional, 1989.

[94] R. E. Gomory. Outline for an algorithm for integer solutions to linear programs. *Bulletin of the American Mathematical Society*, 64:275–278, 1958.

[95] R. E. Gomory. *Recent Advances in Mathematical Programming*, chapter An algorithm for integer solutions to linear programs. McGraw-Hill, New York, 1963.

[96] M. Grønager. A nordic tier-1 for LHC. Conference Presentation, GLORIAD, 2007.

[97] M. Grötschel, L. Lovasz, and A. Schrijver. The ellipsoid method and its consequences in combinatorial optimization. *Combinatorica*, 1:169–197, 1981.

[98] F. Halsall. *Multimedia Connections. Applications, Networks, Protocols and Standards*. Addison-Wesley, 2001.

[99] G. Y. Handler and I. Zang. A dual algorithm for the constrained shortest path problem. *Networks*, 10(4):293 – 309, 1980.

[100] C. Huang, D. Chen, Y. Zheng, and H. Hu. *Performance-Driven Task and Data Co-scheduling Algorithms for Data-Intensive Applications in Grid Computing*, chapter 36, pages 331–340. Lecture Notes in Computer Science: Advanced Web Technologies and Applications. Springer Berlin / Heidelberg, 2004.

[101] E. Hyytiä and J. Virtamo. Wavelength assignment and routing in WDM network. In *Nordic Teletraffic Seminar (NTS)*, pages 18–20, 1998.

[102] ILOG CPLEX. http://www.ilog.com/products/cplex/, 2010.

[103] J. Iness. *Efficient Use of Optical Components in WDM-Based Optical Networks*. PhD thesis, University of California, Davis, 1997.

[104] J. Iness and B. Mukherjee. Sparse wavelength conversion in wavelength-routed WDM optical networks. *Photonic Network Communications*, 1(3):183–205, 1999.

[105] S. Iredi, D. Merkle, and M. Middendorf. Bi-criterion optimization with multi colony ant algorithms. In *Proceedings of the First International Conference on Evolutionary Multi-Criterion Optimization (EMO 2001), LNCS*, pages 359–372, 1993.

[106] B. Jaumard, C. Meyer, and B. Thiongane. Comparison of ILP formulations for the RWA problem. *Optical Switching and Networking*, 4:157–172, 2007.

[107] B. Jaumard, C. Meyer, and B. Thiongane. On column generation formulations for the RWA problem. *Discrete Applied Mathematics*, 157:1291–1308, 2009.

[108] B. Jaumard, C. Meyer, and X. Yu. How much wavelength conversion allows the reduction of the blocking rate? *Journal of Optical Networking*, 5(12):881–900, 2006.

[109] T. Jensen and B. Toft. *Graph Coloring Problems*. Wiley-Interscience, 1994.

[110] G. Jeong and E. Ayanoglu. Comparison of wavelength-interchanging and wavelength-selectivecross-connects in multiwavelength all-optical networks. In *INFOCOM '96. Fifteenth Annual Joint Conference of the IEEE Computer Societies. Networking the Next Generation.*, volume 1, pages 156–163, 1996.

[111] M. K. Jepsen, B. Petersen, S. Spoorendonk, and D. Pisinger. Subset-row inequalities applied to the vehicle-routing problem with time windows. *Operations Research*, 56(2):497–511, 2008.

[112] J. Jiang and G. Yang. An optimal replication strategy for data grid systems. *Frontiers of Computer Science in China*, 1(3):338–348, 2007.

[113] P. Judea. *Heuristics: Intelligent Search Strategies for Computer Problem Solving*. The Addison-Wesley series in artificial intelligence. Addison-Wesley Publishers, 1984.

[114] J. P. Jue. *Design and Analysis of Architecture and Protocols for WDM Optical Networks*. PhD thesis, University of California, Davis, 1999.

[115] B. Kallehauge, J. Larsen, and O. B. G. Madsen. Lagrangian duality applied to the vehicle routing problem with time windows. *Computers and Operations Research*, 33(5):1464–1487, 2006.

[116] A. Kamath and O. Palmon. Improved interior point algorithms for exact and approximate solution of multicommodity flow problems. In *Proceedings of the Sixth Annual ACM-SIAM Symposium on Discrete Algorithms*, 1995.

[117] E. Karasan and E. Ayanoglu. Effects of wavelength routing and selection algorithms on wavelength conversion gain in WDM optical networks. *IEEE/ACM Transactions on Networking*, 6(2):186 – 196, 1998.

[118] H. Kellerer, U. Pferschy, and D. Pisinger. *Knapsack Problems*. Springer, 2004.

[119] J. Kennington. A survey of linear cost multicommodity network flows. *Operations Research*, 26:209–236, 1978.

[120] S. Kim, K. N. Chang, and J. Y. Lee. A descent method with linear programming subproblems for nondifferentiable convex optimization. *Mathematical Programming*, 71:17–28, 1995.

[121] J. M. Kleinberg. *Approximation algorithms for disjoint paths problems*. PhD thesis, MIT, Cambridge, MA, 1996.

[122] R. Koch, M. Skutella, and I. Spenke. Approximation and complexity of k-splittable flows. In *Approximation and Online Algorithms, Third International Workshop, WAOA*, pages 244–257, 2005.

[123] R. Koch, M. Skutella, and I. Spenke. Maximum k-splittable s, t -flows. *Theory of Computing Systems*, 43(1):1432–4350, 2008.

[124] P. Kokkinos, K. Christodoulopoulos, A. Kretsis, and E. Varvarigos. Data consolidation: A task scheduling and data migration technique for grid networks. In *Eighth IEEE International Symposium on Cluster Computing and the Grid*, 2008.

[125] A. Koster and M. Scheffel. A routing and network dimensioning strategy to reduce wavelength continuity conflicts in all-optical networks. In *Proceedings of the International Network Optimization Conference (INOC)*, 2007.

[126] A. M. Koster. Wavelength assignment in multi-fiber WDM networks by generalized edge coloring. Technical Report ZIB-Report 05-13, Konrad-Zuse-Zentrum für Informationstechnik Berlin, 2005.

[127] A. M. Koster and A. Zymolka. Linear programming lower bounds for minimum converter wavelength assignment in optical networks. Technical Report ZIB Report 04-41, Konrad-Zuse-Zentrum für Informationstechnik Berlin, 2004.

[128] A. M. Koster and A. Zymolka. Provably good solutions for wavelength assignment in optical networks. Technical Report ZIB-Report 04-40, Konrad-Zuse-Zentrum für Informationstechnik Berlin, 2004.

[129] A. M. Koster and A. Zymolka. Tight LP-based lower bounds for wavelength conversion in optical networks. *Statistica Neerlandica*, 61(1):115–136, 2007.

[130] M. Kovacevic and A. Acampora. Benefits of wavelength translation in all-optical clear-channel networks. *IEEE Journal on Selected Areas in Communications*, 14:868–880, 1996.

[131] R. M. Krishnaswamy and K. N. Sivarajan. Design of logical topologies: a linear formulation for wavelength-routed optical networks with no wavelength changers. *IEEE/ACM Transactions on Networking*, 9(2):186 – 198, 2001.

[132] J. F. Kurose and K. W. Ross. *Computer Networking. A Top-Down Approach Featuring the Internet*. Addison-Wesley, fifth edition, 2009.

[133] A. H. Land and A. G. Doig. An automatic method of solving discrete programming problems. *Econometrica*, 28(3):497–520, 1960.

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

[135] M. E. Lübbecke and J. Desrosiers. Selected topics in column generation. Les Cahiers du GERAD G-2002-64, HEC Montréal and GERAD, Canada, 2002.

[136] M. E. Lübbecke and J. Desrosiers. Selected topics in column generation. *Operations Research*, 53(6):1007–1023, 2005.

[137] K. Lee and V. O. K. Li. A wavelength-convertible optical network. *Journal of lightwave technology*, 11:962–970, 1993.

[138] T. Lee, K. Lee, and S. Park. Optimal routing and wavelength assignment in WDM ring networks. *IEEE Journal on selected areas in communications*, 18(10):2146 – 2154, 2000.

[139] L. Li and A. K. Somani. Dynamic wavelength routing using congestion and neighborhood information. *IEEE/ACM Transactions on Networking*, 7:779–786, 1999.

[140] R. Lougee-Heimer. The common optimization interface for operations research. *IBM Journal of Research and Development*, 47:57–66, 2003.

[141] J. Lysgaard, A. N. Letchford, and R. W. Eglese. A new branch-and-cut algorithm for the capacitated vehicle routing problem. *Mathematical Programming*, 100(A)(2):423–445, 2004.

[142] L. Marchal, Y. Robert, P. V.-B. Primet, and J. Zeng. Scheduling network requests with transmission window. Technical Report 2005-32, Laboratoire de L'Informatique du Parallélisme, École Normale Supérieure de Lyon, France, 2005.

[143] E. Marcotte and S. Date. Exploiting big biology: integrating large-scale biological data for function inference. *Brief Bioinform.*, 2(4):363–374, 2001.

[144] C. E. Mariano, P. Cuauhnahuac, and E. Morales. A multiple objective Ant-Q algorithm for the design of water distribution irrigation networks. Technical Report HC-9904, Instituto Mexicano de Tecnología del Agua, Mexico, 1999.

[145] R. E. Marsten, W. W. Hogan, and J. W. Blankenship. The BOXSTEP method for large-scale optimization. *Operations Research*, 23:389–405, 1975.

[146] R. K. Martin. *Large scale linear and integer optimization*. Kluwer Academic Publishers, 2004.

[147] R. McClatchey, A. Anjum, H. Stockinger, A. Ali, I. Willers, and M. Thomas. Data intensive and network aware (DIANA) grid scheduling. *Journal of Grid Computing*, 5:43–64, 2007.

[148] L. Meyer, D. Scheftner, J. Vöckler, M. Mattoso, M. Wilde, and I. Foster. An opportunistic algorithm for scheduling workflows on grids. In *High Performance Computing for Computational Science - VECPAR 2006*, Lecture Notes in Computer Science, 2006.

[149] Z. Michalewicz and D. B. Fogel. *How to Solve It: Modern Heuristics*. Springer, 2004.

[150] A. Mokhtar and M. Azizoglu. Adaptive wavelength routing in all-optical networks. *IEEE/ACM Transactions on Networking*, 6:197–206, 1998.

[151] P. Neame. *Nonsmooth dual methods in integer programming*. PhD thesis, University of Melbourne, 1999.

[152] G. L. Nemhauser and L. A. Wolsey. *Integer and combinatorial optimization*. John Wiley & Sons, Inc., 1999.

[153] R. Neves, N. Mestre, F. Machado, and J. Lopes. Parallel and distributed computing: Boinc grid implementation. http://www.slideshare.net/neiualg/grid-computing-boinc-overview-1609533, 2010. White paper.

[154] Nordic DataGrid Facility. http://www.ndgf.org/, 2009.

[155] T. F. Noronha and C. C. Ribeiro. Routing and wavelength assignment by partition colouring. *European Journal of Operational Research*, 171(3):797 – 810, 2006.

[156] A. E. Ozdaglar and D. P. Bertsekas. Routing and wavelength assignment in optical networks. *IEEE/ACM Transactions on Networking*, 11(2):259– 272, 2003.

[157] J. Paciello, H. Martínez, C. Lezcano, and B. Barán. Algoritmos de optimización multi-objetivos basados en colonias de hormigas. In *Proceedings of CLEI'2006. Latin-American Conference on Informatics (CLEI). Santiago, Chile*, 2006.

[158] P. M. Pardalos, T. Mavridou, and J. Xue. *Handbook of Combinatorial Optimization*, volume 1-3, chapter The Graph Coloring Problem: A Bibliographic Survey. Kluwer Academic Publishers, 1998.

[159] D. Pinto and B. Barán. Solving multiobjective multicast routing problem with a new ant colony optimization approach. In *Proceedings of the 3rd international IFIP/ACM Latin American conference on Networking*, pages 11–19, 2005.

[160] D. Pisinger. *Algorithms for Knapsack Problems*. PhD thesis, Dept. of Computer Science, University of Copenhagen, Febraury 1995.

[161] D. Pisinger and M. M. Sigurd. The two-dimensional bin packing problem with variable bin sizes and costs. *Discrete Optimization*, 2:154–167, 2005.

[162] P. Raghavan and E. Upfal. Efficient routing in all-optical networks. In *Proceedings of the twenty-sixth annual ACM symposium on Theory of computing*, pages 134 – 143, 1994.

[163] B. Ramamurthy and B. Mukherjee. Wavelength conversion in WDM networking. *IEEE Journal on selected areas in Communications*, 16:1061–1073, 1998.

[164] R. Ramaswami and K. N. Sivarajan. Routing and wavelength assignment in all-optical networks. *IEEE/ACM Transactions on Networking*, 3:489–500, 1995.

[165] K. Ranganathan and I. Foster. Simulation studies of computation and data scheduling algorithms for data grids. *Journal of Grid Computing*, 1(1):53–62, March 2003.

[166] M. G. C. Resende and P. M. Pardalos, editors. *Handbook of optimization in telecommunications*. Springer Science+Business Media, Inc., 2006.

[167] K. Ricker. Archiving the universe, 2003. http://access.ncsa.illinois.edu/Stories/NVOatNCSA/index.html.

[168] J. T. Robacker. Concerning multicommodity networks. Technical Report RM-1799, The RAND Corporation, Santa Monica, Calfornia, 1956.

[169] F. Rothlauf. *Application and Adaptation of Heuristic Optimization Methods*. Springer, 2010.

[170] L.-M. Rousseau, M. Gendreau, and D. Feillet. Interior point stabilization for column generation. Technical Report Publication CRT-2003-39, Centre de recherche sur les transports, Université de Montréal, 2003.

[171] D. M. Ryan and B. A. Foster. An integer programming approach to scheduling. *Computer Scheduling of Public Transport: Urban passenger and crew scheduling*, 1:269–280, 1981. North-Holland Publishing Company.

[172] F. Schintke and A. Reinefeld. Modeling replica availability in large data grids. *Journal of Grid Computing*, 1:219–227, 2003.

[173] A. Schrijver. *Theory of Linear and Integer Programming*. John Wiley & Sons, Chichester, 1986.

[174] A. Schrijver. *Combinatorial Optimization*. Springer, 2003.

[175] SETI@home. http://setiathome.ssl.berkeley.edu/, 2009.

[176] J. Shiers. The worldwide LHC computing grid (worldwide LCG). In *Conference on Computational Physics CCP*, volume 117, pages 219–223, 2006.

[177] M. M. Sigurd. *Column Generation Methods and Applications*. PhD thesis, Dept. of Computer Science, University of Copenhagen, Denmark, 2004.

[178] M. C. Sinclair. Minimum cost routing and wavelength allocation using a genetic-algorithm/heuristic hybrid approach. In *Proc. 6th IEE Conf. on Telecommunications*, pages 66–71, 1998.

[179] N. Skorin-Kapov. Routing and wavelength assignment in optical networks using bin packing based algorithms. *European journal of operational research*, 177(2):1167–1179, 2007.

[180] M. M. Solomon. Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations Research*, 35(2):254 – 265, 1987.

[181] R. B. Sørensen. Analysing resource allocation in minimum intrusion grid (MiG) using mechanism design. Master's thesis, Department of Computer Science, Copenhagen University (DIKU), 2007.

[182] Standard Performance Evaluation Corporation. http://www.spec.org/, 2010.

[183] S. Subrarnaniam and R. A. Barry. Wavelength assignment in fixed routing WDM networks. In *IEEE Int. Conf. Communications*, pages 406–410, 1997.

[184] M. Tang, B.-S. Lee, X. Tang, and C.-K. Yeo. Combining data replication algorithms and job scheduling heuristics in the data grid. In *11th International Euro-Par Conference, Lisbon, Portugal*, Lecture Notes in Computer Science, 2005.

[185] M. Tang, B.-S. Lee, C.-K. Yeo, and X. Tang. Dynamic replication algorithms for the multi-tier data grid. *Future Generation Computer Systems*, 21:775–790, 2005.

[186] D. Thain, J. Bent, A. Arpaci-Dusseau, R. Arpaci-Dusseau, and M. Livny. Gathering at the well: Creating communities for grid I/O. In *ACM/IEEE SC2001 Conference (SC'01)*, 2001.

[187] H. J. Thiele and M. Nebeling. *Coarse Wavelength Division Multiplexing: Technologies and Applications (Optical Science and Engineering Series)*. CRC, 2007.

[188] P. Thysebaert, B. Volckaert, F. D. Turck, B. Dhoedt, and P. Demeester. Network aspects of grid scheduling algorithms. In *17th international conference on Parallel and Distributed Computing (ISCA)*, 2004.

[189] Transit grid generator, 2010. http://www.informatik.uni-trier.de/˜naeher/Professur/research/generators/maxflow/tg/index.html.

[190] J. Truffot and C. Duhamel. A branch and price algorithm for the k-splittable maximum flow problem. *Discrete Optimization*, 5(3):629–646, 2008.

[191] J. Truffot, C. Duhamel, and P. Mahey. Branch & price pour le problème du multiflot k-séparable de coût minimal. In *LIMOS, UMR 6158 - CNRS, ROADEF'05*, February 2005.

[192] J. Truffot, C. Duhamel, and P. Mahey. Using branch-and-price to solve multicommodity k-splittable flow problems. In *Proceedings of International Network Optimization Conference (INOC)*, 2005.

[193] P. van Laarhoven and E. Aarts. *Simulated Annealing: Theory and Applications*. Springer, 1987.

[194] F. Vanderbeck and M. Savelsbergh. A generic view of dantzig-wolfe decomposition in mixed integer programming. *Operations Research Letters*, 34(3):296–306, 2006.

[195] G. N. Varela and M. C. Sinclair. Ant colony optimisation for virtual-wavelength-path routing and wavelength allocation. In *Proceedings of the Congress on Evolutionary Computation (CEC'99)*, pages 1809–1816, 1999.

[196] E. Varvarigos, V. Sourlas, and K. Christodoulopoulos. Routing and scheduling connections in networks that support advance reservations. *Computer Networks*, 52:2988–3006, 2008.

[197] V. V. Vazirani. *Approximation Algorithms*. Springer, 2001.

[198] S. Venugopal. *Scheduling Distributed Data-Intensive Applications on Global Grids*. PhD thesis, Department of Computer Science and Software Engineering, The University of Melbourne, 2006.

[199] D. Villeneuve and G. Desaulniers. The shortest path problem with forbidden paths. *European Journal of Operational Research*, 165(1):97–107, 2005.

[200] D. Villeneuve, J. Desrosiers, M. E. Lübbecke, and F. Soumis. On compact formulations for integer programs solved by column generation. *Annals of Operations Research*, 139(1):375–388, 2005.

[201] B. Vinter. The architecture of the minimum intrusion grid, MiG. In *Communicating Process Architecture*, pages 189–201, 2005.

[202] B. Volckaert, P. Thysebaert, M. D. Leenheer, F. D. Turck, B. Dhoedt, and P. Demeester. Network aware scheduling in grids. In *9th European Conference on Networks & Optical Communications (NOC 2004)*, 2004.

[203] C. Weng, M. Li, and X. Lu. An online scheduling algorithm for assigning jobs in the computational grid. *IEICE Trans. inf. & Syst.*, 89(2):597–604, 2006.

[204] L. A. Wolsey. *Integer Programming*. Wiley-interscience publication, 1998.

[205] J. Yates, J. Lacey, D. Everitt, and M. Summerfield. Limited-range wavelength translation in all-optical networks. In *IEEE INFOCOM '96*, volume 3, 1996.

[206] H. Zang, J. P. Jue, and B. Mukherjee. A review of routing and wavelength assignment approaches for wavelength-routed optical wdm networks. *Optical Networks Magazine*, 1:47–60, January 2000.

[207] X. Zhang and C. Qiao. Wavelength assignment for dynamic traffic in multi-fiber WDM networks. In *ICCCN'98*, pages 479 – 585, 1998.

[208] J. Zheng and H. T. Mouftah. *Optical WDM Networks: Concepts and Design Principles*. Wiley-IEEE, 2004.

[209] A. Zymolka. *Design of Survivable Optical Networks by Mathematical Optimization*. PhD thesis, Mathematik und Naturwissenschaften, Technischen Universität Berlin, 2007.

This thesis concerns scheduling of network traffic in grid context. Grid computing consists of a number of geographically distributed computers, which work together for solving large problems. The computers are connected through a network. When scheduling job execution in grid computing, data transmission has so far not been taken into account. This causes stability problems, because data transmission takes time and thus causes delays to the execution plan.

This thesis proposes the integration of job scheduling and network routing. The scientific contribution is based methods from operations research and consists of six papers. The first four considers data transmission in grid context. The last two solves the data transmission problem, where the number of paths per data connection is bounded from above.

The thesis shows that it is possible to solve the integrated job scheduling and network routing problem to optimality for a grid, where computers are connected through a packet-switched network. When the network topology is optical, the routing problem becomes significantly more complex and the problem should thus be solved heuristically. Furthermore, the thesis proposes a number of new exact methods for the data transmission problem, where the number of paths is bounded from above. The new exact solution methods outperform existing methods from the literature.

ISBN 978-87-90855-83-3

DTU Management Engineering
Department of Management Engineering
Technical University of Denmark

Produktionstorvet
Building 424
DK-2800 Kongens Lyngby
Denmark
Tel. +45 45 25 48 00
Fax +45 45 93 34 35

www.man.dtu.dk