FACULTY OF SCIENCE
UNIVERSITY OF COPENHAGEN

PhD thesis

Simon Spoorendonk

Cut and Column Generation

Academic advisor: David Pisinger

Submitted: 31/10/08

Preface

This Ph.D. thesis was written at the Department of Computer Science, University of Copenhagen (DIKU) from November 2005 to October 2008 under the supervision of Professor David Pisinger.

I would like to thank Professors Guy Desaulniers and Jacques Desrosiers whom I visited for half year in Montreal, and had the opportunity to continue working with after I returned to Denmark. Also, I would like to thank my colleagues at DIKU for a good place to work, especially Mads Jepsen, Bjørn Petersen and my supervisor David Pisinger for the many good discussions, and Mette Gamst and Laurent Flindt Muller for proof reading some chapters in this thesis. Thanks.

Simon Spoorendonk
October, 2008

Contents

Part I

Introduction

Chapter 1

Introduction

Simon Spoorendonk
DIKU Department of Computer Science, University of Copenhagen

1 Motivation

In the industry many resources are spent on handling the challenges in complex planning problems. The increasing need for more detailed and precise plans to larger planning problems makes decision support tools indispensable. Operations research strives to support the decision making by providing a number of tools such as mathematical modeling and mathematical programming. Mathematical modeling is used to formulate problems in a concrete way using mathematical equations, whereas mathematical programming covers solution methods for the mathematical formulations. These approaches have been known for more than sixty years and their popularity have been consolidated through the several commercial software packages, that have become available in the last twenty years. These packages provide general purpose algorithms based on mathematical formulations and have been very successful; partly due too increased computer power, but primarily due to algorithmic improvements. Practical applications of mathematical programming can be found within the railway, airline, and the maritime industries, see e.g. Huisman [5], Bélanger et al. [1], Christiansen et al. [2]. Textbooks on mathematical programming and solution methods in the areas of transportation and production planning can be found in, e.g., Toth and Vigo [16], Golden et al. [4], Pinedo [12], Pochet and Wolsey [13].

1.1 Mathematical Modeling and Programming

Generally speaking mathematical modeling can be described as a set of variables, that are used to represent decisions in a problem and a set of equations (or inequalities) denoted *constraints*, that are used to limit the amount of valid decisions. For example, in production planning the decisions are the amount of products produced on a machine and a constraint could be an overall maximum capacity of products, that can be produced on the machine. The constraints define the feasible solution space of a problem, i.e., a polytope in a multi-dimensional space that contains all valid solutions to the problem (assuming that the problem is bounded). The objective function is a function of the variables, that guides the process to a solution in the feasible solution space, where the objective function reaches the global minimum (or maximum). In production planning it could be a function minimizing the overall production cost. When variables are continuous, and the constraints and the objective function are linearized, we have a linear program (LP). When integrality is imposed on the variables, it is denoted an integer program (IP) or a mixed integer program (MIP) if both types of variables exists. The two latter problem types are the type of problems considered in this thesis.

Many problems can be formulated as MIPs and solution methods have received a lot of attention during the years. The solution methods can roughly be split into three approaches:

- *Exact algorithms* are exact in the sense that the solution found is proven to be optimal, i.e., there does not exist any other solution with a better objective function value. Exact algorithms for MIPs are often based on enumeration schemes and may be impractical in practice due to long running times.
- *Heuristics* are often used if it is not imperative that an optimal solution is found and if running time is a factor. In this case, good solutions a sought but with no guarantee w.r.t. solution quality.
- *Approximation algorithms* are basically heuristics that have a solution quality guarantee.

Although, exact algorithms appear less attractive from a practical point of view due to long computation times, the study of exact methods often give insight into the problem behavior that may be hard to grasp. Furthermore, the improvement of exact methods are pushing the boundaries for their usefulness, hence many heuristic nowadays are based on exact approaches or incorporate them to solve subproblems. The solution method investigated in this thesis is an exact solution algorithm.

1.2 Exact Methods

Many exact methods are based on the branch-and-bound paradigm. A branch-and-bound algorithm searches a tree, where each branch represents a split of the solution space. Starting in the root node, a problem can be split in two by dividing the feasible region of an integer variable in the middle (note rounding can be used to tighten the new regions as the variable must be integer in a feasible solution). Relaxations of the problem are used in each node of the branch tree, to calculate lower bounds (in case of a minimization objective function). Each time an improved feasible solution (an upper bound) is found in the branch tree, it is possible to fathom part of the branch tree if the gap between the lower and upper bounds are zero. A simple way to calculate a lower bound is to use the LP relaxation of the MIP, i.e., to consider the enlarged solution space where integer variables are allowed to obtain continuous values.

A well studied way to improve the lower bound calculation is the use of cutting planes. Cutting planes (or cuts) are inequalities for the solution space of the problem that cuts off some of the current fractional solution, i.e., the non-integer solution obtained by the LP relaxation. The cuts must be valid inequalities, i.e., they may never be able to cut off any feasible solutions. Combining cutting planes with the branching paradigm is known as a branch-and-cut algorithm. Branch-and-cut algorithms are among the best general purpose solvers for arbitrary MIPs due to the very general framework that are used. Both LP solvers and cut generators for general cutting planes in MIPs have been widely studied, see e.g., Nemhauser and Wolsey [10], Wolsey [18].

Even though branch-and-cut algorithms are well tuned algorithms, there do exist problems where these algorithms fall short. Problem formulations may be very structured, i.e., there are certain variable sets where some constraints are non-overlapping. In this case, it is possible to use the Dantzig-Wolfe decomposition principle to decompose the problem into smaller subproblems that have their solutions combined in a *master* problem, see e.g., Martin [9]. The master problem is often small in the number of constraints but may have a large (worst case exponential) number of variables. Often a decomposition is sought such that the subproblems (also denoted *pricing* problems) are easy to solve. An LP relaxation of the master problem is a different way of calculating the lower bound of the non-decomposed *original* problem and may in many cases be a better bound. The drawback is that the subproblem must be solved iteratively until the master problem objective value cannot be minimized further. This is also known as *column generation*, since subproblems solution gives rise to new columns in the master problem. When considering the decomposition lower bound with branching, it is denoted a branch-and-price algorithm or an integer column generation algorithm, see e.g., Desaulniers et al. [3], Lübbecke and Desrosiers [8]. Note, that current branch-and-price algorithms are not as generic as branch-and-cut algorithms, since a valid decomposition of the problem must be added by the user. Also the subproblem algorithms are often very problem specific and do only apply for few problem types. These are reasons why very few commercial branch-and-price algorithms are available and that non of these are used as general purpose MIP solvers.

It is possible to combine the ideas of cutting planes with the column generation approach. This is denoted a branch-and-cut-and-price algorithm. However, adding cuts is not as straight forward as in branch-and-cut algorithms. The cut scenario can be viewed from two angles:

- Cuts expressed in the original formulation.
- Cuts expressed in the master problem formulation.

The first case have some relation with the second case, since it possible to decompose any cut expressed in the original problem into a cut expressed in the master problem formulation, see Villeneuve et al. [17]. This approach is fairly easy to implement, as it only gives rise to changed objective function coefficients in the subproblem and otherwise leaves the number of variables at status quo. However, it is not always obvious how to separate cuts in the original formulation when only regarding the master problem. For combinatorial derived cuts, it is possible to map the master problem solution back to a solution in the original problem, but for general purpose cuts, e.g., cuts that rely on optimal simplex tableaus etc., the separation may prove very difficult. Indeed, combinatorial cuts based on the original formulation is the most popular approach in branch-and-cut-and-price algorithms.

The second case addresses cutting planes derived directly on the master problem formulation. This may be either combinatorial cuts based on the master problem structure or general purpose cuts. Neither of these cuts have a clear representation in the original problem and it can be difficult to incorporate their impact into the subproblems. Different approaches have been suggested such as expressing the change as a non-linear objective function or adding additional variables in the subproblems. Both approaches changes the complexity aspect of the subproblem and can result in much higher computational effort to solve the subproblems. This case is less studied both theoretically and experimentally and is the main focus of this thesis.

2 Goals

The focus of this thesis is on the application of cutting planes in a column generation context. The main goals can be summarized as:

- Investigate how to apply cutting planes derived from the master problem formulation.
- Use experiments to investigate the impact of the cutting planes on the subproblems complexity, on the quality of the lower bounds for the master problem, and the overall running time of the branch-and-cut-and-price algorithm.
- Describe the theoretical basis for explaining the connection between cuts for the master problem formulation and the relation to the original problem formulation.

When investigating the application of cuts, it is the goal to consider cuts based both on combinatorial aspects and from the family of general purpose cuts. Many problems decompose into a set partitioning master problem for which very efficient combinatorial cuts a known, e.g., the clique inequalities. A general purpose cut family is the Chvátal-Gomory cuts that have been known for years and is implemented in any commercial branch-and-cut algorithm nowadays.

It is the goal, that the additional cuts can be incorporated into the existing subproblem algorithms as efficiently as possible and by changing any special purpose algorithms as little as possible. The increased complexity (and thereby potentially increased running times) of the subproblems is a trade-off with the quality of the lower bound obtained in the master problem. The running time saved by exploring fewer branch nodes due to the improved lower bound is hopefully overshadowing the increased effort put in solving the subproblems.

A description in generic terms of the cutting plane process and the relation between the the original problem formulation and the master problem formulation are hoped to bring an understanding to this field and provide a common basis for future research.

3 Contributions

The main contributions of the thesis is summarized in the points below.

- It is shown, theoretically and experimentally, how to apply the Chvátal-Gomory cuts of rank 1, known from branch-and-cut algorithms for general MIPs, to the vehicle routing problem with time windows. Furthermore, it is shown how to incorporate this into a dynamic programming algorithm for the subproblem. The approach appears very successful and it is possible to solve several previously unsolved instances from the benchmarks of Solomon [14].
- It is shown how to apply the clique inequalities, a combinatorial cut for the set partitioning problem, to the vehicle routing problem with time windows.
- A generic framework is presented showing how cutting planes interact between the original formulation and the master problem formulation and how this is modeled in branch-and-cut-and-price algorithms. Examples are given on how previous work with cutting planes fit into this framework.

A more detailed description of the contributions of each chapter can be found in the reading guide in the following Section 4.

4 Reading Guide

Following is a chapter-wise guide for reading this thesis.

Chapter 2: Preliminaries on Cut and Column Generation. This chapter is as a short introduction into BCP algorithms. Also, it presents the basic concepts and notation used throughout the thesis. The experienced reader may skip this chapter.

4.1 Part II: Cut and Column Generation

This part concerns the main topic of the thesis. That is, how to apply cutting planes in a BCP algorithm based on the master problem formulation.

Chapter 3: Subset-Row Inequalities Applied to the Vehicle Routing Problem with Time Windows. The paper presents how a subset of the Chátal-Gomory cuts may be applied to the master problem of a decomposition of the vehicle routing problem with time windows. It is shown how each cut in the master problem increases the complexity of the subproblem and how this is be handled in a dynamic programming algorithm. Experimental results were carried out on the Solomon instances and it was possible so solve several previously unsolved instances with this new approach. Furthermore, experiments showed that the cuts improved the lower bounds to an extend that significantly reduced the size of the branch tree. The paper is co-authored with Mads Jepsen, Bjørn Petersen, and David Pisinger and has been published in the journal *Operation Research*, see Jepsen et al. [7].

Chapter 4: Chvátal-Gomory Rank 1 Cuts used in a Dantzig-Wolfe Decomposition of the Vehicle Routing Problem with Time Windows. This paper is an extension of the work done in Jepsen et al. [7], where it is shown how any Chátal-Gomory rank 1 cut can be applied to the vehicle routing problem with time windows. Experimental results showed that it was possible to solve even more instances without branching. However, the cut separation times were substantial. The work is co-authored by Bjørn Petersen and David Pisinger and have been published as a book chapter in a book on recent advances within vehicle routing problems, see the chapter by Petersen et al. [11] in the book edited by Golden et al. [4].

Chapter 5: Clique Inequalities Applied to Vehicle Routing Problem with Time Windows. Following the line of work done on cutting planes in the master problem for the vehicle routing problem with time windows, this paper presents the application of the clique inequalities. Data structures to represent the cliques in a dynamic programming algorithm for the subproblem are described and experiments have been carried out. Although the experiments show some promise with regard to the quality of the lower bounds, the complexity of the subproblems are overwhelming. However, the paper contributes with the description of an application of a combinatorial cut for the master problem and could be the basis of further research in this area. This is joint work with Guy Desaulniers. The work has been submitted for publication.

Chapter 6: A Note on Cutting Planes in Dantzig-Wolfe Decompositions of Integer Programs. This paper describes a general methodology to apply cutting planes derived from the master problem variables. The main contribution is a description of the relation between the variables of the original formulation and the variables of the master problem and how it is possible to reformulate the original problem into an equivalent augmented problem such that valid inequalities of the master problem are represented. This is joint work with Guy Desaulniers and Jacques Desrosiers.

4.2 Part III: Conclusion

This part of the thesis concludes on the work presented in Part II.

Chapter 7: Conclusion. This chapter contains the concluding remarks and discussion of potential directions for future research.

Chapter 8: Summary in Danish. This chapter contains a Danish summary of the thesis.

4.3 Part IV: Other Contributions

This part of the thesis presents contributions that involves column generation or cutting planes, but is not directly connected with the main topic of applying cutting planes in the master problem. The first paper is a branch-and-cut algorithm for a specific type of the constrained shortest path problem, which is a very common subproblem encountered in many branch-and-cut-and-price algorithms. The two latter papers present branch-and-price algorithms for practical problems in the telecommunication and maritime industries.

Chapter 9: Branch-and-Cut Algorithm for the Elementary Shortest Path Problem with a Capacity Constraint. Elementary shortest path problems with resource constraints occur as a subproblem in many decompositions. This paper presents a very efficient branch-and-cut algorithm that regards a single capacity constraint. This is joint work with Mads Jepsen and Bjørn Petersen. The paper has been submitted for publication. An earlier version has been published as a technical report at the Department of Computer Science, University of Copenhagen, see Jepsen et al. [6].

Chapter 10: Optimal Routing with Failure Independent Path Protection. This paper presents a practical application of the column generation approach in the telecommunication industry. The problem to solve is to find a collection of paths in a telecommunication network, that covers a given bandwidth demand and follows a certain backup policy. Experimental results that the implemented backup strategy gives significant bandwidth savings. This is co-authored with Thomas K. Stidsen, Bjørn Petersen, Kasper Bonne, and Martin Zachariasen. The paper has been submitted for publication. An earlier version has been published in *Proceedings of the International Network Optimization Conference (INOC)* with additional authors Frans Rambach and Moritz Kiese, see Stidsen et al. [15].

Chapter 11: Liner Shipping Revenue Management with Empty Container Repositioning. A practical problem in the shipping industry is the amount of empty containers that has to be transported from high import areas to low import areas, e.g., from Europe and North America to Asia. This paper presents a column generation algorithm for revenue management of a liner shipping company that takes the transportation of empty containers into account. Experiments show that the column generation algorithm is superior compared to commercial solvers working in the original problem formulation. This is joint work with Berit Løfstedt and David Pisinger.

References

[1] N. Bélanger, G. Desaulniers, F. Soumis, and J. Desrosiers. Periodic airline fleet assignment with time windows, spacing constraints, and time dependent revenues. *European Journal of Operational Research*, 175(3):1754–1766, 2006. doi: 10.1016/j.ejor.2004.04.051.

[2] M. Christiansen, K. Fagerholt, B. Nygreen, and D. Ronen. Maritime transportation. *Handbooks in Operations Research & Management Science: Transportation*, 14:189–284, 2007.

[3] G. Desaulniers, J. Desrosiers, and M. M. Solomon, editors. *Column Generation*. Springer, 2005. doi: 10.1007/b135457.

[4] B. Golden, R. Raghavan, and E. Wasil, editors. *The Vehicle Routing Problem: Latest Advances and New Challenges*. Springer, 2008. doi: 10.1007/978-0-387-77778-8.

[5] D. Huisman. A column generation approach for the rail crew re-scheduling problem. *European Journal of Operational Research*, 180(1):163–173, 2007. doi: 10.1016/j.ejor.2006.04.026.

[6] M. Jepsen, B. Petersen, and S. Spoorendonk. A branch-and-cut algorithm for the elementary shortest path problem with a capacity constraint. Technical Report 08/01, DIKU Department of Computer Science, University of Copenhagen, Denmark, 2008.

[7] M. 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. doi: 10.1287/opre.1070.0449.

[8] M. E. Lübbecke and J. Desrosiers. Selected topics in column generation. *Operations Research*, 53(6):1007–1023, 2005. doi: 10.1287/opre.1050.0234.

[9] R. K. Martin. *Large Scale Linear and Integer Optimization: A Unified Approach*. Kluwer academic Publisher, 1999.

[10] G. L. Nemhauser and L. A. Wolsey. *Integer and Combinatorial Optimization*. John Wiley & Sons, Inc., 1988.

[11] B. Petersen, D. Pisinger, and S. Spoorendonk. Chvátal-Gomory rank-1 cuts used in a Dantzig-Wolfe decomposition of the vehicle routing problem with time windows. In B. Golden, R. Raghavan, and E. Wasil, editors, *The Vehicle Routing Problem: Latest Advances and New Challenges*, pages 397–420. Springer, 2008. doi: 10.1007/978-0-387-77778-8_18.

[12] M. L. Pinedo. *Planning and scheduling in Manufacturing and Services*. Springer, 2005. doi: 10.1007/b139030.

[13] Y. Pochet and L. A. Wolsey. *Production Planning by Mixed Integer Programming.* Springer, 2006. doi: 10.1007/0-387-33477-7.

[14] M. M. Solomon. Algorithms for the vehicle routing and scheduling problem with time window constraints. *Operations Research*, 35(2):234–265, 1987. doi: 10.1287/opre.35.2.254.

[15] T. Stidsen, B. Petersen, K. B. Rasmussen, S. Spoorendonk, M. Zachariasen, F. Rambach, and M. Kiese. Optimal routing with single backup path protection. In *Proceedings of the International Network Optimization Conference (INOC)*, pages 1–6, 2007.

[16] P. Toth and D. Vigo, editors. *The Vehicle Routing Problem.* SIAM, 2002.

[17] 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. doi: 10.1007/s10479-005-3455-9.

[18] L. A. Wolsey. *Integer Programming.* John Wiley & Sons, Inc., 1998.

Chapter 2

Preliminaries on Cut and Column Generation

Simon Spoorendonk
DIKU Department of Computer Science, University of Copenhagen

1 Basic Concepts

This chapter presents the Dantzig-Wolfe decomposition by Dantzig and Wolfe [6] for integer programs, see e.g., Wolsey [19], Lübbecke and Desrosiers [12], and how this is used in conjunction with column and cut generation in a branch-and-bound algorithm. This is not intended to be an in-depth survey, but primarily a guide to the understanding of the basic mechanisms of the Dantzig-Wolfe decomposition and column and cut generation in that context. For more details, see e.g., Barnhart et al. [3], Lübbecke and Desrosiers [12], Vanderbeck and Savelsbergh [17].

Consider an integer program (IP) where the constraint matrix is block-angular, meaning that the constraint matrix is divided in blocks of non-zero variable coefficients, where some of the blocks are independent with regard to the rows (constraints). This IP is referred to as the *original problem* or the *compact formulation*. When applying the Dantzig-Wolfe decomposition principle to the IP, an *integer programming master problem* containing the constraints given by the dependent blocks is obtained. The new variables correspond to feasible solutions subject to the constraints of each of the independent blocks. The linear relaxation of the master problem (the *linear programming master problem*) is found by *column generation*, where only a subset of variables from the master problem is considered. The reduced problem is denoted the *restricted linear programming master problem* and is iteratively expanded by solving a *subproblem* or *pricing problem* to identify the most promising feasible solutions of the independent blocks, such that the objective can be improved. When no improvements are possible, the relaxation is solved and a lower bound of the IP is obtained. When embedding this relaxation as a bounding procedure into a branch-and-bound algorithm, it is referred to as a *branch-and-price* or an *integer column generation* algorithm.

As in a branch-and-cut algorithm, the lower bound can be improved with the use of cutting planes and lead to what is known as a *branch-cut-and-price* (BCP) algorithm. The cutting plane approach is divided in two cases: i) where cutting planes are derived from valid inequalities of the original problem formulation, and ii) where the cutting planes are derived from valid inequalities of the master problem formulation. The former method is well studied and have been applied in many practical cases, as opposed to the latter method which is much less studied.

Throughout this chapter, we will exemplify the decomposition, and column and cut generation on the generalized assignment problem. The chapter is organized as follows: Section 2 contains a presentation of the Dantzig-Wolfe decomposition for an IP, in Section 3 the column generation approach is presented, in Section 4 we present the use of cutting planes, and in Section 5 a short description of branching rules is found.

2 Dantzig-Wolfe Decomposition

The general idea behind decomposing the IP is to use knowledge about the problem structure to reformulate the problem into another equivalent problem that is more desirable, e.g., with regard to complexity, computational running time, or quality of obtainable lower bounds. The Dantzig-Wolfe decomposition exploits a block-angular structure in problems, i.e., independent submatrices of the coefficient matrix. We specifically consider the discritization version (compared to the convexification approach) of the decomposition, see e.g., Vanderbeck [16], Vanderbeck and Savelsbergh [17], since the main goal of this thesis is to apply cutting planes on the integer master problem formulation. Consider the block-angular IP:

$$
\text{(IP)} \qquad
\begin{aligned}
\min \quad & \sum_{k \in K} c^k x^k && && (1)\\
\text{s.t.} \quad & \sum_{k \in K} A^k x^k \leq b && && (2)\\
& D^k x^k \leq d^k && k \in K && (3)\\
& x^k \in \mathbb{Z}_+^{n_k} && k \in K && (4)
\end{aligned}
$$

where K is the set of blocks, and A^k and D^k are the constraints submatrices for the dependent and the independent blocks respectively. Constraints (2) where the blocks depend on each other are referred to as the *linking* constraints. The matrix structure is depicted left in Figure 1.

The constraints (3) given by the $|K|$ independent matrix blocks and the integrality con-

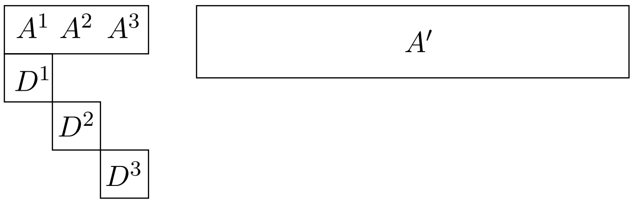


Figure 1: The block-angular IP (left) illustrated for blocks $K = \{1, 2, 3\}$ by the constraint matrices A^1, A^2, A^3 and D^1, D^2, D^3 is reformulated into the IPM (right) illustrated by the matrix A'. The matrix A' of the IPM contains fewer rows, but an exponential number of variables compared to the constraint matrix of the IP.

straints (4) are used to define the $|K|$ domains:

$$X^k = \left\{x^k \in \mathbb{Z}_+^{n_k} : D^k x^k \le d^k\right\} \qquad k \in K$$

The IP can be rewritten into the smaller (in the number of constraints) problem

$$\begin{aligned}
\min \; & \sum_{k\in K} c^k x^k \\
\text{s.t.} \; & \sum_{k\in K} A^k x^k \le b \\
& x^k \in X^k && k \in K
\end{aligned}$$

containing only the constraints given by the upper matrix blocks A^k for $k \in K$. That is, x^k must satisfy the constraints of block D^k before to be part of the smaller problem. For each $k \in K$, there exists a finite set of integer points $\{x^{kp}\}_{p\in P^k}$ and a finite set of integer rays $\{x^{kr}\}_{r\in R^k}$, see Nemhauser and Wolsey [14], such that

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

Substitute this into the IP, and the integer programming master problem (IPM) is obtained:

$$\text{(IPM)} \quad \begin{aligned}
\min \; & \sum_{k\in K}\Big(\sum_{p\in P^k} (c^k x^{kp})\lambda_{kp} + \sum_{r\in R^k} (c^k x^{kr})\lambda_{kr}\Big) \\
\text{s.t.} \; & \sum_{k\in K}\Big(\sum_{p\in P^k} (A^k x^{kp})\lambda_{kp} + \sum_{r\in R^k} (A^k x^{kr})\lambda_{kr}\Big) \le b \\
& \sum_{p\in P^k} \lambda_{kp} = 1 && k \in K \\
& \lambda_{kp} \in \{0,1\} && p \in P^k, k \in K \\
& \lambda_{kr} \in \mathbb{Z}_+ && r \in R^k, k \in K
\end{aligned}$$

This problem contains a smaller number of constraints than the IP problem, but it may have an exponential number of variables. The reformulation of the IP into the IPM is illustrated in Figure 1 with IP on the left and the IPM on the right.

It may happen that the master problem formulation is the starting point when solving problems with BCP algorithms, e.g., for the cutting stock problem. It may be desirable to have the original formulation at hand to derive cutting planes and determining branching rules. Villeneuve et al. [18] discuss how to obtain the original formulation without knowing it before-hand. Even if it is possible to obtain the original formulation, it is pointed out that, it may be difficult to find a suitable original formulation. However, this is an essential insight and is especially important when considering cutting planes and branching schemes.

Example 1. The generalized assignment problem (GAP) is in general given as: There are a number of agents and a number of tasks. Any agent can be assigned to perform any task, incurring some cost and profit that may vary depending on the agent-task assignment. Moreover, each agent has a budget, and the sum of the costs of tasks assigned to it are not allowed to exceed this budget. It is required to find an assignment in which all agents do not exceed their budget and total profit of the assignment is maximized.

Let M be the set of agents, let N be set of tasks, let p_{ij} be the profit of agent i performing task j, let w_{ij} be the cost of task j for agent i, let b_i be the budget of agent i, and let x_{ij} be the binary variable indicating if agent i performs task j. The mathematical model is given as:

$$\begin{aligned}
\max \quad & \sum_{i\in M}\sum_{j\in N} p_{ij}x_{ij} && && (5)\\
\text{s.t.} \quad & \sum_{i\in M} x_{ij} \leq 1 && j \in N && (6)\\
& \sum_{j\in N} w_{ij}x_{ij} \leq b_i && i \in M && (7)\\
& x_{ij} \in \{0,1\} && i \in M, j \in N && (8)
\end{aligned}$$

Decomposition is applied, keeping (6) as the linking constraints and (7) as the constraints in the $|M|$ block-angular matrices. Let $x_i = \{x_{ij} : j \in N\}$ be a solution (assignment of tasks j) for agent i, then the domain of feasible assignments for agent i are written as $X^i = \{x_i \in \{0,1\}^{|N|} : \sum_{j\in N} w_{ij}x_{ij} \leq b_i\}$ which is the 0-1 knapsack polytope. Noting that all X^i for $i \in M$ are finite, it is sufficient to consider only the extreme points $\{x_i^p\}_{p\in P^i}$. Hence, $x_i = \sum_{p\in P^i} x_i^p \lambda_{pi}, \sum_{p\in P^i} \lambda_{pi} = 1,$ and $\lambda_{pi} \in \{0,1\}$. By substitution, the IMP is obtained:

$$\begin{aligned}
\max \quad & \sum_{i\in M}\sum_{p\in P^i}(\sum_{j\in N} p_{ij}x_{ij}^p)\lambda_{pi} && && (9)\\
\text{s.t.} \quad & \sum_{i\in M}\sum_{p\in P^i}(x_{ij}^p)\lambda_{pi} \leq 1 && j \in N && (10)\\
& \sum_{p\in P^i} \lambda_{pi} = 1 && i \in M && (11)\\
& \lambda_{pi} \in \{0,1\} && p \in P^i, i \in M && (12)
\end{aligned}$$

3 Column Generation

The LP relaxation of the IPM denoted the LP master problem (LPM) can be used to calculate lower bounds when solving the IPM with a branch-and-bound algorithm. In the LPM the decision variables λ_{kp} and λ_{kr} of the IPM are all continuous. That is:

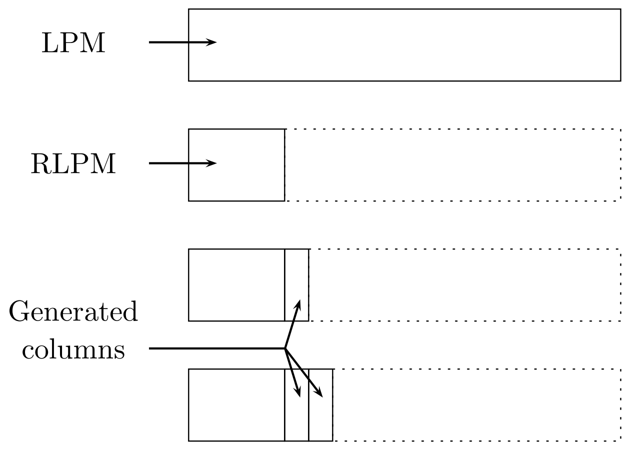


Figure 2: Column generation. The LPM explicitly contains all variables whereas the RLPM contains a subset. The RLPM grows in the number of variables as new columns are generated.

$$
\text{(LPM)}\quad
\begin{aligned}
\min\ & \sum_{k\in K}\Big(\sum_{p\in P^k}(c^k x^{kp})\lambda_{kp} + \sum_{r\in R^k}(c^k x^{kr})\lambda_{kr}\Big) && && (13)\\
\text{s.t.}\ & \sum_{k\in K}\Big(\sum_{p\in P^k}(A^k x^{kp})\lambda_{kp} + \sum_{r\in R^k}(A^k x^{kr})\lambda_{kr}\Big) \le b && && (14)\\
& \sum_{p\in P^k}\lambda_{kp} = 1 && k\in K && (15)\\
& \lambda_{kp}\ge 0 && p\in P^k, k\in K && (16)\\
& \lambda_{kr}\ge 0 && r\in R^k, k\in K && (17)
\end{aligned}
$$

To overcome the huge number of columns in the LPM, the idea is to enumerate a small subset of the columns and then generate columns as needed. Only a subset of variables in the LPM is considered and this gives rise to the restricted LP master problem (RLPM). Generating new columns is done by solving a subproblem (SP) for each of the domains X^k by using information from the current solution of the RLPM, more precisely, the value of the dual vector π and dual value μ^k. Figure 2 illustrates the relation between the LPM and the RLPM with regard to the number of variables, and illustrates how the RLPM is growing in the number of columns for each iteration of the column generation process.

By modifying the objective function in the SPs, one can identify columns with negative reduced cost that can become part of the RLPM. Recall the simplex method that when using Dantzig's pivot rule the most negative reduced cost non-basic variable is chosen to enter the basis of an LP minimization problem, see Hamacher and Klamroth [9] for details. Let π and μ be the dual vectors for constraints (14) and (15), respectively. The reduced cost of a column in the kth SP can be calculated as:

$$
\begin{aligned}
\bar{c}_{kp} &= (c^k - \pi A^k)x^{kp} - \mu_k && p\in P^k\\
\bar{c}_{kr} &= (c^k - \pi A^k)x^{kr} && r\in R^k
\end{aligned}
$$

for the extreme points p and rays r, respectively. To find the column with the least cost z_k, subproblems $k \in K$ are solved:

$$\text{(SP)} \qquad \begin{aligned} z_k = \min\ & (c^k - \pi A^k)x - \mu_k \\ \text{s.t.}\ & D^k x \leq d^k \\ & x \in \mathbb{Z}_+^n \end{aligned}$$

The column generation terminates when no columns with negative reduced cost exists, that is, when $z_k \geq 0$ for all the SPs $k \in K$. When the SP is finite an the objective value is negative, i.e., $z_k < 0$, then the solution is an extreme point x^{kp} that gives rise to the column $[(c^k x^{kp}), (A^k x^{kp}), 1]^T$ in the RLPM. If the SP is unbounded, i.e., $z_k = -\infty$, it gives rise to an extreme ray x^{kr} and the column $[(c^k x^{kr}), (A^k x^{kr}), 0]^T$ is added. If an optimal solution is obtained for each SP and an upper bound UB of the IPM is known, it is possible to obtain a lower bound LB of the IPM calculated as $LB = UB - \sum_{k \in K} z_k$.

A general method to speed up the convergence of the column generation process, is to solve the subproblems heuristically and add several negative reduced cost columns in the same iteration. Note, that the lower bound calculation for the IPM is only valid for true lower bounds or optimal solutions for the SPs. In practice, Dantzig-Wolfe decomposition is often used such that the subproblems can be solved with combinatorial algorithms, e.g., dynamic programming algorithms, that are faster than solving subproblems with general IP solvers. The column generation method is also denoted *delayed* column generation to distinguish it from an a priori enumeration of all columns for the entire master problem.

Example 2. Continuing with the GAP. The LPM is written as the linear relaxation of the IPM formulation (9)-(12) of the GAP:

$$\max \sum_{i \in M} \sum_{p \in P^i} (\sum_{j \in N} p_{ij} x_{ij}^p) \lambda_{pi} \tag{18}$$

$$\text{s.t.} \sum_{i \in M} \sum_{p \in P^i} (x_{ij}^p) \lambda_{pi} \leq 1 \qquad j \in N \tag{19}$$

$$\sum_{p \in P^i} \lambda_{pi} = 1 \qquad i \in M \tag{20}$$

$$\lambda_{pi} \leq 0 \qquad p \in P^i, i \in M \tag{21}$$

When considering the RLPM, the subproblem to solve is a minimization version of the 0-1 knapsack problem for each agent. Let π and μ be the dual vectors of (19) and (20) respectively, then the subproblem $i \in M$ is given as:

$$\min \sum_{j \in N} (-p_{ij} - \pi_j) x_{ij} - \mu_i \tag{22}$$

$$\text{s.t.} \sum_{j \in N} w_{ij} x_{ij} \leq b_i \tag{23}$$

$$x_{ij} \in \{0, 1\} \qquad j \in N \tag{24}$$

4 Cutting Planes

Cutting planes are valid inequalities for the IP or the IPM that cuts of a part of the fractional solution given by the RLPM, and can thereby increase the objective value. Adding cutting

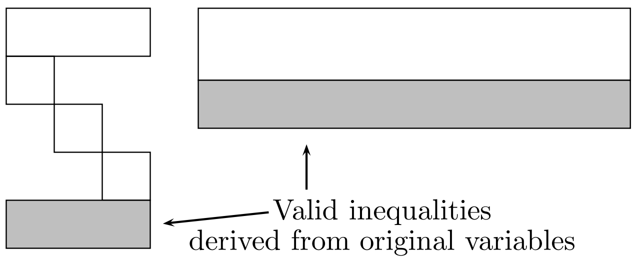


Figure 3: Following the same approach as illustrated on Figure 1, the valid inequalities (light gray) for the IP are reformulated in the IPM. This makes it uncomplicated to handle dual variables of the valid inequalities.

planes to the IPM can greatly improve the lower bound obtained by the RLPM and result in fewer branch nodes to be examined in the branch tree.

Valid inequalities for the original problem, the IP, can be readily used in a BCP algorithm by considering the valid inequalities as part of the original problem formulation and therefore also part of the reformulation into the IPM. That is, valid inequalities for the original problem are still valid when reformulated in the master problem. This is explained in details in Villeneuve et al. [18]. Given a valid inequality for the IP:

$$\sum_{k\in K} \alpha^k x^k \leq \beta$$

that is a linear combination of the original variables, it decomposes in the IPM to be:

$$\sum_{k\in K}(\sum_{p\in P^k}(\alpha^k x^{kp})\lambda_{kp} + \sum_{r\in R^k}(\alpha^k x^{kr})\lambda_{kr}) \leq \beta \tag{25}$$

Let ν be the dual variable of (25). Then the reduced cost of a column in subproblem k is calculated as:

$$\begin{aligned} \bar{c}_{kp} &= (c^k - \pi A^k - \nu\alpha^k)x^{kp} - \mu_k && p \in P^k \\ \bar{c}_{kr} &= (c^k - \pi A^k)x^{kr} - \mu_k && r \in R^k \end{aligned}$$

for the extreme points p and rays r, respectively. Figure 3 illustrates this reformulation for a given set of valid inequalities added before the decomposition and how they are decomposed into constraints in the master problem.. Figure 4 illustrates the iterative process in a BCP algorithm, where both the number of columns and rows grow over time. When embedded into a BCP algorithm, the valid inequalities based on the IP only affects the cost of the variables in the subproblem. Hence, in most cases this approach does not affect any specially designed algorithms for solving the subproblem, making the cutting planes a very powerful tool. This approach is widely used, see e.g., Fukasawa et al. [8] for the capacitated vehicle routing problem, Kohl et al. [11] for the vehicle routing with time windows, Alves and Valério de Carvalho [1] for the multiple length cutting stock problem.

Consider the valid inequality for the IPM:

$$\sum_{k\in K}(\sum_{p\in P^k}\alpha^{kp}\lambda_{kp} + \sum_{r\in R^k}\alpha^{kr}\lambda_{kr}) \leq \beta \tag{26}$$

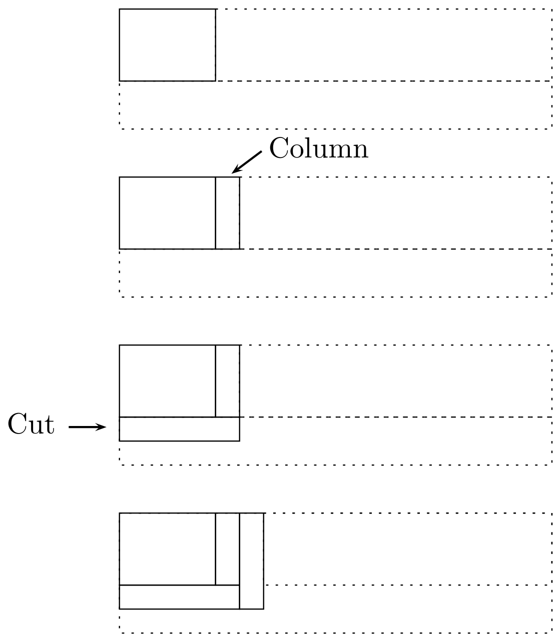


Figure 4: The RLPM is growing in both dimensions, when both column and cut generation is done. Naturally, this adds to the complexity of reoptimizing the RLPM, but good lower bounds are often obtained. The crux of the matter is how to correctly determine the coefficient in the cuts, when generating columns such that the reduced cost can be calculated.

Let σ be the dual variable of (26), then the reduced cost of a column in subproblem k is calculated as:

$$\begin{aligned} \bar{c}_{kp} &= (c^k - \pi A^k)x^{kp} - \mu_k - \sigma\alpha^{kp} && p \in P^k \\ \bar{c}_{kr} &= (c^k - \pi A^k)x^{kr} - \sigma\alpha^{kr} && r \in R^k \end{aligned}$$

for the extreme points p and rays r, respectively. Observe, that when $\alpha^k x^{kp} = \alpha^{kp}$ and $\alpha^k x^{kr} = \alpha^{kr}$, then (25) is equivalent to (26). Otherwise the calculation of the last term $-\sigma\alpha^{kp}$ (or $-\sigma\alpha^{kr}$) in the objective function of the above subproblems may not be straightforward, since α^{kp} (or α^{kr}) can depend on A^k and point x^{kp} (or ray x^{kr}).

The iterative process (as depicted in Figure 4) for adding cuts on the master problem variables (26) are similar to the process of applying cutting planes derived from the original variables (25). But as mentioned above, the calculation of the cut coefficient for the generated problems may be more problematic than for the cuts derived on the original variables. Recall that the IPM is a reformulation of the IP problem, which means that valid inequalities for the IPM cannot necessarily be expressed in the original variables in an obvious way. In order to formulate such a valid inequality (26) in terms of original variables, it may be necessary to add additional variables and constraints to the IP. This may result in more complicated subproblems, because the added variables and constraints are decomposed into the subproblems thereby raising their complexity and changing the characteristics. As a result, any special purpose algorithm designed to solve the old subproblems may be unusable when considering the new augmented subproblems.

BCP algorithms that make use of cutting planes derived from the IPM formulation have received less attention in the literature compared to adding cutting planes derived from the

IP formulation. To our best knowledge the work in this area (not considering this thesis) is limited to the following contributions: Nemhauser and Park [13] present circuit constraints for the edge coloring problem, where the subproblem is solved with a branch-and-cut algorithm. Belov and Scheithauer [4] present Chvátal-Gomory cuts for the one-dimensional cutting stock problem and the multiple length one-dimensional cutting stock problem, where the subproblem is solved with an approximation to obtain a lower bound used in a branch-and-bound algorithm. In Belov and Scheithauer [5], this is extended to the two-dimensional two-stage cutting stock problem and also the Gomory mixed-integer cuts are added. Baldacci et al. [2] present clique inequalities for the capacitated vehicle routing problem where the subproblem is solved by an enumeration scheme based on dynamic programming. Desaulniers et al. [7] apply a subset of the Chvátal-Gomory cuts for the vehicle routing problem with time windows using a method introduced by Jepsen et al. [10] (see Chapter 3), where the subproblem is solved using a dynamic programming algorithm. However, none of the papers discusses the connection between the original formulation and the master problem although this can be considered a key issue in order to prove correctness of the valid inequalities and to develop algorithms that are more efficient than brute-force methods. This is the main topic of this thesis, and in Chapters 3 to 5 examples with cut and column generation on the vehicle routing problem with time windows are given. This is concluded in Chapter 6 by a description of a general framework for cutting planes in BCP algorithms.

Example 3. A valid cutting plane for the original formulation (5)-(8) of the GAP is the cover inequalities, see Wolsey [19]. For $i \in N$ let $C \subseteq M$ be cover where $\sum_{j\in C} w_{ij} > b_i$. A cover inequality for i is given as:

$$\sum_{j\in C} x_{ij} \le |C| - 1 \tag{27}$$

Inequality (27) decomposes into the IPM formulation (9)-(12) as:

$$\sum_{p\in P^i} (\sum_{j\in C} x^p_{ij})\lambda_{pi} \le |C| - 1 \tag{28}$$

Given the dual variable ν of (28), subproblem (22)-(24) for $i \in M$ is modified to:

$$\min_{x_i\in X^p} \sum_{j\in N}(-p_{ij} - \pi_j)x_{ij} - \sum_{j\in C}(w_{ij}\nu)x_{ij} - \mu_i$$

Note, that since the cover inequalities are separable for each agent $i \in M$, it is possible to apply these cuts directly into the SP. Indeed, the cuts do not provide any improvement in the RLPM since all columns are already valid assignments.

The IPM constraints (9)-(12) of the GAP can be recognized as a set packing polytope. A well known valid inequality for this polytope is the clique inequality, see Wolsey [19]. Let $Q \subseteq \bigcup_{i\in M} P^i$ be a clique such that there for all pairs of paths in $p, q \in Q, p \in P^i, q \in P^h$ and $p \neq q$ exist a $j \in N$ where $x^p_{ij} = x^q_{hj}$. Then the clique inequality is given as:

$$\sum_{p\in Q} \lambda_{pi} \le 1 \tag{29}$$

Given the dual variable σ of (29), subproblem (22)-(24) for $i \in M$ is modified to:

$$\min_{x_i\in X^p} \sum_{j\in N}(-p_{ij} - \pi_j)x_{ij} - \mu_i - \left\{ \begin{array}{ll} \sigma & x_i \in Q \\ 0 & \text{otherwise} \end{array} \right.$$

This is clearly a non-linear objective function, and it may not seem obvious how to calculate this last term in the 0-1 knapsack problem. In Spoorendonk and Desaulniers [15] (Chapter 5) an example is given on how to handle clique inequalities in the vehicle routing problem with time windows.

5 Branching

To obtain integrality, it may be necessary to perform branching. This is not a main focus area of this thesis and will only be handled briefly.

For traditional integer programs, branching may be done by choosing an integer variable that has a fractional value and create two branches. One branch where the value of the variables is less than or equal to the rounded down value of the variables, and another branch where the variables must have values greater than or equal to the rounded up value. In BCP algorithms, a branch on an original variable can be done in the master problem the same way as adding cutting planes on the variables from the original formulation as described in Section 4. For example, branching on variable x_i^k from the original formulation with the fractional value $a \notin \mathbb{Z}$ results in cutting planes

$$x_i^k \leq \lfloor a \rfloor \quad \vee \quad x_i^k \geq \lceil a \rceil$$

that are decomposed into the two branch cutting planes

$$\sum_{k \in K} (\sum_{p \in P^k} (x_i^{kp}) \lambda_{kp} + \sum_{r \in R^k} (x_i^{kr}) \lambda_{kr}) \leq \lfloor a \rfloor$$

and

$$\sum_{k \in K} (\sum_{p \in P^k} (x_i^{kp}) \lambda_{kp} + \sum_{r \in R^k} (x_i^{kr}) \lambda_{kr}) \geq \lceil a \rceil$$

in the master problem.

Branching directly on the master problem variables can give rise to the complications similar to cutting on master problem variables. Hence, this approach must be followed carefully to avoid an explosion in running time due to complication of the subproblems. See Vanderbeck [16], Villeneuve et al. [18] for a detailed discussion on branching schemes.

References

[1] C. Alves and J.M. Valério de Carvalho. A stabilized branch-and-price-and-cut algorithm for the multiple length cutting stock problem. *Computers & Operations Research*, 35(4): 1315–1328, 2008. doi: 10.1016/j.cor.2006.08.014.

[2] R. Baldacci, N. Christofides, and A. Mingozzi. An exact algorithm for the vehicle routing problem based on the set partitioning formulation with additional cuts. *Mathematical Programming*, 115(2):351–385, 2008. doi: 10.1007/s10107-007-0178-5.

[3] 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(3):316–329, 1998. doi: 10.1287/opre.46.3.316.

[4] G. Belov and G. Scheithauer. A cutting plane algorithm for the one-dimensional cutting stock problem with multiple lengths. *European Journal of Operations Research*, 141(2): 274–294, 2002. doi: 10.1016/S0377-2217(02)00125-X.

[5] G. Belov and G. Scheithauer. A branch-and-cut-and-price algorithm for one-dimensional stock cutting and two-dimensional two-stage cutting. *European Journal of Operations Research*, 171(1):85–106, 2006. doi: 10.1016/j.ejor.2004.08.036.

[6] G. B. Dantzig and P. Wolfe. Decomposition principle for linear programs. *Operations Research*, 8(1):101–111, 1960. doi: 10.1287/opre.8.1.101.

[7] G. Desaulniers, F. Lessard, and A. Hadjar. Tabu search, partial elementarity, and generalized k-path inequalities for the vehicle routing problem with time windows. *Transportation Science*, 42(3):387–404, 2008. doi: 10.1287/trsc.1070.0223.

[8] R. Fukasawa, H. Longo, J. Lysgaard, M. Poggi de Aragão, M. Reis, E. Uchoa, and R.F. Werneck. Robust branch-and-cut-and-price for the capacitated vehicle routing problem. *Mathematical Programming*, 106(3):491–511, 2006. doi: 10.1007/s10107-005-0644-x.

[9] H. W. Hamacher and K. Klamroth. *Linear and Network Optimization.* Vieweg, 2000.

[10] M. 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. doi: 10.1287/opre.1070.0449.

[11] N. Kohl, J. Desrosiers, O. B. G. Madsen, M. M. Solomon, and F. Soumis. 2-path cuts for the vehicle routing problem with time windows. *Transportation Science*, 33(1):101–116, 1999. doi: 10.1287/trsc.33.1.101.

[12] M. E. Lübbecke and J. Desrosiers. Selected topics in column generation. *Operations Research*, 53(6):1007–1023, 2005. doi: 10.1287/opre.1050.0234.

[13] G. Nemhauser and S. Park. A polyhedral approach to edge coloring. *Operations Research Letters*, 10(6):315–322, 1991. doi: 10.1016/0167-6377(91)90003-8.

[14] G. L. Nemhauser and L. A. Wolsey. *Integer and Combinatorial Optimization.* John Wiley & Sons, Inc., 1988.

[15] S. Spoorendonk and G. Desaulniers. Clique inequalities applied to vehicle routing problem with time windows. 2008.

[16] F. Vanderbeck. On Dantzig-Wolfe decomposition in integer programming and ways to perform branching in a branch-and-price algorithm. *Operation Research*, 48(1):111–128, 2000. doi: 10.1287/opre.48.1.111.12453.

[17] F. Vanderbeck and M. W. P. Savelsbergh. A generic view of Dantzig-Wolfe decomposition in mixed integer programming. *Operations Research Letters*, 34(3):296–306, 2006. doi: 10.1016/j.orl.2005.05.009.

[18] 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. doi: 10.1007/s10479-005-3455-9.

[19] L. A. Wolsey. *Integer Programming.* John Wiley & Sons, Inc., 1998.

Part II

Cut and Column Generation

Chapter 3

Subset-Row Inequalities Applied to the Vehicle Routing Problem with Time Windows

Mads Jepsen
DIKU Department of Computer Science, University of Copenhagen

Bjørn Petersen
DIKU Department of Computer Science, University of Copenhagen

Simon Spoorendonk
DIKU Department of Computer Science, University of Copenhagen

David Pisinger
DIKU Department of Computer Science, University of Copenhagen

Abstract

This paper presents a branch-and-cut-and-price algorithm for the vehicle routing problem with time windows. The standard Dantzig-Wolfe decomposition of the arc flow formulation leads to a set partitioning problem as the master problem and an elementary shortest path problem with resource constraints as the pricing problem. We introduce the subset-row inequalities, which are Chvatal-Gomory rank-1 cuts based on a subset of the constraints in the master problem. Applying a subset-row inequality in the master problem increases the complexity of the label-setting algorithm used to solve the pricing problem since an additional resource is added for each inequality. We propose a modified dominance criterion that makes it possible to dominate more labels by exploiting the step-like structure of the objective function of the pricing problem. Computational experiments have been performed on the Solomon benchmarks where we were able to close several instances. The results show that applying subset-row inequalities in the master problem significantly improves the lower bound, and in many cases makes it possible to prove optimality in the root node.

Published in the journal *Operations Research*, 2008

1 Introduction

The vehicle routing problem with time windows (VRPTW) can be described as follows: A set of customers, each with a demand, needs to be serviced by a number of vehicles all starting and ending at a central depot. Each customer must be visited exactly once within a given time window, and the capacity of the vehicles must not be exceeded. The objective is to service all customers traveling the least possible distance. In this paper we consider a homogenous fleet, i.e., all vehicles are identical.

The standard Dantzig-Wolfe decomposition of the arc flow formulation of the VRPTW is to split the problem into a master problem (a set partitioning problem) and a pricing problem (an elementary shortest path problem with resource constraints (ESPPRC), where capacity and time are the constrained resources). A restricted master problem can be solved with delayed column generation and embedded in a branch-and-bound framework to ensure integrality. Applying cutting planes either in the master or the pricing problem leads to a branch-and-cut-and-price algorithm (BCP).

Kohl et al. [23] implemented a successful BCP algorithm for the VRPTW by applying subtour elimination constraints and *two-path* cuts. Cook and Rich [8] generalized the two-path cuts to the *k-path* cuts. Common for these BCP algorithms is that all applied cuts are valid inequalities for the VRPTW, i.e., the *original* arc flow formulation, and contain a structure making it possible to handle values of the dual variables in the pricing problem without increasing the complexity of the problem. Fukasawa et al. [17] refer to this as a *robust* approach in their paper, where a range of valid inequalities for the capacitated vehicle routing problem are used in a BCP algorithm. The topic of column generation and BCP algorithms has been surveyed by Barnhart et al. [1] and Lübbecke and Desrosiers [27].

Dror [13] showed that the ESPPRC is strongly $\mathcal{NP}$-hard, hence a relaxation of the ESPPRC was used as a pricing problem in earlier BCP approaches for the VRPTW. The relaxed pricing problem where non-elementary paths are allowed is denoted the shortest path problem with resource constraints (SPPRC) and can be solved in pseudo-polynomial time using a label-setting algorithm, which was initially done by Desrochers [11]. To improve lower bounds of the master problem, Desrochers et al. [12] used 2-cycle elimination, which was later extended by Irnich and Villeneuve [20] to k-cycle elimination (k-cyc-SPPRC) where cycles containing k or less nodes are not permitted.

Beasley and Christofides [2] proposed to solve the ESPPRC using Lagrangian relaxation. However, recently label-setting algorithms have become the most popular approach to solve the ESPPRC; see e.g. Dumitrescu [14] and Feillet et al. [16]. When solving the ESPPRC with a label-setting algorithm a binary resource for each node is added, which increases the complexity of the algorithm compared to solving the SPPRC or the k-cyc-SPPRC. Righini and Salani [32] developed a label-setting algorithm using the idea of Dijkstra's bi-directional shortest path algorithm that expands both forward and backward from the depot and connects routes in the middle, thereby potentially reducing the running time of the algorithm. Furthermore Righini and Salani [32] and Boland et al. [3] proposed a decremental state space algorithm that iteratively solves a SPPRC by applying resources that force nodes to be visited at most once. Recently Chabrier [5], Danna and Le Pape [9], and Salani [33] successfully solved several previously unsolved instances of the VRPTW from the benchmarks of Solomon [34] using a label-setting algorithm for the ESPPRC.

In this paper, we extend the BCP framework to include valid inequalities for the master problem, more specifically by applying the subset-row (SR) inequalities to the set partitioning master problem. Nemhauser and Park [28] developed a similar BCP algorithm for the edge coloring problem, but to our knowledge no such algorithms for the VRPTW have been presented. Applying the SR inequalities leads to an increased complexity of the pricing problem since each inequality is represented by an additional resource. To improve the performance of the label-setting algorithm, we introduce a modified dominance criterion that handles the reduced cost calculation in a reasonable way. Moreover, the SR inequalities potentially provide better lower bounds and smaller branch trees.

The paper is organized as follows: In Section 2 we give an overview of the Dantzig-Wolfe decomposition of the VRPTW and describe how to calculate the reduced cost of columns when column generation is used. In Section 3 we introduce the SR inequalities and show that the separation problem is $\mathcal{NP}$-complete. In Section 4 we review the basics of a label-setting algorithm for solving the ESPPRC and show how to handle the modified pricing problem in the same label-setting algorithm. For details regarding label-setting algorithms (including bi-directionality) we refer to Desaulniers et al. [10], Irnich and Desaulniers [19], Irnich [18], Righini and Salani [31]. An algorithmic outline and computational results, using the Solomon benchmark instances, are presented in Section 5. Section 6 concludes the paper.

2 Decomposition

Let C be the set of customers, let the set of nodes be $V = C \cup \{o, o'\}$ where $\{o\}$ denotes the depot at the start of the routes and $\{o'\}$ denotes the depot at the end; and let $E = \{(i,j) : i,j \in V,\ i \neq j\}$ be the edges between the nodes. Let K be the set of vehicles with $|K|$ unbounded, each vehicle having capacity D, and let d_i be the demand of customer $i \in C$ and $d_o = d_{o'} = 0$. Let a_i be the beginning and b_i be the end of the time window for node $i \in V$. Let s_i be the service time for $i \in V$ and let t_{ik} be the time vehicle $k \in K$ visits node $i \in V$, if k visits i. Let c_{ij} be the travel cost on edge $(i,j) \in E$ and let x_{ijk} be a variable indicating whether vehicle $k \in K$ traverses edge $(i,j) \in E$. Last let $\tau_{ij} = c_{ij} + s_i > 0$ be the travel time on edge $(i,j) \in E$ plus the service time of customer i. The three-index flow model (Toth and Vigo [36]) for the VRPTW is:

$$\min \sum_{k \in K} \sum_{(i,j) \in E} c_{ij} x_{ijk} \tag{1}$$

$$\text{s.t.} \sum_{k \in K} \sum_{(i,j) \in \delta^+(i)} x_{ijk} = 1 \qquad \forall i \in C \tag{2}$$

$$\sum_{(i,j) \in \delta^+(o)} x_{ijk} = \sum_{(i,j) \in \delta^-(o')} x_{ijk} = 1 \qquad \forall k \in K \tag{3}$$

$$\sum_{(j,i) \in \delta^-(i)} x_{jik} - \sum_{(i,j) \in \delta^+(i)} x_{ijk} = 0 \qquad \forall i \in C,\ \forall k \in K \tag{4}$$

$$\sum_{(i,j) \in E} d_i x_{ijk} \leq D \qquad k \in K \tag{5}$$

$$a_i \leq t_{ik} \leq b_i \qquad \forall i \in V,\ \forall k \in K \tag{6}$$

$$x_{ijk}(t_{ik} + \tau_{ij}) \leq t_{jk} \qquad \forall (i,j) \in E,\ \forall k \in K \tag{7}$$

$$x_{ijk} \in \{0,1\} \qquad \forall (i,j) \in E,\ \forall k \in K \tag{8}$$

Here (2) ensures that every customer $i \in C$ is visited, while (3) ensures that each route starts and ends in the depot. Constraint (4) maintains flow conservation, while (5) ensures that the capacity of each vehicle is not exceeded. Constraints (6), (7) ensure that the time windows are satisfied. Note that (7) together with the assumption that $\tau_{ij} > 0$ for all $(i,j) \in E$ eliminates sub-tours. The last constraints define the domain of the arc flow variables. Note that a zero-cost edge $x_{oo'k}$ between the start and end depot must be present for all vehicles for (3) to hold if not all vehicles are used.

The standard Dantzig-Wolfe decomposition of the VRPTW, see e.g. Desrochers et al. [12], leads to the following master problem:

$$\min \sum_{p \in P} \sum_{(i,j) \in E} c_{ij} \alpha_{ijp} \lambda_p \tag{9}$$

$$\text{s.t} \sum_{p \in P} \sum_{(i,j) \in \delta^+(i)} \alpha_{ijp} \lambda_p = 1 \qquad \forall i \in C \tag{10}$$

$$\lambda_p \in \{0,1\} \qquad \forall p \in P \tag{11}$$

where P is the set of all feasible routes, the binary constant α_{ijp} is one if and only if edge (i,j) is used by route $p \in P$, and the binary variable λ_p indicates whether route p is used. The master problem can be recognized as a set partitioning problem, and the LP relaxation may be solved using delayed column generation. Let $\pi \in \mathbb{R}$ be the dual variables of (10) and let $\pi_0 = 0$. Then the reduced cost of a route p is:

$$\overline{c}_p = \sum_{(i,j) \in E} c_{ij} \alpha_{ijp} - \sum_{(i,j) \in E} \pi_j \alpha_{ijp} = \sum_{(i,j) \in E} (c_{ij} - \pi_j) \alpha_{ijp} \tag{12}$$

The pricing problem becomes an ESPPRC where the cost of each edge is $\overline{c}_{ij} = c_{ij} - \pi_j$ for all edges $(i,j) \in E$. When applying cuts during column generation we will distinguish between valid inequalities for the VRPTW constraints (2)-(8) and valid inequalities for the set partitioning constraints (10)-(11).

Consider a valid inequality for the VRPTW constraints (2)–(8) in terms of the arc flow variables x:

$$\sum_{k \in K} \sum_{(i,j) \in E} \beta_{ij} x_{ijk} \leq \beta_0 \tag{13}$$

When decomposed into the master problem, inequality (13) is reformulated as:

$$\sum_{p \in P} \sum_{(i,j) \in E} \beta_{ij} \alpha_{ijp} \lambda_p \leq \beta_0 \tag{14}$$

Let $\mu \leq 0$ be the dual variable of (14). The reduced cost of a column p is then

$$\begin{aligned} \overline{c}_p &= \sum_{(i,j) \in E} c_{ij} \alpha_{ijp} - \sum_{(i,j) \in E} \pi_j \alpha_{ijp} - \mu \sum_{(i,j) \in E} \beta_{ij} \alpha_{ijp} \\ &= \sum_{(i,j) \in E} (c_{ij} - \pi_j - \mu \beta_{ij}) \alpha_{ijp} \end{aligned} \tag{15}$$

Compared to (12) an additional coefficient $\mu\beta_{ij}$ is subtracted from the cost of edge (i,j) and the complexity of the pricing problem remains unchanged if we use the edge costs $\overline{c}_{ij} = c_{ij} - \pi_j - \mu\beta_{ij}$.

Now, consider adding a valid inequality for the set partitioning master problem (10)–(11) that cannot be written as a linear combination of the arc flow variables:

$$\sum_{p \in P} \beta_p \lambda_p \leq \beta_0 \tag{16}$$

Let $\sigma \leq 0$ be the dual variable of (16). The reduced cost of a column p is:

$$\hat{c}_p = \overline{c}_p - \sigma\beta_p = \sum_{(i,j)\in E} \overline{c}_{ij}\alpha_{ijp} - \sigma\beta_p \tag{17}$$

In addition to the reduced cost computed for a column p in (15) the cost $-\sigma\beta_p$ must be considered. To reflect the possible extra cost $-\sigma\beta_p$ it may be necessary to modify the pricing problem by adding constraints or variables, thereby increasing its complexity.

3 Subset-Row Inequalities

The set of valid inequalities for the set packing problem is a subset of the set of valid inequalities for the set partitioning problem since the latter problem is a special case of first-mentioned. Two well-known valid inequalities for the set packing problem are the clique and the odd-hole inequalities, where the first is known to be facet-defining for the set partitioning problem (Nemhauser and Wolsey [29]).

Since the master problem is a set partitioning problem, it would be obvious to go in this direction when looking for valid inequalities for the master problem. Consider the separation of a clique or an odd-hole inequality. The undirected conflict graph $G'(P, E')$ is defined as follows: Each column is a vertex in G' and the edge set is given as:

$$E' = \left\{ (p,q) : \sum_{(i,j)\in\delta^+(i)} \alpha_{ijp} = 1 \wedge \sum_{(i,j)\in\delta^+(i)} \alpha_{ijq} = 1,\ i \in C,\ p,q \in P,\ p \neq q \right\}$$

That is, an edge is present if the two columns p and q have coefficient one in the same row. In a VRPTW context it reads: Two routes are conflicting if they are visiting the same customer. A clique in G' leads to the valid clique inequality:

$$\sum_{p \in \hat{P}} \lambda_p \leq 1 \tag{18}$$

where $\hat{P} \subseteq P$ are the columns corresponding to the vertices of a clique in G'. A cycle visiting an odd number of vertices P in G' leads to the valid odd-hole inequality:

$$\sum_{p \in \hat{P}} \lambda_p \leq \left\lfloor \frac{|\hat{P}|}{2} \right\rfloor \tag{19}$$

where $\hat{P} \subseteq P$ are the columns corresponding to the vertices visited on the cycle in G'. However, when column generation is applied, it is not obvious how to reflect the reduced cost of (18) or (19) in the pricing problem since there is no specific knowledge of the columns of the master problem when solving the pricing problem.

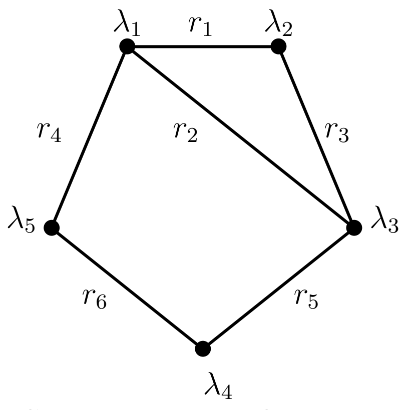
Example 1

SR inequalities derived from the conflict graph of a set packing problem. In the LP-solution to $A\lambda \leq 1$ all λ variables are $\frac{1}{2}$, which results in two violated SR inequalities:

- With $|S| = 3$ and $k = 2$ due to variables λ_1, λ_2, and λ_3 giving the set of rows $S = \{r_1, r_2, r_3\}$
- With $n = 5$ and $k = 2$ due to variables λ_1, λ_2, λ_3, λ_4, and λ_5 giving the set of rows $S = \{r_1, r_3, r_4, r_5, r_6\}$

	λ_1	λ_2	λ_3	λ_4	λ_5		
r_1	1	1				$\leq$	1
r_2	1		1			$\leq$	1
r_3		1	1			$\leq$	1
r_4	1				1	$\leq$	1
r_5			1	1		$\leq$	1
r_6				1	1	$\leq$	1

Set packing problem $A\lambda \leq 1$.



Corresponding conflict graph.

Inspired by the above inequalities (18) and (19) we introduce the *subset-row inequalities* (SR inequalities). These inequalities are specifically linked to the rows (rather than the columns) of the set packing problem, hence making it possible to identify the coefficient of a column in an SR inequality.

Definition 1. *Consider the set packing structure*

$$X = \{\lambda \in \mathbb{B}^{|P|} : A\lambda \leq 1\} \tag{20}$$

with the set of rows M and columns P, and a $|M| \times |P|$ binary coefficient matrix A. The SR inequality is defined as:

$$\sum_{p \in P} \left\lfloor \frac{1}{k} \sum_{i \in S} \alpha_{ip} \right\rfloor \lambda_p \leq \left\lfloor \frac{|S|}{k} \right\rfloor \tag{21}$$

where $S \subseteq M$ and $0 < k \leq |S|$.

Example 1 illustrates some SR inequalities derived from the conflict graph of a set packing problem.

Given a column $p \in P$ we need to have $\sum_{i \in S} \alpha_{ip} \geq k$ to get a non-zero coefficient of λ_p in (21). For the master problem of VRPTW the coefficient matrix can be translated as $\alpha_{ip} = \sum_{(i,j) \in \delta^+(i)} \alpha_{ijp}$, i.e., α_{ip} is the sum of all the outgoing edges of a customer i. Hence,

$$\left\lfloor \frac{1}{k} \sum_{i \in S} \alpha_{ip} \right\rfloor = \left\lfloor \frac{1}{k} \sum_{i \in S} \sum_{(i,j) \in \delta^+(i)} \alpha_{ijp} \right\rfloor$$

which is only 1 or larger when k or more customers of S are visited on route p.

Proposition 1. *The SR inequalities* (21) *are valid for the Set Packing structure X.*

Proof. The proof follows directly from Chavtal-Gomory's procedure to construct valid inequalities (Wolsey [37]). Scale the $|S|$ inequalities $\sum_{p\in P}\alpha_{ip}\lambda_p \leq 1$ for each row $i \in S \subseteq M$ from (20) with $\frac{1}{k} \geq 0$ and add them:

$$\sum_{p\in P}\frac{1}{k}\sum_{i\in S}\alpha_{ip}\lambda_p \leq \frac{|S|}{k}$$

Flooring on left side and right side leads to (21). □

Observe that, when the coefficient $\left\lfloor \frac{1}{k}\sum_{i\in S}\alpha_{ip}\right\rfloor$ evaluates to 0 or 1 for all $p \in P$ and the right hand side $\left\lfloor \frac{|S|}{k}\right\rfloor = 1$ then the set of SR inequalities (21) is a subset of the clique inequalities (18).

From Definition 1 it is clear that the SR inequalities are Chvatal-Gomory rank-1 cuts, see Chvatal [6]. Eisenbrand [15] has shown that the separation problem is $\mathcal{NP}$-complete for general Chvatal-Gomory rank-1 cuts. However, in some special cases polynomial time separation is possible, e.g. the maximally violated mod-k cuts for a fixed k by Caprara et al. [4]. Since the SR inequalities are another special case, the separation problem will be investigated further.

3.1 Separation of Subset-Row Inequalities

The separation problem of SR inequalities is defined as follows: Given the current LP-solution λ where $\lambda_p < 1$ for all $p \in P$, and let n be the size of S. For some fixed values n and k where $1 < k \leq n$, find the most violated SR inequality. Using the binary variable x_i to denote whether $i \in S$ this can be stated as:

$$\max \sum_{p\in P}\left\lfloor \frac{1}{k}\sum_{i\in M} a_{ip}x_i\right\rfloor \lambda_p - \left\lfloor \frac{n}{k}\right\rfloor \tag{22}$$

$$\text{s.t.} \sum_{i\in M} x_i = n \tag{23}$$

$$x_i \in \{0,1\} \qquad \forall i \in M \tag{24}$$

The corresponding decision problem SR-DECISION asks whether

$$\sum_{p\in P}\left\lfloor \frac{1}{k}\sum_{i\in M} a_{ip}x_i\right\rfloor \lambda_p \geq c \tag{25}$$

is feasible subject to (23) and (24), where $1 \leq c < n$ and $c \in \mathbb{Z}$. Since we may multiply (25) by any coefficient $\frac{1}{\gamma} > 0$, the coefficient bounds $\lambda_p < 1$ and $c < n$ can be softened to

$$\lambda_p < \frac{1}{\gamma}, \qquad c < \frac{n}{\gamma} \tag{26}$$

This leads to the following proposition:

Proposition 2. *The separation problem SR-DECISION is $\mathcal{NP}$-complete.*

Example 2

Illustration of the transformation 3CNF-SAT to SR-DECISION. Given the 3CNF-SAT expression

$$\phi = (x_1 \vee \neg x_1 \vee \neg x_2) \wedge (x_3 \vee x_2 \vee x_4) \wedge (\neg x_1 \vee \neg x_3 \vee \neg x_4)$$

the matrix $A = (a_{ij})$ becomes

		1	$\dots$	m	$m+1$	$\dots$	$\dots$	$m+n$	$m+n+1$
		C_1	$\dots$	C_m	x_1	$\dots$	$\dots$	x_n	
1	x_1	1			1				
2	$\neg x_1$	1		1	1				
	x_2		1			1			
$\vdots$	$\neg x_2$	1				1			
	x_3		1				1		
	$\neg x_3$			1			1		
	x_4		1					1	
$2n$	$\neg x_4$			1				1	
$2n+1$									1
$2n+2$		1	1	1	1	1	1	1	1
$2n+3$		1	1	1	1	1	1	1	1

while we set $k = 3$, $\lambda_p = 1$ for $p \in P$ and $c = 8$.

Proof. We will show the statement by reduction from 3-conjunctive normal form satisfiability (3CNF-SAT). Given an expression ϕ written in three-conjunctive normal form, the 3CNF-SAT problem asks whether there is an assignment of binary values to the variables such that ϕ evaluates to true. An expression is in three-conjunctive normal form when it consists of a collection of disjunctive clauses $C_1, \dots, C_m$ of literals, where a literal is a variable x_i or a negated variable $\neg x_i$, and each clause contains exactly three literals.

Let $x_1, \dots, x_n$ be the set of variables which occurs in the clause ϕ. We transform the 3CNF-SAT instance to a SR-DECISION instance by constructing a matrix $A = (a_{ij})$ with $2n+3$ rows and $m+n+1$ columns, i.e., $M = \{1, \dots, 2n+3\}$ and $P = \{1, \dots, m+n+1\}$.

The rows $1, \dots, 2n$ of matrix A corresponds to literals $x_1, \neg x_1, x_2, \neg x_2, \dots, x_n, \neg x_n$, while columns $j = 1, \dots, m$ correspond to clauses $C_1, \dots, C_m$, and columns $j = m+1, \dots, m+n$ correspond to variables $x_1, \dots, x_n$.

We now define matrix A as follows: For $j = 1, \dots, m$ let $a_{ij} = 1$ iff the corresponding literal appears in clause C_j. For $j = 1, \dots, n$ let $a_{i,j+m} = 1$ iff the corresponding literal is x_j or $\neg x_j$. For $j = m+n+1$ let $a_{ij} = 0$. The last three rows of A are defined as follows: For $j = 1, \dots, m+n$ let $a_{2n+1,j} = 0$, while $a_{2n+1,m+n+1} = 1$. For $j = 1, \dots, m+n+1$ let $a_{2n+2,j} = a_{2n+3,j} = 1$. Finally we set $k = 3$, $\lambda_p = 1$ for all $p \in P$ and $c = m+n+1$. Note that all coefficients are within the bounds (26) for γ sufficiently large. An example of the transformation is illustrated in Example 2.

With the chosen constants, the SR-DECISION problem (25) reads

$$\sum_{p \in P} \left\lfloor \frac{1}{3} \sum_{i \in M} a_{ip} x_i \right\rfloor \geq m+n+1 = |P|$$

which is satisfied if and only if

$$\sum_{i \in M} a_{ip} x_i \geq 3 \qquad \forall p \in P$$

As the last three rows of A always must be chosen, it is equivalent to

$$\sum_{i=1}^{2n} a_{ip} x_i \geq 1 \qquad \forall p = 1, \ldots, m+n$$

(i) Assume that there is a feasible assignment of binary values to $x_1, \ldots, x_n$ such that ϕ evaluates to true in the 3CNF-SAT instance. In the corresponding SR-DECISION problem choose row i if and only if the corresponding literal is true in ϕ. Since exactly n literals are true, we will in this way choose n rows. Since at least one literal is true in each clause, and each column $1, \ldots, m$ corresponds to a clause in A we will get a contribution of at least one in each of these columns. Moreover, since exactly one of x_i and $\neg x_i$ is true in ϕ we will get a contribution of exactly one in column $m+1, \ldots, m+n$. Hence, the corresponding SR-DECISION problem is true.

(ii) Assume on the other hand that SR-DECISION is true. Let $P' \subseteq P$ be the set of rows corresponding to the solution. By assumption $|P'| = n$. First we notice that exactly one of the rows corresponding to the literals x_i and $\neg x_i$ is chosen. This follows from the fact that we have n columns $m+1, \ldots, m+n$ which needs to be covered by n rows, and each row covers exactly one column. For each literal in ϕ let x_i or $\neg x_i$ be true if the corresponding row was chosen in SR-DECISION. Each variable will be well-defined due to the above argument. Moreover, since the rows P' must cover at least one $a_{pi} = 1$ for each column $j = 1, \ldots, m$, we see that each clause in ϕ becomes true.

Since the reduction is polynomial, and SR-DECISION obviously is in $\mathcal{NP}$, we have proved the statement. □

Example 3 shows that typical separation problems of SR inequalities actually possess the properties assumed in the $\mathcal{NP}$-completeness proof.

4 Label-Setting Algorithm

When solving the pricing problem, it is noted that finding a route with negative reduced cost corresponds to finding a negative cost path starting and ending at the depot, i.e., an ESPPRC. Our ESPPRC algorithm is based on standard label setting techniques presented by e.g. Beasley and Christofides [2], Dumitrescu [14], Feillet et al. [16], Chabrier [5], Danna and Le Pape [9]; hence in the following we mainly focus on the dominance criterion used for handling the modifications stemming from the SR inequalities of the master problem.

The ESPPRC can be formally defined as: Given a weighted directed graph $G(V, E)$ with nodes V and edges E, and a set of resources R. For each edge $(i, j) \in E$ and resource $r \in R$ three parameters are given: A lower limit $a_r(i, j)$ on the accumulation of resource r when traversing edge $(i, j) \in E$; an upper limit $b_r(i, j)$ on the accumulation of resource r when traversing edge $(i, j) \in E$; and finally an amount $c_r(i, j)$ of resource r consumed by traversing edge $(i, j) \in E$. The objective is to find a minimum cost path p from a source node $o \in V$ to

Example 3

To illustrate that the bounds (26) indeed are realistic consider the case $k = 3$. Choose $\gamma = \frac{m+n+1}{\beta}$ where $\beta = \frac{n-2}{3}$ or $\beta = \frac{n-1}{3}$ depending on which of the expressions that evaluates to an integral value. The right hand side of (25) evaluates to

$$c \cdot \frac{1}{\gamma} = (m+n+1) \cdot \frac{\beta}{m+n+1} = \beta$$

where an integral value of β gives

$$\beta = \left\lfloor \frac{n}{3} \right\rfloor < n$$

The value of λ gives

$$\lambda_p \cdot \frac{1}{\gamma} = 1 \cdot \frac{\beta}{m+n+1} \leq 1 \qquad \forall p \in P$$

Hence all bounds are valid according to the separation problem (22)-(24).

---

a target node $o' \in V$, where the accumulated resources of p satisfy the limits for all resources $r \in R$. Without loss of generality, we assume that the limits must be satisfied at the start of each edge (i, j), i.e., before $c_r(i, j)$ has been consumed.

Remark that equivalent upper and lower limits and consumptions on the nodes can be "pushed" onto the edges, e.g., the ingoing edges of the node.

For the pricing problem of the VRPTW, the resources are demand d, time t, a binary visit-counter for each customer $v \in C$ and reduced cost $\overline{c}$. Note that also the reduced cost is considered a resource. When considering the pricing problem of the VRPTW, the consumptions and upper and lower limits of the resources at each edge (i, j) in ESPPRC are:

$$\begin{array}{llllr}
a_d(i,j) = 0, & b_d(i,j) = D - d_j, & c_d(i,j) = d_j & & \forall(i,j) \in E \\
a_t(i,j) = a_i, & b_t(i,j) = b_i, & c_t(i,j) = \tau_{ij} & & \forall(i,j) \in E \\
a_v(i,j) = 0, & b_v(i,j) = 1, & c_v(i,j) = 1 & \forall v \in V : v = j, & \forall(i,j) \in E \\
a_v(i,j) = 0, & b_v(i,j) = 1, & c_v(i,j) = 0 & \forall v \in V : v \neq j, & \forall(i,j) \in E \\
a_{\overline{c}}(i,j) = -\infty, & b_{\overline{c}}(i,j) = \infty\ , & c_{\overline{c}}(i,j) = \overline{c}_{ij} & & \forall(i,j) \in E
\end{array}$$

In the label-setting algorithm labels at node v represent partial paths from o to v. The following attributes for a label L are considered:

$\overline{v}(L)$	The current end-node of the partial path represented by L.
$\overline{c}(L)$	The sum of the reduced cost along path L.
$r(L)$	The accumulated consumption of resource $r \in R$ along path L.

A feasible extension $\epsilon \in \mathcal{E}(L)$ of a label L is a partial path starting in a node $\overline{v}(L) \in V$ and ending in the target node o', that does not violate any resources when concatenated with the partial path represented by L.

In the following it is assumed that all resources are bounded strongly from above, and weakly from below. This means that if the current resource accumulation of a label is below the lower limit on a given edge, it is allowed to fill up the resource to the lower limit, e.g., waiting for a time window to open. This means that two consecutive labels L_u and L_v related by an edge (u, v), i.e., L_u is extended and creates L_v, where $\overline{v}(L_u) = u$ and $\overline{v}(L_v) = v$, must satisfy

$$r(L_v) \leq b_r(u,v), \qquad \forall r \in R \tag{27}$$
$$r(L_v) = \max\{r(L_u) + c_r(u,v), a_r(u,v)\}, \qquad \forall r \in R \tag{28}$$

Here (27) demands that each label L_u satisfies the upper limit $b_r(u,v)$ of resource r corresponding to edge (u,v), while (28) states that resource r at label L_v corresponds to the resource consumption at label L_u plus the amount consumed by traversing edge (u,v), respecting the lower limit $a_r(u,v)$ on edge (u,v).

A simple enumeration algorithm could be used to produce all these labels, but to limit the number of labels considered, dominance rules are introduced to fathom labels which do not lead to an optimal solution.

Definition 2. *A label L_i dominates label L_j if*

$$\overline{v}(L_i) = \overline{v}(L_j) \tag{29}$$
$$\overline{c}(L_i) \leq \overline{c}(L_j) \tag{30}$$
$$\mathcal{E}(L_j) \subseteq \mathcal{E}(L_i) \tag{31}$$

In other words, the paths corresponding to labels L_i and L_j should end at the same node $\overline{v}(L_i) = \overline{v}(L_j) \in V$, the path corresponding to label L_i should cost no more than the path corresponding to label L_j, and finally any feasible extension of L_j is also a feasible extension of L_i.

Feillet et al. [16] suggested to consider the set of nodes that cannot be reached from a label L_i and compare the set with the unreachable nodes of a label L_j in order to determine if some extensions are impossible. Or in other words: update the node resources in an eager fashion instead of a lazy. The following definition is a generalization of Definition 3 in Feillet et al. [16].

Definition 3. *Given a start node $o \in V$, a label L, and a node $u \in V$ where $\overline{v}(L) = u$ a node $v \in V$ is considered* unreachable *if v has already been visited on the path from o to u or if a resource window is violated, e.g.:*

$$\exists r \in R \qquad r(L) + \ell_r(u,v) > b_r(v)$$

where $\ell_r(u,v)$ is a lower bound on the consumption of resource r on all feasible paths from u to v. The node resources *are then given as: $v(L) = 1$ indicates that node $v \in V$ is unreachable from node $\overline{v}(L) \in V$, and $v(L) = 0$ otherwise.*

Determining if (31) holds can be quite cumbersome because the straightforward definition demands that we calculate all extensions of the two labels. Therefore, a sufficient criterion for (31) is sought that can be computed faster. If label L_i has consumed less resources than label L_j then no resources are limiting the possibilities of extending L_i compared to L_j, hence the following proposition can be used as a relaxed version of the dominance criterion.

Proposition 3. Desaulniers et al. [10]. *If all resource extension functions are non-decreasing, then label L_i dominates label L_j if:*

$$\overline{v}(L_i) = \overline{v}(L_j) \tag{32}$$
$$\overline{c}(L_i) \leq \overline{c}(L_j) \tag{33}$$
$$r(L_i) \leq r(L_j) \qquad \forall r \in R \tag{34}$$

Using Proposition 3 as a dominance criterion is a relaxation of the dominance criterion of Definition 2 since only a subset of labels satisfying (29), (30) and (31) satisfies inequalities (32), (33) and (34).

4.1 Solving the Modified Pricing Problem

Consider some valid SR inequality of the form (21),

$$\sum_{p\in P}\left\lfloor \frac{1}{k}\sum_{i\in S}\alpha_{ip}\right\rfloor \lambda_p \leq \left\lfloor \frac{|S|}{k}\right\rfloor$$

where $S \subseteq M$ and $0 < k \leq |S|$. Let $\sigma \leq 0$ be the corresponding dual variable when solving the master problem to LP-optimality. From (17) the reduced cost of a column in the VRPTW master problem is:

$$\hat{c}_p = \overline{c}_p - \sigma\left\lfloor \frac{\sum_{i\in S}\sum_{(i,j)\in\delta^+(i)}\alpha_{ijp}}{k}\right\rfloor = \sum_{(i,j)\in E}\overline{c}_{ij}\alpha_{ijp} - \sigma\left\lfloor \frac{\sum_{i\in S}\sum_{(i,j)\in\delta^+(i)}\alpha_{ijp}}{k}\right\rfloor \tag{35}$$

We analyze how this additional cost can be handled in the label-setting algorithm for ESPPRC.

Let $V(L)$ be the nodes visited on the partial path of label L. The cost of a label L can then be expressed as:

$$\hat{c}(L) = \overline{c}(L) - \sigma\left\lfloor \frac{|S\cap V(L)|}{k}\right\rfloor \tag{36}$$

A new resource m can be used to compute the coefficient of penalty σ for label L, i.e., $m(L) = |S \cap V(L)|$, the number of customers involved in the cut. Note that the consumption of resource m is 1 for each e.g. outgoing edge of the involved customers. Therefore the usual dominance criterion of Proposition 3 can be used. Note that in case L_i dominates L_j, $\overline{c}(L_i) \leq \overline{c}(L_j)$ and $m(L_i) \leq m(L_j)$ so $\hat{c}(L_i) \leq \hat{c}(L_j)$ since $-\sigma > 0$. Hence the penalty term must only be considered on the last edge to the target node to compute the reduced cost $\hat{c}(L)$ of path L. However, further labels can be eliminated by exploiting the structure of (36).

For a label L let

$$\mathcal{T}(L) = |S\cap V(L)| \bmod k$$

be the number of visits made to S since the last penalty was paid for visiting k nodes in S. Recall $\mathcal{E}(L)$ as the set of feasible extensions from the label L to the target node o' and note that when label L_i dominates label L_j, their common extensions are $\mathcal{E}(L_j)$ due to (31). The following cost dominance criterion is obtained for a single SR inequality:

Proposition 4. *If $\mathcal{T}(L_i) \leq \mathcal{T}(L_j)$, $\overline{v}(L_i) = \overline{v}(L_j)$, $\hat{c}(L_i) \leq \hat{c}(L_j)$, and $r(L_i) \leq r(L_j)$ $\forall r \in R$, then label L_i dominates label L_j.*

Proof. Consider any common extension $\epsilon \in \mathcal{E}(L_j)$. Since $\mathcal{T}(L_i) \leq \mathcal{T}(L_j)$ the relation between the number of future penalties for the two labels when concatenated with ϵ is:

$$\left\lfloor \frac{|S\cap\epsilon| + \mathcal{T}(L_i)}{k}\right\rfloor \leq \left\lfloor \frac{|S\cap\epsilon| + \mathcal{T}(L_j)}{k}\right\rfloor$$

This leads to the following relation between the costs:

$$\begin{aligned} &\hat{c}(L_i + \epsilon) = \hat{c}(L_i) + \overline{c}(\epsilon) - \sigma \left\lfloor \frac{|S \cap \epsilon| + \mathcal{T}(L_i)}{k} \right\rfloor \\ \leq \quad &\hat{c}(L_j + \epsilon) = \hat{c}(L_j) + \overline{c}(\epsilon) - \sigma \left\lfloor \frac{|S \cap \epsilon| + \mathcal{T}(L_j)}{k} \right\rfloor \end{aligned}$$

Hence label L_i dominates label L_j. □

Proposition 5. *If $\mathcal{T}(L_i) > \mathcal{T}(L_j)$, $\overline{v}(L_i) = \overline{v}(L_j)$, $\hat{c}(L_i) - \sigma \leq \hat{c}(L_j)$, and $r(L_i) \leq r(L_j)$ $\forall r \in R$, then label L_i dominates label L_j.*

Proof. Consider any common extension $\epsilon \in \mathcal{E}(L_j)$. Since $\mathcal{T}(L_i) > \mathcal{T}(L_j)$ the relation between the number of future penalties for the two labels when concatenated with ϵ is:

$$\left\lfloor \frac{|S \cap \epsilon| + \mathcal{T}(L_i)}{k} \right\rfloor \geq \left\lfloor \frac{|S \cap \epsilon| + \mathcal{T}(L_j)}{k} \right\rfloor \tag{37}$$

Since $0 \leq \mathcal{T}(L_j) < \mathcal{T}(L_i) \leq k$ it is clear that the left hand side of (37) is at most one unit larger than the right hand side, i.e., label L_i will pay the penalty at most one more time than label L_j. Hence,

$$\left\lfloor \frac{|S \cap \epsilon| + \mathcal{T}(L_i)}{k} \right\rfloor - 1 \leq \left\lfloor \frac{|S \cap \epsilon| + \mathcal{T}(L_j)}{k} \right\rfloor$$

That is, the additional cost of extending L_i with ϵ is at most $-\sigma$ more than extending L_j with ϵ. This leads to the following relation between the costs:

$$\begin{aligned} \hat{c}(L_i + \epsilon) &= \hat{c}(L_i) + \overline{c}(\epsilon) - \sigma \left\lfloor \frac{|S \cap \epsilon| + \mathcal{T}(L_i)}{k} \right\rfloor \\ &= \hat{c}(L_i) - \sigma + \overline{c}(\epsilon) - \sigma \left(\left\lfloor \frac{|S \cap \epsilon| + \mathcal{T}(L_i)}{k} \right\rfloor - 1 \right) \\ &\leq \hat{c}(L_j) + \overline{c}(\epsilon) - \sigma \left\lfloor \frac{|S \cap \epsilon| + \mathcal{T}(L_j)}{k} \right\rfloor \\ &= \hat{c}(L_j + \epsilon) \end{aligned}$$

Hence label L_i dominates label L_j. □

Observe that if $\mathcal{T}(L_i) + |S \cap \epsilon| < k$ for all $\epsilon \in \mathcal{E}(L_j)$, it is not possible to visit S enough times to trigger a penalty, i.e., the temporary penalty to the cost of L_i can be disregarded.

In case of several SR inequalities, the new dominance criterion is as follows:

Proposition 6. *Let $Q = \{q : \sigma_q < 0 \wedge \mathcal{T}_q(L_i) > \mathcal{T}_q(L_j)\}$. Then label L_i dominates label L_j if:*

$$\overline{v}(L_i) = \overline{v}(L_j) \tag{38}$$

$$\hat{c}(L_i) - \sum_{q \in Q} \sigma_q \leq \hat{c}(L_j) \tag{39}$$

$$r(L_i) \leq r(L_j) \qquad \forall r \in R \tag{40}$$

Proof. The validity of (39) follows directly from Propositions 4 and 5. The validity of (38) and (40) follows from Proposition 3. □

5 Computational Results

The BCP algorithm has been implemented using the BCP framework and the open source linear programming solver CLP, both parts of the framework COIN [7]. All tests are run on an Intel® Pentium® 4 3.0 GHz PC with 4 GB of memory.

The benchmarks of Solomon [34] follow a naming convention of `DTm.n`. The distribution `D` can be R, C and RC, where the C instances have a clustered distribution of customers, the R instances have a random distribution of customers, and the RC instances are a mix of clustered and randomly distributed customers. The time window `T` is either 1 or 2, where instances of type 1 have tighter time windows than instances of type 2. The instance number is given by `m` and the number of customers is given by `n`.

The outline of the BCP algorithm presented in this paper is as follows:

Step 1. Choose an unprocessed branch node. If the lower bound is above the upper bound, then fathom branch node.

Step 2. Solve the LP master problem.

Step 3. Solve the pricing problem heuristically. If columns with negative reduced cost have been found, then add them to the master problem and go back to Step 2.

Step 4. Solve the pricing problem to optimality. Update the lower bound. If the lower bound is above the upper bound, then fathom the branch node. If some new columns have been found, then add them to the master problem and go to Step 2.

Step 5. Separate SR inequalities. If any violated cuts are found, then add them to the master problem and go to Step 2.

Step 6. If the LP solution is fractional then branch and add the children to the set of unprocessed branch nodes. Mark the current node as processed and go to Step 1.

We allow a maximum of 400 variables and 50 cuts to be generated in each of steps 3, 4, and 5 respectively. The pricing-problem heuristic is based on the label-setting algorithm but a simpler heuristic dominance criterion is used. If a label L_i dominates L_j on *cost*, *demand* and *time* it is regarded as dominated and L_j is discarded. That is, no concern is taken to the *node* resources. The separation of SR inequalities is done with a complete enumeration of all inequalities with $|S| = 3$ and $k = 2$. Let B be the set of basic variables in the current LP solution and C be the set of customers, then the separation can be done in $O(|C|^3|B|)$. Preliminary tests showed that SR inequalities with different values of n and k seldom appeared in the VRPTW instances, hence no separation of these inequalities was done.

The branch tree is explored with a best-bound search strategy, i.e., the node with the lowest lower bound is chosen first, breaking ties based on the LP result of the strong branching. We have adapted the branching rule used by Fukasawa et al. [17]: For a subset of customers $S \subset C$ the number of vehicles to visit that set is either two or greater than or equal to four, i.e.,

$$\sum_{k \in K} \sum_{(i,j) \in \delta^+(S)} (x_{ijk} + x_{jik}) = 2$$

and

$$\sum_{k \in K} \sum_{(i,j) \in \delta^+(S)} (x_{ijk} + x_{jik}) \geq 4$$

We are using the cut library of Lysgaard [25] to separate candidate sets for branching, which is an implementation of the heuristic methods described in Lysgaard et al. [26].

Author(s)	CPU	SpecINT	SpecCFP	Normalized
Irnich and Villeneuve [20]	P3 600 MHz*	295	204	0.23
Chabrier [5]	P4 1.5 GHz	526	606	0.52
Jepsen et al. [this paper]	P4 3.0 GHz	1099	1077	1.00

Table 1: Comparison of computer speed. Based on CPU2000 benchmarks from SPEC [35]. (*) benchmarks are given for P3 650 MHz since no benchmarks were available for P3 600. The normalized value is an average of SpecINT and SpecCFP.

5.1 Running Times

To give a fair comparison between running times of our algorithm and the two most recent algorithms presented by Irnich and Villeneuve [20] and Chabrier [5], the CPU speed is taken into account. This is done according to the CPU2000 benchmarks reported by The Standard Performance Evaluation Corporation SPEC [35]. Table 1 gives the integer and floating point benchmark scores and a normalized value, e.g. our computations were carried out on a computer approximately twice as fast as that of Chabrier.

A comparison of running times is shown in Table 2. To save space we only report results on what we consider hard instances, i.e., the Solomon instances that were closed by either Irnich and Villeneuve [20] or Chabrier [5] and by us.

Our algorithm outperforms those of Irnich and Villeneuve and Chabrier for 17 out of 22 instances. Seven of these instances were solved without any SR inequalities. In these cases, the faster running times were probably due to the bi-directional label-setting algorithm.

With the introduction of SR inequalities our algorithm becomes competitive with the algorithm based on solving k-cyc-SPPRC (e.g. instances R104.100, RC104.100, RC107.100, RC108.100, and R211.50) and clearly outperforms the ESPPRC based algorithm on the harder instances (e.g., instances R210.50, RC202.100, RC205.100, and RC208.25). In some cases when solving the C1 and C2 instances the BCP algorithm tails off leading to slow solution times or no solution at all. However, this must be seen in the light of a simple implementation and no use of other cutting planes than the SR inequalities.

5.2 Comparing Lower Bounds in the Root Node

Table 3 reports the lower bounds obtained in the root node of the master problem with and without SR inequalities and with best bounds obtained by Irnich and Villeneuve [20] using k-cyc-SPPRC. Again we only report results on what we consider the hard instances from Table 2 plus the instances closed by us.

As seen, the lower bounds obtained with SR inequalities are improved quite significantly for most of the instances. Moreover, in most cases the problems are solved without branching. Out of the 32 instances considered, the gap was closed in the root node in 8 instances due to the ESPPRC and in an additional 16 instances due to the SR inequalities. However, one needs to take into account that the running time of solving the root node is increased due to the increased difficulty of the pricing problems.

	Irnich and Villeneuve [20]	Chabrier [5]	Jepsen et al. [this paper]			
Instance	Time (s)	Time (s)	Time (s)	Speedup		
R104.100	268106.0	-	**32343.9**	1.9	/	-
RC104.100	986809.0	-	**65806.8**	3.4	/	-
RC107.100	42770.7	-	**153.8**	64.0	/	-
RC108.100	71263.0	-	**3365.0**	4.9	/	-
R203.50	**217.1**	3320.9	50.8	1.0	/	34.0
R204.25	123.1	171.6	**7.5**	3.8	/	11.9
R205.50	585.7	531.0	**15.5**	8.6	/	17.8
R206.50	22455.3	4656.1	**190.9**	27.1	/	12.7
R208.25	321.9	741.5	***2.9**	25.5	/	133.0
R209.50	142.4	195.4	**16.6**	2.0	/	6.1
R210.50	11551.4	65638.6	***332.7**	8.0	/	102.6
R211.50	**21323.0**	-	10543.8	0.5	/	-
RC202.50	241.6	**13.0**	*10.7	5.2	/	0.6
RC202.100	124018.0	19636.5	**312.6**	91.2	/	32.7
RC203.25	1876.0	5.1	***0.7**	616.4	/	3.8
RC203.50	54229.2	4481.5	***190.9**	65.3	/	12.2
RC204.25	-	13.0	***2.0**	-	/	3.4
RC205.50	52.6	**10.6**	*5.9	2.1	/	0.9
RC205.100	13295.9	15151.7	**221.2**	13.8	/	35.6
RC206.50	469.1	**9.4**	*8.2	13.2	/	0.6
RC207.50	-	71.1	***21.5**	-	/	1.7
RC208.25	-	33785.3	**78.4**	-	/	224.1

Table 2: Comparison of running time. Speedup is calculated based on the normalized values in Table 1 and are versus Irnich and Villeneuve and Chabrier respectively. Results with (*) are based on an algorithm without the SR inequalities. Results in **boldface** indicate the fastest algorithm after normalization. (-) indicates that no running times were provided by the author(s) or that the instance was not solved.

		Irnich and Villeneuve [20]		Jepsen et al. [this paper]	
Instance	UB	k	LB	LB(1)	LB(2)
R104.100	971.5	3	955.8	956.9	971.3
R108.100	932.1	4	913.9	913.6	**932.1**
R112.100	948.6	3	925.9	926.8	946.7
RC104.100	1132.3	3	1114.4	1101.9	1129.9
RC106.100	1372.7	4	1343.1	1318.8	1367.3
RC107.100	1207.8	4	1195.4	1183.4	**1207.8**
RC108.100	1114.2	3	1100.5	1073.5	**1114.2**
R202.100	1029.6	0	933.5	1022.3	1027.3
R203.50	605.3	4	598.6	598.6	**605.3**
R203.100	870.8	2	847.1	867.0	**870.8**
R204.25	355.0	4	349.1	350.5	**355.0**
R205.50	690.1	4	682.8	682.9	**690.1**
R206.50	632.4	4	621.3	626.4	**632.4**
R207.50	575.5	4	557.4	564.1	**575.5**
R208.25	328.2	4	327.1	**328.2**	**328.2**
R209.50	600.6	4	599.9	599.9	**600.6**
R209.100	854.8	3	834.4	841.5	854.4
R210.50	645.6	4	633.1	636.1	645.3
R211.50	535.5	4	526.0	528.7	**535.5**
RC202.50	613.6	4	604.5	**613.6**	**613.6**
RC202.100	1092.3	3	1055.0	1088.1	**1092.3**
RC203.25	326.9	4	297.7	**326.9**	**326.9**
RC203.50	555.3	4	530.0	**555.3**	**555.3**
RC203.100	923.7	0	693.7	922.6	**923.7**
RC204.25	299.7	4	266.3	**299.7**	**299.7**
RC205.50	630.2	4	**630.2**	**630.2**	**630.2**
RC205.100	1154.0	3	1130.5	1147.7	**1154.0**
RC206.50	610.0	4	597.1	**610.0**	**610.0**
RC206.100	1051.1	3	1017.0	1038.6	**1051.1**
RC207.50	558.6	4	504.9	**558.6**	**558.6**
RC208.25	269.1	4	238.3	**269.1**	**269.1**
RC208.50	476.7	3	422.3	472.3	**476.7**

Table 3: Comparison of root lower bounds. LB by Irnich and Villeneuve is the best lower bound obtained with k-cyc-SPPRC and valid inequalities, LB(1) is with ESPPRC and LB(2) is with ESPPRC and SR inequalities. Lower bounds in **boldface** indicate lower bounds equal to the upper bound. Instances in **boldface** are the Solomon instances closed by us.

		25 customers		50 customers		100 customers	
Class	No.	Prev.	Jepsen et al. [this paper]	Prev.	Jepsen et al. [this paper]	Prev	Jepsen et al. [this paper]
R1	12	12	12	12	12	10	12
C1	9	9	9	9	9	9	9
RC1	8	8	8	8	8	8	8
R2	11	11	11	9	9	1	4
C2	8	8	8	8	7	8	7
RC2	8	8	8	8	7	3	5
Summary	56	56	56	55	52	39	45

Table 4: Summary of solved Solomon instances. No. is the number of instances in that class, and for 25, 50 and 100 customers the two columns refers to the number of instances previously solved to optimality and the number of instances solved to optimality by us.

Instance	UB	LB	Vehicles	Tree	LP	$\text{Time}_{\text{root}}$(s)	Time_{var}(s)	Time_{LP}(s)	Time (s)
R108.100	932.1	932.1	10	1	132	5911.71	5796.04	77.36	5911.74
R112.100	948.6	946.7	10	9	351	55573.68	199907.03	1598.63	202803.94
R202.100	1029.6	1027.3	8	13	514	974.51	730.04	4810.47	8282.38
R203.100	870.8	870.8	6	1	447	54187.15	48474.45	3973.42	54187.40
R207.50	575.5	575.5	3	1	107	34406.92	34282.47	118.69	34406.96
R209.100	854.8	854.4	5	3	337	31547.45	74779.58	2978.42	78560.47
RC203.100	923.7	923.7	5	1	402	14917.18	13873.53	1025.65	14917.36
RC206.100	1051.1	1051.1	7	1	179	339.63	159.33	171.34	339.69

Table 5: Instances closed by Jepsen et al. [this paper]. *UB* is the optimal solution found by us, *LB* is lower bound at the root node, *Vehicles* is the number of vehicles in the solution, *Tree* is the number of branch nodes, *LP* is the number of LP iterations, $Time_{root}$ is the time solving the root node, $Time_{var}$ is time spent solving the pricing problem, $Time_{LP}$ is the time spent solving LP problems, and *Time* is the total time.

5.3 Closed Solomon Instances

Table 4 gives an overview of how many instances were solved for each class of the Solomon instances. We were able to close 8 previously unsolved instances. We did not succeed to solve four previously solved instances (R204.50, C204.50, C204.100, and RC204.50).

Information on all solved Solomon instances can be found in Tables 6–8 in Appendix A. Furthermore Table 5 provides detailed information of the instances closed in this paper. The solutions can be found in Tables 9–16 in Appendix B.

6 Concluding Remarks

The introduction of the SR inequalities significantly improved the results of the BCP algorithm. This made it possible to solve 8 previously unsolved instances from the Solomon benchmarks.

Except for four cases (R204.50, C204.50 and C204.100 solved with k-cyc-SPPRC by Irnich and Villeneuve [20] and RC204.50 solved by Danna and Le Pape [9]) our BCP algorithm is competitive and in most cases superior to earlier algorithms within this field. With minor modifications in the definition of the conflict graph the SR inequalities can be applied to the k-cyc-SPPRC algorithm using the same cost-modified dominance criterion as described in this paper. Preliminary results by Jepsen et al. [21] have shown that the lower bounds obtained in a BCP algorithm for VRPTW using the k-cyc-SPPRC algorithm and SR inequalities are almost as good as those obtained using the approach presented in this paper. This seems to be a promising direction of research in order to solve large VRPTW instances, since the ESPPRC algorithm is considerably slower than the k-cyc-SPPRC algorithm when the number of customers increases.

Moreover, we note that the SR inequalities can be applied to any set packing problem. That is, they can be used in BCP algorithms for other problems with a set packing problem master problem. One only needs to consider how the dual variables of the SR inequalities are handled in the pricing problems, however this is not necessarily trivial and must be investigated for the individual pricing problems.

Adding SR inequalities to the master problem means that the pricing problem becomes a shortest path problem with non-additive non-decreasing constraints or objective function. By modifying the dominance criterion, we have shown that this is tractable in a label-setting algorithm. A further discussion of shortest path problems with various non-additive constraints can be found in Pisinger and Reinhardt [30]. The development of algorithms which efficiently handle non-additive constraints is important to increase the number of valid inequalities which can be handled.

A Results on Solomon Instances

This appendix contains detailed information about solved Solomon instances. The first column of the tables is the instance name, then three columns for the branch-and-cut-and-price algorithm with ESPPRC and with ESPPRC and SR-inequalities follow. The columns are the lower bound in the root node, the number of branch tree nodes and the total running time. A (-) means that the instance was not solved. The last two columns are the optimal upper bound and a reference to the authors who were the first to solve that instance, disregarding Desrochers et al. [12] who solved many of the instances with a different calculation of the travel times making it hard to compare with later solutions. The author legend is:

C:	Chabrier [5]
CR:	Cook and Rich [8]
DLP:	Danna and Le Pape [9]
IV:	Irnich and Villeneuve [20]
JPSP:	Jepsen et al. [this paper]
KDMSS:	Kohl et al. [23]
KLM:	Kallehauge et al. [22]
L:	Larsen [24]
S:	Salani [33]

	with ESPPRC			with ESPPRC and SR				
Instance	LB	Tree	Time (s)	LB	Tree	Time (s)	UB	Ref.
R101	617.1	1	0.02	617.1	1	0.02	617.1	KDMSS
R102	546.4	3	0.13	547.1	1	0.09	547.1	KDMSS
R103	454.6	1	0.11	454.6	1	0.11	454.6	KDMSS
R104	416.9	1	0.12	416.9	1	0.12	416.9	KDMSS
R105	530.5	1	0.02	530.5	1	0.02	530.5	KDMSS
R106	457.3	5	0.29	465.4	1	0.10	465.4	KDMSS
R107	424.3	1	0.12	424.3	1	0.12	424.3	KDMSS
R108	396.9	3	0.31	397.3	1	0.24	397.3	KDMSS
R109	441.3	1	0.06	441.3	1	0.06	441.3	KDMSS
R110	438.4	17	1.16	444.1	3	0.29	444.1	KDMSS
R111	427.3	3	0.23	428.8	1	0.13	428.8	KDMSS
R112	387.1	13	1.19	393.0	1	0.52	393.0	KDMSS
C101	191.3	1	0.13	191.3	1	0.13	191.3	KDMSS
C102	190.3	1	0.53	190.3	1	0.53	190.3	KDMSS
C103	190.3	1	0.80	190.3	1	0.80	190.3	KDMSS
C104	186.9	1	3.29	186.9	1	3.29	186.9	KDMSS
C105	191.3	1	0.17	191.3	1	0.17	191.3	KDMSS
C106	191.3	1	0.14	191.3	1	0.14	191.3	KDMSS
C107	191.3	1	0.20	191.3	1	0.20	191.3	KDMSS
C108	191.3	1	0.37	191.3	1	0.37	191.3	KDMSS
C109	191.3	1	0.62	191.3	1	0.62	191.3	KDMSS
RC101	406.7	5	0.20	461.1	1	0.09	461.1	KDMSS
RC102	351.8	1	0.05	351.8	1	0.05	351.8	KDMSS
RC103	332.8	1	0.19	332.8	1	0.19	332.8	KDMSS
RC104	306.6	1	0.52	306.6	1	0.52	306.6	KDMSS
RC105	411.3	1	0.06	411.3	1	0.06	411.3	KDMSS
RC106	345.5	1	0.10	345.5	1	0.10	345.5	KDMSS
RC107	298.3	1	0.29	298.3	1	0.29	298.3	KDMSS
RC108	294.5	1	0.67	294.5	1	0.67	294.5	KDMSS
R201	460.1	3	0.44	463.3	1	0.27	463.3	CR+KLM
R202	410.5	1	0.61	410.5	1	0.61	410.5	CR+KLM
R203	391.4	1	0.80	391.4	1	0.80	391.4	CR+KLM
R204	350.5	19	18.40	355.0	1	7.51	355.0	IV+C
R205	390.6	3	1.62	393.0	1	1.06	393.0	CR+KLM
R206	373.6	3	1.67	374.4	1	0.93	374.4	CR+KLM
R207	360.1	5	4.03	361.6	1	1.39	361.6	KLM
R208	328.2	1	2.87	328.2	1	2.87	328.2	IV+C
R209	364.1	9	4.99	370.7	1	2.26	370.7	KLM
R210	404.2	3	1.52	404.6	1	1.04	404.6	CR+KLM
R211	341.4	29	38.17	350.9	1	22.62	350.9	KLM
C201	214.7	1	0.84	214.7	1	0.84	214.7	CR+L
C202	214.7	1	3.00	214.7	1	3.00	214.7	CR+L
C203	214.7	1	3.02	214.7	1	3.02	214.7	CR+L
C204	213.1	1	7.00	213.1	1	7.00	213.1	CR+KLM
C205	214.7	1	1.10	214.7	1	1.10	214.7	CR+L
C206	214.7	1	1.75	214.7	1	1.75	214.7	CR+L
C207	214.5	1	2.70	214.5	1	2.70	214.5	CR+L
C208	214.5	1	1.85	214.5	1	1.85	214.5	CR+L
RC201	360.2	1	0.25	360.2	1	0.25	360.2	CR+L
RC202	338.0	1	0.58	338.0	1	0.58	338.0	CR+KLM
RC203	326.9	1	0.72	326.9	1	0.72	326.9	IV+C
RC204	299.7	1	1.95	299.7	1	1.95	299.7	C
RC205	338.0	1	0.62	338.0	1	0.62	338.0	L+KLM
RC206	324.0	1	0.87	324.0	1	0.87	324.0	KLM
RC207	298.3	1	0.88	298.3	1	0.88	298.3	KLM
RC208	269.1	1	78.42	269.1	1	78.42	269.1	C

Table 6: Instances with 25 customers.

	with ESPPRC			with ESPPRC and SR				
Instance	LB	Tree	Time (s)	LB	Tree	Time (s)	UB	Ref.
R101	1043.4	3	0.14	1044.0	1	0.09	1044.0	KDMSS
R102	909.0	1	0.27	909.0	1	0.27	909.0	KDMSS
R103	769.3	13	4.98	772.9	1	2.02	772.9	KDMSS
R104	619.1	21	33.29	625.4	1	6.73	625.4	KDMSS
R105	892.2	29	2.78	893.7	5	1.15	899.3	KDMSS
R106	791.4	5	1.41	793.0	1	0.83	793.0	KDMSS
R107	707.3	11	5.56	711.1	1	4.76	711.1	KDMSS
R108	594.7	789	1723.29	607.4	23	1601.68	617.7	CR+KLM
R109	775.4	77	20.11	783.3	7	11.54	786.8	KDMSS
R110	695.1	9	3.38	697.0	1	1.46	697.0	KDMSS
R111	696.3	41	19.21	707.2	1	3.67	707.2	CR+KLM
R112	614.9	165	169.26	630.2	1	35.67	630.2	CR+KLM
C101	362.4	1	0.47	362.4	1	0.47	362.4	KDMSS
C102	361.4	1	1.59	361.4	1	1.59	361.4	KDMSS
C103	361.4	1	6.06	361.4	1	6.06	361.4	KDMSS
C104	358.0	1	1564.88	358.0	1	1564.88	358.0	KDMSS
C105	362.4	1	0.49	362.4	1	0.49	362.4	KDMSS
C106	362.4	1	0.69	362.4	1	0.69	362.4	KDMSS
C107	362.4	1	0.97	362.4	1	0.97	362.4	KDMSS
C108	362.4	1	1.55	362.4	1	1.55	362.4	KDMSS
C109	362.4	1	3.62	362.4	1	3.62	362.4	KDMSS
RC101	850.1	39	5.60	944.0	1	2.12	944.0	KDMSS
RC102	721.9	127	60.41	822.5	1	8.68	822.5	KDMSS
RC103	645.3	9	8.56	710.9	1	40.05	710.9	KDMSS
RC104	545.8	1	5.71	545.8	1	5.71	545.8	KDMSS
RC105	761.6	21	7.22	855.3	1	4.31	855.3	KDMSS
RC106	664.5	11	3.35	723.2	1	3.88	723.2	KDMSS
RC107	603.6	7	4.60	642.7	1	4.49	642.7	KDMSS
RC108	541.2	5	15.88	594.8	5	260.95	598.1	KDMSS
R201	791.9	1	4.97	791.9	1	4.97	791.9	CR+KLM
R202	698.5	1	9.88	698.5	1	9.88	698.5	CR+KLM
R203	598.6	25	355.99	605.3	1	50.80	605.3	IV+C
R204			-			-	506.4	IV
R205	682.9	35	118.12	690.1	1	15.45	690.1	IV+C
R206	626.4	47	288.00	632.4	1	190.86	632.4	IV+C
R207	564.1	141	15400.44	575.5	1	34406.96	575.5	**JPSP**
R208			-			-	-	-
R209	599.9	3	24.45	600.6	1	16.63	600.6	IV+C
R210	636.1	49	332.70	645.3	3	18545.61	645.6	IV+C
R211	528.7	31	44644.89	535.5	1	10543.81	535.5	IV+DLP
C201	360.2	1	42.07	360.2	1	42.07	360.2	CR+L
C202	360.2	1	67.05	360.2	1	67.05	360.2	CR+KLM
C203	359.8	1	214.88	359.8	1	214.88	359.8	CR+KLM
C204			-			-	350.1	KLM
C205	359.8	1	64.18	359.8	1	64.18	359.8	CR+KLM
C206	359.8	1	38.91	359.8	1	38.91	359.8	CR+KLM
C207	359.6	1	72.81	359.6	1	72.81	359.6	CR+KLM
C208	350.5	1	55.79	350.5	1	55.79	350.5	CR+KLM
RC201	684.8	1	3.00	684.8	1	3.00	684.8	L+KLM
RC202	613.6	1	10.69	613.6	1	10.69	613.6	IV+C
RC203	555.3	1	190.88	555.3	1	190.88	555.3	IV+C
RC204			-			-	442.2	DLP
RC205	630.2	1	5.88	630.2	1	5.88	630.2	IV+C
RC206	610.0	1	8.17	610.0	1	8.17	610.0	IV+C
RC207	558.6	1	21.53	558.6	1	21.53	558.6	C
RC208			-	476.7	1	1639.40	476.7	S

Table 7: Instances with 50 customers.

	with ESPPRC			with ESPPRC and SR				
Instance	LB	Tree	Time (s)	LB	Tree	Time (s)	UB	Ref.
R101	1631.2	57	20.08	1634.0	3	1.87	1637.7	KDMSS
R102	1466.6	1	4.39	1466.6	1	4.39	1466.6	KDMSS
R103	1206.8	19	55.78	1208.7	1	23.85	1208.7	CR+L
R104			-	971.3	3	32343.92	971.5	IV
R105	1346.2	113	126.96	1355.2	5	43.12	1355.3	KDMSS
R106	1227.0	147	511.07	1234.6	1	75.42	1234.6	CR+KLM
R107			-	1064.3	3	1310.30	1064.6	CR+KLM
R108			-	932.1	1	5911.74	932.1	**JPSP**
R109			-	1144.1	19	1432.41	1146.9	CR+KLM
R110			-	1068.0	3	1068.31	1068.0	CR+KLM
R111			-	1045.9	39	83931.48	1048.7	CR+KLM
R112			-	946.7	9	202803.94	948.6	**JPSP**
C101	827.3	1	3.02	827.3	1	3.02	827.3	KDMSS
C102	827.3	1	12.92	827.3	1	12.92	827.3	KDMSS
C103	826.3	1	33.89	826.3	1	33.89	826.3	KDMSS
C104	822.9	1	4113.09	822.9	1	4113.09	822.9	KDMSS
C105	827.3	1	5.34	827.3	1	5.34	827.3	KDMSS
C106	827.3	1	7.15	827.3	1	7.15	827.3	KDMSS
C107	827.3	1	6.55	827.3	1	6.55	827.3	KDMSS
C108	827.3	1	14.46	827.3	1	14.46	827.3	KDMSS
C109	827.3	1	20.53	827.3	1	20.53	827.3	KDMSS
RC101	1584.1	59	56.62	1619.8	1	12.39	1619.8	KDMSS
RC102			-	1457.4	1	76.69	1457.4	CR+KLM
RC103			-	1257.7	3	2705.78	1258.0	CR+KLM
RC104			-	1129.9	7	65806.79	1132.3	IV
RC105	1472.0	191	309.83	1513.7	1	26.73	1513.7	KDMSS
RC106			-	1367.3	37	15891.55	1372.7	S
RC107			-	1207.8	1	153.80	1207.8	IV
RC108			-	1114.2	1	3365.00	1114.2	IV
R201			-	1143.2	1	139.03	1143.2	KLM
R202			-	1027.3	13	8282.38	1029.6	**JPSP**
R203			-	870.8	1	54187.40	870.8	**JPSP**
R204			-			-	-	-
R205			-			-	-	-
R206			-			-	-	-
R207			-			-	-	-
R208			-			-	-	-
R209			-	854.8	3	78560.47	854.8	**JPSP**
R210			-			-	-	-
R211			-			-	-	-
C201	589.1	1	203.34	589.1	1	203.34	589.1	CR+KLM
C202	589.1	1	3483.15	589.1	1	3483.15	589.1	CR+KLM
C203	588.7	1	13070.71	588.7	1	13070.71	588.7	KLM
C204			-			-	588.1	IV
C205	586.4	1	416.56	586.4	1	416.56	586.4	CR+KLM
C206	586.0	1	594.92	586.0	1	594.92	586.0	CR+KLM
C207	585.8	1	1240.97	585.8	1	1240.97	585.8	CR+KLM
C208	585.8	1	555.27	585.8	1	555.27	585.8	KLM
RC201			-	1261.7	3	229.27	1261.8	KLM
RC202			-	1092.3	1	312.57	1092.3	IV+C
RC203	922.6	11	34063.95	923.7	1	14917.36	923.7	**JPSP**
RC204			-			-	-	-
RC205			-	1154.0	1	221.24	1154.0	IV+C
RC206			-	1051.1	1	339.69	1051.1	**JPSP**
RC207			-			-	-	-
RC208			-			-	-	-

Table 8: Instances with 100 customers.

B Solutions of Closed Solomon Instances

Cost	Route
8.8	53
119.2	70, 30, 20, 66, 65, 71, 35, 34, 78, 77, 28
105.4	92, 98, 91, 44, 14, 38, 86, 16, 61, 85,100, 37
84.1	2, 57, 15, 43, 42, 87, 97, 95, 94, 13, 58
106.5	73, 22, 41, 23, 67, 39, 56, 75, 74, 72, 21, 40
114.6	52, 88, 62, 19, 11, 64, 63, 90, 32, 10, 31
78.4	6, 96, 59, 99, 93, 5, 84, 17, 45, 83, 60, 89
107.3	26, 12, 80, 68, 29, 24, 55, 4, 25, 54
93.2	27, 69, 76, 3, 79, 9, 51, 81, 33, 50, 1
114.6	18, 7, 82, 8, 46, 36, 49, 47, 48
932.1	10

Table 9: Solution of R108.100. The left column is the cost of the routes and the total cost. The right column is a comma separated list indicating the customers visited on the routes in the order of visit and the total number of routes.

Cost	Route
78.1	6, 94, 95, 87, 42, 43, 15, 57, 58
115.8	2, 41, 22, 75, 56, 23, 67, 39, 25, 55, 54
117.4	28, 76, 79, 78, 34, 35, 71, 65, 66, 20, 1
128.2	31, 62, 19, 11, 63, 64, 49, 36, 47, 48
62.8	53, 40, 21, 73, 74, 72, 4, 26
98.0	52, 88, 7, 82, 8, 46, 45, 17, 84, 5, 89
76.4	12, 80, 68, 24, 29, 3, 77, 50
100.5	61, 16, 86, 38, 14, 44, 91,100, 37, 59, 96
67.6	18, 83, 60, 99, 93, 85, 98, 92, 97, 13
103.8	27, 69, 33, 81, 9, 51, 30, 32, 90, 10, 70
948.6	10

Table 10: Solution of R112.100.

Cost	Route
8.8	53
93.6	52, 62, 63, 90, 10, 32, 70
177.2	83, 45, 82, 48, 47, 36, 19, 11, 64, 49, 46, 17, 5, 60, 89
223.8	50, 33, 65, 71, 29, 76, 3, 79, 78, 81, 9, 51, 20, 66, 35, 34, 68, 77
140.2	27, 69, 1, 30, 31, 88, 7, 18, 8, 84, 86, 91,100, 37, 98, 93, 59, 94
67.1	40, 73, 41, 22, 74, 2, 58
148.9	28, 26, 21, 72, 75, 39, 67, 23, 56, 4, 54, 55, 25, 24, 80, 12
170.0	95, 92, 42, 15, 14, 38, 44, 16, 61, 85, 99, 96, 6, 87, 57, 43, 97, 13
1029.6	8

Table 11: Solution of R202.100.

Cost	Route
24.2	53, 40, 58
142.1	27, 69, 1, 76, 3, 79, 78, 81, 9, 66, 71, 35, 34, 29, 68, 77, 28
187.3	89, 18, 45, 46, 36, 47, 48, 19, 11, 62, 88, 7, 82, 8, 83, 60, 5, 84, 17, 61, 91,100, 37, 98, 93, 59, 94
183.3	95, 92, 97, 42, 15, 43, 14, 44, 38, 86, 16, 85, 99, 96, 6, 87, 57, 41, 22, 74, 73, 2, 13
190.3	50, 33, 51, 71, 65, 20, 30, 32, 90, 63, 64, 49, 10, 70, 31, 52
143.6	26, 21, 72, 75, 39, 67, 23, 56, 4, 55, 25, 54, 24, 80, 12
870.8	6

Table 12: Solution of R203.100.

Cost	Route
202.5	27, 31, 7, 48, 47, 36, 46, 45, 8, 18, 6, 37, 44, 14, 38, 16, 17, 5, 13
130.5	2, 42, 43, 15, 23, 39, 22, 41, 21, 40
242.5	28, 12, 3, 33, 50, 1, 30, 11, 49, 19, 10, 32, 20, 9, 35, 34, 29, 24, 25, 4, 26
575.5	3

Table 13: Solution of R207.50.

Cost	Route
146.8	52, 7, 82, 83, 18, 6, 94, 13, 87, 57, 15, 43, 42, 97, 92, 37,100, 91, 93, 96
198.7	95, 99, 59, 98, 85, 5, 84, 61, 16, 44, 14, 38, 86, 17, 45, 8, 46, 36, 49, 48, 60, 89
205.9	27, 69, 31, 88, 62, 47, 19, 11, 64, 63, 90, 30, 51, 71, 9, 81, 33, 79, 3, 77, 68, 80, 24, 54, 26
157.6	28, 12, 76, 29, 78, 34, 35, 65, 66, 20, 32, 10, 70, 1, 50
145.8	40, 2, 73, 21, 72, 75, 23, 67, 39, 25, 55, 4, 56, 74, 22, 41, 58, 53
854.8	5

Table 14: Solution of R209.100.

Cost	Route
139.4	81, 54, 72, 37, 36, 39, 42, 44, 41, 38, 40, 35, 43, 61, 68
172.8	90, 65, 83, 64, 85, 63, 89, 76, 23, 21, 48, 18, 19, 49, 22, 20, 51, 84, 56, 66
241.4	69, 98, 88, 53, 82, 99, 52, 86, 87, 9, 10, 47, 17, 13, 74, 59, 97, 75, 58, 77, 25, 24, 57
211.0	1, 3, 5, 45, 60, 12, 11, 15, 16, 14, 78, 73, 79, 7, 6, 8, 46, 4, 2, 55,100, 70
159.1	91, 92, 95, 62, 33, 32, 30, 27, 26, 28, 29, 31, 34, 50, 67, 94, 93, 71, 96, 80
923.7	5

Table 15: Solution of RC203.100.

Cost	Route
8.4	90
186.6	81, 94, 67, 84, 85, 51, 76, 89, 48, 25, 77, 58, 74
168.6	92, 71, 72, 42, 39, 38, 36, 40, 44, 43, 41, 37, 35, 54, 93, 96
180.9	65, 83, 64, 95, 62, 63, 33, 30, 31, 29, 27, 28, 26, 32, 34, 50, 56, 91, 80
189.6	61, 2, 45, 5, 8, 7, 79, 73, 78, 53, 88, 6, 46, 4, 3, 1,100, 70, 68
120.9	82, 99, 52, 86, 57, 23, 21, 18, 19, 49, 20, 22, 24, 66
196.1	69, 98, 12, 14, 47, 16, 15, 11, 59, 75, 97, 87, 9, 13, 10, 17, 60, 55
1051.1	7

Table 16: Solution of RC206.100.

References

[1] 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(3):316–329, 1998. doi: 10.1287/opre.46.3.316.

[2] J.E. Beasley and N. Christofides. An algorithm for the resource constrained shortest path problem. *Networks*, 19:379–394, 1989. doi: 10.1002/net.3230190402.

[3] N. Boland, J. Dethridge, and I. Dumitrescu. Accelerated label setting algorithms for the elementary resource constrained shortest path problem. *Operation Research Letters*, 34 (1):58–68, 2006. doi: 10.1016/j.orl.2004.11.011.

[4] A. Caprara, M. Fischetti, and A. N. Letchford. On the separation of maximally violated mod-k cuts. *Mathematical Programming*, 87(A):37–56, 1999. doi: 10.1007/s101079900107.

[5] A. Chabrier. Vehicle routing problem with elementary shortest path based column generation. *Computers & Operations Research*, 33(10):2972–2990, 2006. doi: 10.1016/j.cor.2005.02.029.

[6] V. Chvatal. Edmonds polytopes and hierarchy of combinatorial problems. *Discrete Mathematics*, 4(4):305–337, 1973. doi: 10.1016/0012-365X(73)90167-2.

[7] COIN. COIN — COmputational INfrastructure for Operations Research, 2005. http://www.coin-or.org.

[8] W. Cook and J. L. Rich. A parallel cutting plane algorithm for the vehicle routing problem with time windows. Technical Report TR99-04, Computational and Applied Mathematics, Rice University, Houston, Texas, USA, 1999.

[9] E. Danna and C. Le Pape. Branch-and-price heuristics: A case study on the vehicle routing problem with time windows. In G. Desaulniers, J. Desrosiers, and M. M. Solomon, editors, *Column Generation*, chapter 4, pages 99–129. Springer, 2005. doi: 10.1007/0-387-25486-2_4.

[10] G. Desaulniers, J. Desrosiers, J. Ioachim, I. M. Solomon, F. Soumis, and D. Villeneuve. A unified framework for deterministic time constrained vehicle routing and crew scheduling problems. In T. G. Crainic and G. Laporte, editors, *Fleet Management and Logistics*, pages 57–93. Kluwer, 1998.

[11] M. Desrochers. *La fabrication d̆2019horaires de travail pour les conducteurs d̆2019autobus par une méthode de génération de colonnes.* PhD thesis, Université de Montréal, Montréal, Canada, 1986.

[12] M. Desrochers, J. Desrosiers, and M. Solomon. A new optimization algorithm for the vehicle routing problem with time windows. *Operations Research*, 40(2):342–354, 1992. doi: 10.1287/opre.40.2.342.

[13] M. Dror. Note on the complexity of the shortest path models for column generation in VRPTW. *Operations Research*, 42(5):977–979, 1994. doi: 10.1287/opre.42.5.977.

[14] I. Dumitrescu. *Constrained Path and Cycle Problems.* PhD thesis, Department of Mathematics and Statistics, University of Melbourne, Australia, 2002.

[15] F. Eisenbrand. Note - on the membership problem for the elementary closure of a polyhedron. *Combinatorica*, 19(2):297–300, 1999. doi: 10.1007/s004930050057.

[16] D. Feillet, P. Dejax, M. Gendreau, and C. Gueguen. An exact algorithm for the elementary shortest path problem with resource constraints: Application to some vehicle routing problems. *Networks*, 44(3):216–229, 2004. doi: 10.1002/net.v44:3.

[17] R. Fukasawa, H. Longo, J. Lysgaard, M. Poggi de Aragão, M. Reis, E. Uchoa, and R.F. Werneck. Robust branch-and-cut-and-price for the capacitated vehicle routing problem. *Mathematical Programming*, 106(3):491–511, 2006. doi: 10.1007/s10107-005-0644-x.

[18] S. Irnich. Resource extension functions: Properties, inversion, and generalization to segments. *OR Spectrum*, 30(1):113–148, 2008. doi: 10.1007/s00291-007-0083-6.

[19] S. Irnich and G. Desaulniers. Shortest path problems with resource constraints. In G. Desaulniers, Jacques Desrosiers, and M.M. Solomon, editors, *Column Generation*, chapter 2, pages 33–65. Springer, 2005. doi: 10.1007/0-387-25486-2_2.

[20] S. Irnich and D. Villeneuve. The shortest path problem with resource constraints and k-cycle elimination for $k \geq 3$. *INFORMS Journal on Computing*, 18:391–406, 2006. doi: 10.1287/ijoc.1040.0117.

[21] M. Jepsen, B. Petersen, and S. Spoorendonk. A branch-and-cut-and-price framework for the VRP applied on CVRP and VRPTW. Master's thesis, DIKU Department of Computer Science, University of Copenhagen, Denmark, 2005.

[22] B. Kallehauge, J. Larsen, and O.B.G. Madsen. Lagrangean duality and non-differentiable optimization applied on routing with time windows - experimental results. Technical Report Internal report IMM-REP-2000-8, Department of Mathematical Modelling, Technical University of Denmark, Lyngby, Denmark, 2000.

[23] N. Kohl, J. Desrosiers, O. B. G. Madsen, M. M. Solomon, and F. Soumis. 2-path cuts for the vehicle routing problem with time windows. *Transportation Science*, 33(1):101–116, 1999. doi: 10.1287/trsc.33.1.101.

[24] J. Larsen. *Parallelization of the vehicle routing problem with time windows.* PhD thesis, Department of Mathematical Modelling, Technical University of Denmark, Lyngby, Denmark, 1999.

[25] J. Lysgaard. CVRPSEP: A package of separation routines for the capacitated vehicle routing problem. Working Paper 03-04, Dept. of Mgt. Science and Logistics, Aarhus School of Business., 2003.

[26] J. Lysgaard, A. N. Letchford, and R. W. Eglese. A new branch-and-cut algorithm for the capacitated vehicle problem. *Mathematical Programming*, 100(2):423–445, 2004. doi: 10.1007/s10107-003-0481-8.

[27] M. E. Lübbecke and J. Desrosiers. Selected topics in column generation. *Operations Research*, 53(6):1007–1023, 2005. doi: 10.1287/opre.1050.0234.

[28] G. Nemhauser and S. Park. A polyhedral approach to edge coloring. *Operations Research Letters*, 10(6):315–322, 1991. doi: 10.1016/0167-6377(91)90003-8.

[29] G. L. Nemhauser and L. A. Wolsey. *Integer and Combinatorial Optimization*. John Wiley & Sons, Inc., 1988.

[30] D. Pisinger and L. B. Reinhardt. Multi-objective non-additive shortest path. DIKU Department of Computer Science, University of Copenhagen, Denmark, submitted, 2007.

[31] G. Righini and M. Salani. Symmetry helps: bounded bi-directional dynamic programming for the elementary shortest path problem with resource constraints. *Discrete Optimization*, 3(3):255–273, 2006. doi: 10.1016/j.disopt.2006.05.007.

[32] G. Righini and M. Salani. New dynamic programming algorithms for the resource constrained shortest path problem. *Networks*, 51(3):155–170., 2008. doi: 10.1002/net.20212.

[33] M. Salani. *Branch-and-Price Algorithms for Vehicle Routing Problems*. PhD thesis, Universitá Degli Studi Di Milano, Facoltá di Scienza Matematiche, Fisuche e Naturali Dipartimento di Technologie dell'Informazione, Milano, Italy, 2005.

[34] M. M. Solomon. Algorithms for the vehicle routing and scheduling problem with time window constraints. *Operations Research*, 35(2):234–265, 1987. doi: 10.1287/opre.35.2.254.

[35] SPEC. Standard Performance Evaluation Corporation, 2005. http://www.spec.org.

[36] P. Toth and D. Vigo, editors. *The Vehicle Routing Problem*. SIAM, 2002.

[37] L. A. Wolsey. *Integer Programming*. John Wiley & Sons, Inc., 1998.

Chapter 4

Chvátal-Gomory Rank-1 Cuts used in a Dantzig-Wolfe Decomposition of the Vehicle Routing Problem with Time Windows

Bjørn Petersen
DIKU Department of Computer Science, University of Copenhagen

David Pisinger
DIKU Department of Computer Science, University of Copenhagen

Simon Spoorendonk
DIKU Department of Computer Science, University of Copenhagen

Abstract

This chapter shows how Chvátal-Gomory (CG) rank-1 cuts can be used in a branch-and-cut-and-price algorithm for the vehicle routing problem with time windows (VRPTW). Using Dantzig-Wolfe decomposition we split the problem into a set partitioning problem as master problem and an elementary shortest path problem with resource constraints as pricing problem. To strengthen the formulation we derive general CG rank-1 cuts based on the master problem formulation. Adding these cuts to the master problem means that an additional resource is added to the pricing problem for each cut. This increases the complexity of the label algorithm used to solve the pricing problem since normal dominance tests become weak when many resources are present and hence most labels are incomparable. To overcome this problem we present a number of improved dominance tests exploiting the step-like structure of the objective function of the pricing problem. Computational experiments are reported on the Solomon test instances showing that the addition of CG rank-1 cuts improves the lower bounds significantly and makes it possible to solve a majority of the instances in the root node of the branch-and-bound tree. This indicates that CG rank-1 cuts may be essential for solving future large-scale VRPTW

Published in the book *The Vehicle Routing Problem: Latest Advances and New Challenges*, 2008

problems where we cannot expect that the branching process will close the gap between lower and upper bounds in reasonable time.

Keywords: Vehicle routing problem with time windows, Dantzig-Wolfe decomposition, Chvatal-Gomory rank-1 cuts.

1 Introduction

In the vehicle routing problem with time windows (VRPTW) we are given a set of customers with an associated demand and a number of identical vehicles. The task is to find a set of minimum-length routes starting and ending at a central depot such that each customer is visited exactly once within a given time window, and the capacity of each vehicle is respected.

The standard Dantzig-Wolfe decomposition of the arc flow formulation of the VRPTW is to split the problem into a master problem (a set partitioning problem with a convexity constraint, stating that all customers should be visited with a limited number of vehicles) and a pricing problem (an elementary shortest path problem with resource constraints (ESPPRC), where capacity and time are the constrained resources). Delayed column generation may be used to solve the LP-relaxed master problem, which can be used as lower bound in a branch-and-bound algorithm to reach integrality. Applying cutting planes either in the master or the pricing problem leads to a branch-and-cut-and-price algorithm (BCP).

BCP algorithms have been frequently used to solve the VRPTW, e.g., Kohl et al. [25], Cook and Rich [6], Larsen [26], Kallehauge et al. [24], Irnich and Villeneuve [22], Chabrier [4], Danna and Le Pape [7], Salani [31]. In all cases the valid inequalities have been based on the *original* arc flow formulation of the VRPTW, i.e., the inequalities added are valid for both the *original* arc formulation and the master problem. Fukasawa et al. [16] refer to this as a *robust* approach. Recently Jepsen et al. [23] showed how the subset row (SR) inequalities, which are valid inequalities for the set partitioning problem, successfully can be applied to VRPTW in a column generation context. In their computational results they report solving 8 out of 18 previously unsolved instances from the set of benchmarks by Solomon [33]. In a following paper Desaulniers et al. [9] added fast pricing heuristics and improved cutting policies for the SR inequalities to obtain even better results by closing an additional 5 instances. The latter approaches are denoted *non-robust* according to the classification by Fukasawa et al. [16], since the complexity of the pricing problem is increased when SR inequalities are added to the master problem.

Jepsen et al. [23] showed that the separation of SR inequalities is $\mathcal{NP}$-hard and that the inequalities can be recognized as a subset of the Chvátal-Gomory (CG) rank-1 cuts. A simple enumeration algorithm was used to separate the SR inequalities for sets of rows of size three, and even for such small sets the computational results were very good as mentioned above. Not surprisingly the separation of CG rank-1 cuts is also known to be $\mathcal{NP}$-hard, see Eisenbrand [13]. Fischetti and Lodi [15] used the CG rank-1 cuts as cutting planes in an integer problem and showed how the separation can be formulated as a mixed integer problem. They obtained lower bounds when optimizing over the first Chvátal closure, i.e., adding violated CG rank-1 cuts, and were the first to report an optimal solution to one instance from MIPLIB 3.0 by Bixby et al. [1]. These results motivate the incorporation of the CG rank-1 cuts in a BCP algorithm.

The pricing problem of the Dantzig-Wolfe decomposition of VRPTW, i.e., the ESPPRC, was shown to be $\mathcal{NP}$-hard by Dror [11]. Commonly the ESPPRC has been solved with labeling algorithms, see Dumitrescu [12], Feillet et al. [14], Righini and Salani [29, 30], Boland et al. [2]. Due to the difficulty of the ESPPRC most earlier approaches solved relaxations of the ESPPRC, see Desrochers et al. [10], Irnich and Villeneuve [22]. For a general introduction to resource constrained shortest path problems, see Desaulniers et al. [8], Irnich and Desaulniers [21], Irnich [20]. Jepsen et al. [23] provides an introduction of the SR inequalities and how their application in the master problem leads to an additional resource per inequality in the pricing problem. Furthermore, it is shown how the dominance criteria of the label algorithm can be improved.

In this chapter we extend the work by Jepsen et al. [23] to include general CG rank-1 cuts for the Set Partitioning master problem. Each cut results in a new resource constraint in the ESPPRC pricing problem. As the resource extension functions are non-decreasing any dynamic programming algorithm for the ESPPRC can be used to solve the resulting problem. However, the addition of new resources means that more labels become incomparable when using a traditional dominance test, and hence the number of labels in the dynamic programming explodes. To overcome this problem we exploit the fact that in the pricing problem it is sufficient to find a cost-minimal solution, and not all Pareto-optimal solutions. Due to this fact we may temporarily replace each label with a number of equivalent labels such that resources become comparable in the dominance test. This approach considerably decreases the number of labels generated in the dynamic programming algorithm. As demonstrated in the computational results we can in this way solve the ESPPRC pricing problem even when several hundreds of CG rank-1 cuts have been added, and hence several hundreds of resources are to be dealt with in the label algorithm.

The chapter is organized as follows: In Section 2 we give an overview of the Dantzig-Wolfe decomposition of the VRPTW and describe how to calculate the reduced cost of columns when delayed column generation is used. For completeness we review the CG rank-1 cuts and their separation, as described by Fischetti and Lodi [15], in Section 3. Furthermore, we clarify how to use these techniques in a VRPTW context. In Section 4 the improved dominance criteria of the label algorithm are described. An algorithmic outline, implementation details, and computational results using the Solomon benchmark instances are presented in Section 5. Section 6 provides some concluding remarks.

2 Decomposition

Let C be the set of customers, and let the set of nodes be $V = C \cup \{o, o'\}$ where o denotes the depot at the start of the routes and o' denotes the depot at the end. Each customer $i \in C$ has a demand d_i while we set $d_o = d_{o'} = 0$. Each node $i \in V$ has an associated service s_i and a time windows $[a_i, b_i]$ in which it should be visited.

Let $E = \{(i,j) : i,j \in V,\ i \neq j\}$ be the set of arcs between the nodes. The set of vehicles K is sufficiently large, e.g., $|K| = V$, such that the convexity constraint is not binding, and each vehicle has capacity D. If vehicle $k \in K$ service node $i \in V$ then the variable t_{ik} denotes the arrival time of the vehicle. Let c_{ij} be the travel cost on arc $(i,j) \in E$ and let x_{ijk} be the variable indicating whether vehicle $k \in K$ traverses arc $(i,j) \in E$. The overall travel time τ_{ij} on arc $(i,j) \in E$ depends on the travel time of the arc and the service time s_i at customer i.

The 3-index flow model (Toth and Vigo [34]) for the VRPTW becomes:

$$\min \sum_{k\in K}\sum_{(i,j)\in E} c_{ij}x_{ijk} \tag{1}$$

$$\text{s.t.} \sum_{k\in K}\sum_{(i,j)\in\delta^+(i)} x_{ijk} = 1 \qquad \forall i \in C \tag{2}$$

$$\sum_{(i,j)\in\delta^+(o)} x_{ijk} = \sum_{(i,j)\in\delta^-(o')} x_{ijk} = 1 \qquad \forall k \in K \tag{3}$$

$$\sum_{(j,i)\in\delta^-(i)} x_{jik} - \sum_{(i,j)\in\delta^+(i)} x_{ijk} = 0 \qquad \forall i \in C,\ \forall k \in K \tag{4}$$

$$\sum_{(i,j)\in E} d_i x_{ijk} \le D \qquad k \in K \tag{5}$$

$$a_i \le t_{ik} \le b_i \qquad \forall i \in V,\ \forall k \in K \tag{6}$$

$$x_{ijk}(t_{ik} + \tau_{ij}) \le t_{jk} \qquad \forall (i,j) \in E,\ \forall k \in K \tag{7}$$

$$x_{ijk} \in \{0,1\} \qquad \forall (i,j) \in E,\ \forall k \in K \tag{8}$$

Constraints (2) ensure that every customer $i \in C$ is visited, and (3) ensures that each route starts and ends in the depot. Constraint set (4) ensure flow conservation for each vehicle k. Note that a zero-cost arc $x_{oo'k}$ between the start and end depot must be present for all vehicles to allow an empty tour in case not all vehicles are needed. The constraint set (5) ensures that the capacity of each vehicle is not exceeded and constraint sets (6) and (7) ensure that the time window constraints are satisfied. Note that (7) together with the assumption that $\tau_{ij} > 0$ for all $(i,j) \in E$ eliminates all sub-tours. The last constraint define the domain of the arc flow variables.

The standard Dantzig-Wolfe decomposition of the VRPTW, see e.g. Desrochers et al. [10], leads to the following master problem:

$$\min \sum_{p\in P}\sum_{(i,j)\in E} c_{ij}\alpha_{ijp}\lambda_p \tag{9}$$

$$\text{s.t} \sum_{p\in P}\sum_{(i,j)\in\delta^+(i)} \alpha_{ijp}\lambda_p = 1 \qquad \forall i \in C \tag{10}$$

$$\lambda_p \in \{0,1\} \qquad \forall p \in P \tag{11}$$

where P is the set of all feasible routes, the binary constant α_{ijp} is one if and only if arc (i,j) is used by route $p \in P$, and the binary variable λ_p indicates whether route p is used. The master problem is a set partitioning problem and the LP relaxation can be solved using delayed column generation, i.e., consider a *restricted* master problem containing a subset of the columns P and generate additional columns as needed. For the remainder of this chapter the master problem will refer to the the restricted problem. Let $\pi_i \in \mathbb{R}$ for all $i \in C$ be the dual values of (10) and let $\pi_0 = 0$. Then the reduced cost of a route p is:

$$\bar{c}_p = \sum_{(i,j)\in E} c_{ij}\alpha_{ijp} - \sum_{(i,j)\in E} \pi_j\alpha_{ijp} = \sum_{(i,j)\in E} (c_{ij} - \pi_j)\alpha_{ijp} \tag{12}$$

The pricing problem is an ESPPRC where the cost of each arc is $\bar{c}_{ij} = c_{ij} - \pi_j$ for all arcs $(i,j) \in E$.

Valid inequalities based on the VRPTW constraints (2)-(8), i.e.,

$$\sum_{k\in K}\sum_{(i,j)\in E}\beta_{ij}x_{ijk} \leq \beta_0 \tag{13}$$

are handled as follows (Note that β_{ij} can be dependent on a vehicle k but then different pricing problems must be considered). Let μ be the dual values of (13), then an additional $\mu\beta_{ij}$ for all arcs $(i,j) \in E$ has to be subtracted from the reduced cost of a route, i.e., by subtracting the dual value from the arc cost in the the pricing problem, i.e., $\overline{c}_{ij} = c_{ij} - \pi_j - \mu\beta_{ij}$.

Consider adding a valid inequality for the set partitioning master problem (10)–(11) that cannot be written as a linear combination of the arc flow variables, i.e.,

$$\sum_{p\in P}\beta_p\lambda_p \leq \beta_0 \tag{14}$$

Let $\sigma \leq 0$ be the dual values of (14), then an additional $\sigma\beta_p$ has to be subtracted when calculating the reduced cost of the column, i.e, the new reduced cost is $\hat{c}_p = \overline{c}_p - \sigma\beta_p$. To handle the cost $-\sigma\beta_p$ it is necessary to modify the pricing problem by adding constraints or variables, thereby increasing its complexity.

3 Chvátal-Gomory Rank-1 Cuts

CG cuts are well known valid inequalities for integer programming problems, see Gomory [17], Chvatal [5]. However, in a BCP context these cuts have been given little attention. Except for the recent papers by Jepsen et al. [23], Desaulniers et al. [9] only an early attempt by Nemhauser and Park [28] has been found where general mixed-integer cuts for the master problem is applied. Nemhauser and Park [28] solved the pricing problem as a MIP by adding additional variables and constraints to take the dual values of the applied cuts into account. As noted in Jepsen et al. [23], the SR inequalities are a subset of the CG cuts, and since the SR inequalities were successfully used for VRPTW an obvious extension is to include a larger set of the CG cuts into the BCP framework. Hence, in the following the focus will be on the CG rank-1 cuts and their separation starting with the general case as described by Fischetti and Lodi [15]. Next we specify the form of CG rank-1 cuts for the master problem of the VRPTW and formulate the separation problem based the presented theory. Last we briefly discuss the interpretation of the SR inequalities with regards to the CG cuts.

Consider an IP problem:

$$\min\{c\lambda : A\lambda \leq b, \lambda \geq 0, \lambda \in \mathbb{Z}^n\}$$

where A is a $m \times n$ matrix, $N = 1, \ldots, n$ is the set of indices of variables, and $M = 1, \ldots, m$ is the set of indices of constraints. The two polyhedra

$$P_{LP} = \{\lambda \in \mathbb{R}^n : A\lambda \leq b, \lambda \geq 0\}$$
$$P_{IP} = conv\{\lambda \in \mathbb{Z}^n : A\lambda \leq b, \lambda \geq 0\} = conv(P_{LP} \cap \mathbb{Z}^n)$$

describe the solution space of the linear relaxation P_{LP} and the convex hull of the integer solutions in P_{LP}. It is assumed that all coefficients of A and b are integer. A CG cut is a valid inequality for P_{IP} given as:

$$\lfloor uA \rfloor \lambda \leq \lfloor ub \rfloor$$

where $u \geq 0$ is called the CG multiplier vector. The inequality is said to have *rank-1* with respect to $A\lambda \leq b$ and $\lambda \geq 0$. Higher rank cuts are obtained by considering systems that also contain lower rank CG cuts, e.g., a rank-2 cut is based on $A\lambda \leq b$ and $\lambda \geq 0$ and some rank-1 cuts. Note that given the above assumptions on the integrality of A and b, undominated CG cuts only arise for rational CG multipliers $u_i \in [0, 1)$, for all $i = 1, \ldots, m$, see Schrijver [32].

The first *Chvátal closure* of P_{LP} is defined as the polyhedron:

$$P_1 = \{\lambda \leq 0 : A\lambda \leq b, \lfloor uA \rfloor \lambda \leq \lfloor ub \rfloor, u \geq 0 \ \forall u \in \mathbb{R}^n\}$$

Clearly $P_{IP} \subseteq P_1 \subseteq P_{LP}$ but even more interesting is it, that $P_1 \subset P_{LP}$ iff $P_{IP} \neq P_{LP}$. The better approximation of P_{IP} is obtained, since it is possible to use a CG cut to cut off a fractional vertex $\lambda^* \in P_{LP}$ corresponding to the basis B by choosing multipliers u equal to the ith row of B^{-1} where i is the row associated with any fractional part of λ^*, see Gomory [17, 18].

The separation problem is stated by Fischetti and Lodi [15] as:

Definition 1. *Given a point $\lambda* \in P_{LP}$. The CG separation problem consists of finding a CG cut that is violated by λ^*, i.e., find $u \geq 0$ for $u \in \mathbb{R}^n$ such that $\lfloor uA \rfloor \lambda > \lfloor ub \rfloor$, or prove that no such u exist.*

Eisenbrand [13] showed that the separation problem is $\mathcal{NP}$-hard and computational results performed by Fischetti and Lodi [15] indicate that separation times can be cumbersome.

Given a fractional solution $\lambda^* \in P_{LP}$ the maximally violated CG cut $\gamma\lambda \leq \gamma_0$, where $\gamma = \lfloor uA \rfloor$ and $\gamma_0 = \lfloor ub \rfloor$ for some CG multipliers $u \geq 0$ for $u \in \mathbb{R}^n$ can be found by solving the following MIP:

$$\max \gamma\lambda^* - \gamma_0 \tag{15}$$
$$\gamma_j \leq uA_j \qquad \forall j \in N \tag{16}$$
$$\gamma_0 > ub - 1 \tag{17}$$
$$u_i \geq 0 \qquad \forall i \in M \tag{18}$$
$$\gamma_j \in \mathbb{Z} \qquad \forall j \in N \cup \{0\} \tag{19}$$

Note that only basis variables with non-zero values can contribute to the violation of the CG rank-1 cut. Hence, all zero valued variables can be left out of the formulation and their coefficients can be calculated after the CG multipliers are identified. This reduces the size of the MIP problem in both the number of variables and constraints.

Furthermore Fischetti and Lodi [15] suggest to reformulate the problem in order to obtain a stronger formulation and numerical stability. Based on the fact that the CG multipliers of undominated cuts are less than 1, bounding them from above provides a stronger formulation. However, later observations showed that the MIP heuristics performed much better without these bounds. To obtain numerical stability a slack variable $f_j \in [0, 1 - \delta]$ (e.g., $\delta = 0.01$) is introduced for each coefficient α_j.

Equivalent solutions to the separation problem can result in CG rank-1 cuts of different strength with respect to P_{IP}. A strong cut tends to be sparse, i.e., the number of non-zero entries is small. In order to obtain stronger and sparser cuts the objective function is modified by adding a small penalty w_i (e.g., $w_i = 0.0001$) for the selection of a multiplier u_i.

Let $N(\lambda^*)$ is the set of non-zero basis variables. This leads to the following formulation of the separation problem:

$$\max \sum_{j \in N(\lambda^*)} \gamma_j \lambda_j^* - \gamma_0 - \sum_{i \in M} w_i u_i \tag{20}$$
$$f_j = uA_j - \gamma_j \qquad \forall j \in N(\lambda^*) \tag{21}$$
$$f_0 = ub - \gamma_0 \tag{22}$$
$$0 \leq f_j \leq 1 - \delta \qquad \forall j \in N(\lambda^*) \cup \{0\} \tag{23}$$
$$u_i \geq 0 \qquad \forall i \in M \tag{24}$$
$$\gamma_j \in \mathbb{Z} \qquad \forall j \in N(\lambda^*) \cup \{0\} \tag{25}$$

The model (20)-(25) can be modified to handle systems as $A\lambda \geq b$ and $A\lambda = b$ by modifying the bounds of the CG multipliers, i.e., removing (24) and letting u be a free variables is a way to handle equations.

For VRPTW the the CG rank-1 cuts are based on the master problem constraints (10). The set partitioning constraints give rise to cuts with CG multipliers $u \in \mathbb{R}^{|C|}$, since they are equalities. However, since the CG cuts will be used in a column generation context two equally sparse cuts at separation time might not be equally sparse after column generation. This is especially the case for CG rank-1 cuts with negative multipliers in a minimization problem, where cuts tend to become very dense when columns price into the master problem. Hence, we restrict ourselves to consider CG rank-1 cuts with non-negative multipliers for the VRPTW.

The CG rank-1 cuts for the VRPTW with respect to the master problem (9)-(11) and with non-negative CG multipliers are given as:

$$\sum_{p \in P} \left\lfloor \sum_{i \in C} u_i \sum_{(i,j) \in \delta^+(i)} \alpha_{ijp} \right\rfloor \lambda_p \leq \left\lfloor \sum_{i \in C} u_i \right\rfloor \tag{26}$$

Given a fractional solution λ^* for the master problem (9)-(11) the most violated CG cut of rank-1 can be found by solving the following MIP:

$$\max \sum_{p \in P(\lambda^*)} \gamma_p \lambda_p^* - \gamma_0 - \sum_{i \in C} w_i u_i \tag{27}$$
$$f_p = \sum_{(i,j) \in \delta(i)^+} \alpha_{ijp} u_i - \gamma_p \qquad \forall p \in P(\lambda^*) \tag{28}$$
$$f_0 = \sum_{i \in C} u_i - \gamma_0 \tag{29}$$
$$0 \leq f_p \leq 1 - \delta \qquad \forall p \in P(\lambda^*) \cup \{0\} \tag{30}$$
$$0 \leq u_i \qquad \forall i \in C \tag{31}$$
$$\gamma_j \in \mathbb{Z}^+ \qquad \forall p \in P(\lambda^*) \cup \{0\} \tag{32}$$

Again it is possible to reduce the number of variables by only considering the non-zero basis variables.

From Jepsen et al. [23] we recall the SR inequalities for the VRPTW based on the master problem (9)-(11):

$$\sum_{p \in P} \left\lfloor \frac{1}{k} \sum_{i \in S} \sum_{(i,j) \in \delta^+(i)} \alpha_{ijp} \right\rfloor \lambda_p \leq \left\lfloor \frac{1}{k} |S| \right\rfloor \tag{33}$$

where $S \subseteq C$ and $0 < k \leq |S|$. This is equivalent to the set of CG rank-1 cuts where $|S|$ of the CG multipliers are equal to $\frac{1}{k}$ and the rest are equal to 0, i.e., a very sparse CG multiplier vector. A SR cut can also be interpreted as a mod-k cut proposed by Caprara et al. [3]. The mod-k cuts are CG rank-1 cuts with multipliers in the set $\{0, \frac{1}{k}, \dots, \frac{k-1}{k}\}$, i.e., a SR cut is a mod-k cut with $|S|$ multipliers equal to $\frac{1}{k}$ and the rest are equal to 0. Extending the SR cut to allow a row (customer) to be present multiple times in S, i.e., let S be a multiset, leads to an SR cut with maximal $|S|$ multipliers in the set $\{0, \frac{1}{k}, \dots, \frac{k-1}{k}\}$. That is, the CG multiplier of a row is raised by $\frac{1}{k}$ for each time it is present in S. This is indeed also a mod-k cut.

4 Label Algorithm

Finding a route with negative reduced cost in the pricing problem corresponds to finding a negative reduced cost path starting and ending at the depot, i.e., an ESPPRC. In the following sections we formally describe the ESPPRC and show how the pricing problem can be solved when new resources are introduced as a consequence of adding CG cuts.

4.1 The Pricing Problem

Assuming that no cuts have been added, the ESPPRC can be formally defined as: Given a weighted directed graph $G(V, E)$ with nodes V and arcs E, and a set of resources R. For each arc $(i,j) \in E$ and resource $r \in R$ three parameters are given: A lower limit $a_r(i,j)$ on the accumulation of resource r when traversing arc $(i,j) \in E$; an upper limit $b_r(i,j)$ on the accumulation of resource r when traversing arc $(i,j) \in E$; and finally an amount $c_r(i,j)$ of resource r consumed by traversing arc $(i,j) \in E$. The objective is to find a minimum cost path p from a source node $o \in V$ to a target node $o' \in V$, where the accumulated resources of p satisfy the limits for all resources $r \in R$. Without loss of generality we assume that the limits must be satisfied at the end of each arc (i,j), i.e., after $c_r(i,j)$ has been consumed.

If the nodes have associated some resource consumptions and some upper and lower limits on the accumulated resources are present, these can be expressed by equivalent resource constraints on the arcs (e.g. the incoming arcs of the node).

For the pricing problem of VRPTW the resources are load d, time t, and a binary visit-counter for each customer $v \in C$. When considering the pricing problem of VRPTW, the consumptions and upper and lower limits of the resources at each arc (i,j) in ESPPRC are:

$$\begin{array}{llll}
a_d(i,j) = 0, & b_d(i,j) = D - d_j, & c_d(i,j) = d_j & \forall (i,j) \in E \\
a_t(i,j) = a_i, & b_t(i,j) = b_i, & c_t(i,j) = \tau_{ij} & \forall (i,j) \in E \\
a_v(i,j) = 0, & b_v(i,j) = 1, & c_v(i,j) = 1 \quad \forall v \in V : v = j, & \forall (i,j) \in E \\
a_v(i,j) = 0, & b_v(i,j) = 1, & c_v(i,j) = 0 \quad \forall v \in V : v \neq j, & \forall (i,j) \in E
\end{array}$$

In the label algorithm labels at node v represent partial paths from o to v. The following attributes for a label L are considered:

$\overline{v}(L)$	The current end-node of the partial path represented by L.
$\overline{c}(L)$	The sum of the reduced cost along path L.
$r(L)$	The accumulated consumption of resource $r \in R$ along path L.

A feasible extension $\epsilon \in \mathcal{E}(L)$ of a label L is a partial path starting in node $\overline{v}(L) \in V$ and ending in the target node o' without violating any resource constraints when concatenated with the partial path represented by L.

In the following it is assumed that all resources are bounded strongly from above, and weakly from below. This means that if the current resource accumulation of a label is below the lower limit on a given arc, it is allowed to fill up the resource to the lower limit, i.e., waiting for a time window to open. This means that two consecutive labels L_u and L_v related by an arc (u, v), i.e., L_u is extended and creates L_v, where $\overline{v}(L_u) = u$ and $\overline{v}(L_v) = v$, must satisfy

$$r(L_v) \leq b_r(u, v), \qquad \forall r \in R \tag{34}$$

$$r(L_v) = \max\{r(L_u) + c_r(u, v), a_r(u, v)\}, \qquad \forall r \in R \tag{35}$$

Here (34) demands that each label L_v satisfies the upper limit $b_r(u, v)$ of resource r corresponding to arc (u, v), while (35) states that resource r of L_v corresponds to the resource consumption at label L_u plus the amount consumed by traversing arc (u, v), respecting the lower limit $a_r(u, v)$ on arc (u, v). Other authors refer to (35) as a *resource extension function*, see e.g. Desaulniers et al. [8].

A simple enumeration algorithm could be used to produce all these labels, but to limit the number of labels considered, dominance rules are introduced to fathom labels which do not lead to an optimal solution.

Definition 2. *A label L_i* dominates *label L_j if*

$$\overline{v}(L_i) = \overline{v}(L_j) \tag{36}$$

$$\overline{c}(L_i) \leq \overline{c}(L_j) \tag{37}$$

$$\mathcal{E}(L_j) \subseteq \mathcal{E}(L_i) \tag{38}$$

In other words, the paths corresponding to labels L_i and L_j should end at the same node $\overline{v}(L_i) = \overline{v}(L_j) \in V$, the path corresponding to label L_i should cost no more than the path corresponding to label L_j, and finally any feasible extension of L_j is also a feasible extension of L_i. Notice that we are only interested in one cost-minimal path and not all pareto-optimal paths, hence our dominance rule is tighter than the one used in e.g. Desaulniers et al. [8], Irnich and Desaulniers [21].

Feillet et al. [14] suggested to consider the set of nodes that cannot be reached from a label L_i and compare the set with the unreachable nodes of a label L_j in order to determine if some extensions are impossible and thereby potentially dominate where else not possible, since $v_{old}(L_i) \leq v_{old}(L_j) \Rightarrow v_{new}(L_i) \leq v_{new}(L_j)$ but $v_{new}(L_i) \leq v_{new}(L_j) \not\Rightarrow v_{old}(L_i) \leq v_{old}(L_j)$. Or in other words: update the node resources in an eager fashion instead of a lazy one. The following definition is a generalization of Feillet et al. [14][Definition 3].

Definition 3. *Given a start node $o \in V$, a label L, and a node $u \in V$ where $\overline{v}(L) = u$ a node $v \in V$ is considered* unreachable *if v has already been visited on the path from o to u or if a resource window is violated, i.e.:*

$$\exists r \in R \qquad r(L) + \ell_r(u, v) > b_r(v)$$

where $\ell_r(u,v)$ is a lower bound on the consumption of resource r on all feasible paths from u to v. The node resources *are then given as: $v(L) = 1$ indicates that node $v \in V$ is unreachable from node $\overline{v}(L) \in V$, and $v(L) = 0$ otherwise.*

To determine if (38) holds can be quite cumbersome, as the straightforward definition demands that we calculate all extensions of the two labels. Therefore a sufficient criterion for (38) to hold is sought which can be computed faster. If label L_i has consumed less resources than label L_j then no resources are limiting the possibilities of extending L_i compared to L_j, hence the following proposition can be used as a relaxed version of the dominance criteria.

Proposition 4. Desaulniers et al. [8]. *If all resource extension functions are non-decreasing, then label L_i dominates label L_j if:*

$$\overline{v}(L_i) = \overline{v}(L_j) \tag{39}$$

$$\overline{c}(L_i) \leq \overline{c}(L_j) \tag{40}$$

$$r(L_i) \leq r(L_j) \qquad \forall r \in R \tag{41}$$

Using Proposition 4 as a dominance criteria is a relaxation of the dominance criteria of Definition 2 since only a subset of labels satisfying (36), (37), and (38) satisfies inequalities (39), (40), and (41).

4.2 Solving the Pricing Problem with New Resources

Recall that a CG rank-1 cut (26) for the VRPTW master problem (9)–(11) is:

$$\sum_{p \in P} \left\lfloor \sum_{i \in C} u_i \sum_{(i,j) \in \delta^+(i)} \alpha_{ijp} \right\rfloor \lambda_p \leq \left\lfloor \sum_{i \in C} u_i \right\rfloor$$

Let $\sigma \leq 0$ be the corresponding dual variable when solving the master problem to LP-optimality. The reduced cost of column p in the VRPTW master problem is:

$$\hat{c}_p = \overline{c}_p - \sigma \left\lfloor \sum_{i \in C} u_i \sum_{(i,j) \in \delta^+(i)} \alpha_{ijp} \right\rfloor = \sum_{(i,j) \in E} \overline{c}_{ij} \alpha_{ijp} - \sigma \left\lfloor \sum_{i \in C} u_i \sum_{(i,j) \in \delta^+(i)} \alpha_{ijp} \right\rfloor$$

We analyze how this additional cost can be handled in the label algorithm for ESPPRC.

Let $V(L) = \{i \in V : i(L) = 1\}$ be the nodes visited on the partial path of label L. The reduced cost of L can then be expressed as:

$$\hat{c}(L) = \overline{c}(L) - \sigma \left\lfloor \sum_{i \in V(L)} u_i \right\rfloor \tag{42}$$

A new resource m can be used to compute the coefficient of penalty σ for label L, i.e., $m(L) = \sum_{i \in V(L)} u_i$, is the unfloored amount involved in the cut. Note that the consumption of resource m is u_i for each outgoing (incoming) arc of the customers $i \in C$. Even though the update of resource $\hat{c}$ is defined by a decreasing function, the usual dominance criteria of Proposition 4 can still be used, because in case L_i dominates L_j, $\overline{c}(L_i) \leq \overline{c}(L_j)$ and $m(L_i) \leq m(L_j)$ so $\hat{c}(L_i) \leq \hat{c}(L_j)$ since $-\sigma > 0$. Note that the resource $\hat{c}$ can be ignored during the label algorithm and only be considered at the last arc to the target node to compute the reduced cost $\hat{c}(L)$ of path L from $\overline{c}(L)$ and $m(L)$.

Since all resource extension functions (including $m(L)$) are non-decreasing we can apply the label algorithm described in the previous section to solve the ESPPRC, using the dominance rule from Proposition 4 for the extended set of resources. However, as further cuts are added and hence more resources are to be compared in (41) the dominance rule is satisfied very rare. In order to overcome this problem, we note the following property of constraint (42)

$$\hat{c}(L) = \overline{c}(L) - \sigma \lfloor m(L) \rfloor = \overline{c}(L) + k\sigma - \sigma \lfloor m(L) - k \rfloor \tag{43}$$

for any integer k. Hence a label $(\hat{c}(L), r(L), m(L))$ is equivalent to a label $(\hat{c}(L) - k\sigma, r(L), m(L) - k)$, meaning that we can make resources comparable in (41) at the cost of modifying $\overline{c}(L)$ in (40) and vice versa. This is the main idea in Proposition 5, 6 and 7 to be presented.

For a label L let

$$\mathcal{T}(L) = \left(\sum_{i \in V(L)} u_i \right) \bmod 1$$

be the amount involved in the cut since the last penalty was paid, i.e., the fractional part of $\sum_{i \in V(L)} u_i$. Recall $\mathcal{E}(L)$ as the set of feasible extensions from the label L to the target node o' and note that when label L_i dominates label L_j, their common extensions are $\mathcal{E}(L_j)$ due to (38). The following cost dominance criteria are obtained for a single CG rank-1 cut:

Proposition 5. *If* $\mathcal{T}(L_i) \le \mathcal{T}(L_j)$, $\overline{v}(L_i) = \overline{v}(L_j)$, $\hat{c}(L_i) \le \hat{c}(L_j)$, *and* $r(L_i) \le r(L_j)\ \ \forall r \in R$, *then label* L_i *dominates label* L_j.

Proof. Consider any common extension $\epsilon \in \mathcal{E}(L_j)$. Since $\mathcal{T}(L_i) \le \mathcal{T}(L_j)$ the relation between the number of future penalties for the two labels when concatenated with ϵ is:

$$\left\lfloor \sum_{i \in \epsilon} u_i + \mathcal{T}(L_i) \right\rfloor \le \left\lfloor \sum_{i \in \epsilon} u_i + \mathcal{T}(L_j) \right\rfloor$$

This leads to the following relation between the costs:

$$\begin{aligned} \hat{c}(L_i + \epsilon) &= \hat{c}(L_i) + \overline{c}(\epsilon) - \sigma \left\lfloor \sum_{i \in \epsilon} u_i + \mathcal{T}(L_i) \right\rfloor \\ &\le \hat{c}(L_j) + \overline{c}(\epsilon) - \sigma \left\lfloor \sum_{i \in \epsilon} u_i + \mathcal{T}(L_j) \right\rfloor = \hat{c}(L_j + \epsilon) \end{aligned}$$

Hence, label L_i dominates label L_j. □

Proposition 6. *If* $\mathcal{T}(L_i) > \mathcal{T}(L_j)$, $\overline{v}(L_i) = \overline{v}(L_j)$, $\hat{c}(L_i) - \sigma \le \hat{c}(L_j)$, *and* $r(L_i) \le r(L_j)\ \forall r \in R$, *then label* L_i *dominates label* L_j.

Proof. Consider any common extension $\epsilon \in \mathcal{E}(L_j)$. Since $\mathcal{T}(L_i) > \mathcal{T}(L_j)$ the relation between the number of future penalties for the two labels when concatenated with ϵ is:

$$\left\lfloor \sum_{i \in \epsilon} u_i + \mathcal{T}(L_i) \right\rfloor \ge \left\lfloor \sum_{i \in \epsilon} u_i + \mathcal{T}(L_j) \right\rfloor \tag{44}$$

Since $0 \leq \mathcal{T}(L_j) < \mathcal{T}(L_i) < 1$ it is clear that the left hand side of (44) is at most one unit larger than the right hand side, i.e., label L_i will pay the penalty at most one more time than label L_j. Hence,

$$\left\lfloor \sum_{i \in \epsilon} u_i + \mathcal{T}(L_i) \right\rfloor - 1 \leq \left\lfloor \sum_{i \in \epsilon} u_i + \mathcal{T}(L_j) \right\rfloor$$

That is, the additional cost of extending L_i with ϵ is at most $-\sigma$ more than extending L_j with ϵ. This leads to the following relation between the costs:

$$\begin{aligned}
\hat{c}(L_i + \epsilon) &= \hat{c}(L_i) + \bar{c}(\epsilon) - \sigma \left\lfloor \sum_{i \in \epsilon} u_i + \mathcal{T}(L_i) \right\rfloor \\
&= \hat{c}(L_i) - \sigma + \bar{c}(\epsilon) - \sigma \left(\left\lfloor \sum_{i \in \epsilon} u_i + \mathcal{T}(L_i) \right\rfloor - 1 \right) \\
&\leq \hat{c}(L_j) + \bar{c}(\epsilon) - \sigma \left\lfloor \sum_{i \in \epsilon} u_i + \mathcal{T}(L_j) \right\rfloor \\
&= \hat{c}(L_j + \epsilon)
\end{aligned}$$

Hence label L_i dominates label L_j. □

Observe that if $\mathcal{T}(L_i) + \sum_{i \in \epsilon} u_i < 1$ for all $\epsilon \in \mathcal{E}(L_j)$, it is not possible to trigger a penalty, i.e., the temporary penalty to the cost of L_i can be disregarded.

In case of several CG rank-1 cuts, the new dominance criteria are as follows:

Proposition 7. *Let $Q = \{q : \sigma_q < 0 \wedge \mathcal{T}_q(L_i) > \mathcal{T}_q(L_j)\}$. Then label L_i dominates label L_j if:*

$$\bar{v}(L_i) = \bar{v}(L_j) \tag{45}$$

$$\hat{c}(L_i) - \sum_{q \in Q} \sigma_q \leq \hat{c}(L_j) \tag{46}$$

$$r(L_i) \leq r(L_j) \qquad \forall r \in R \tag{47}$$

Proof. The validity of (46) follows directly from Propositions 5 and 6. The validity of (45) and (47) follows from Proposition 4. □

5 Experimental Results

The experimental study is intended to show how much it is possible to strengthen the lower bound by adding CG rank-1 cuts, while still being able to solve the corresponding pricing problem in reasonable time. The SR inequalities have already proved their worth, see Jepsen et al. [23], Desaulniers et al. [9], but in both cases only sets of rows with size 3 were included, i.e., CG rank-1 cuts with precisely 3 non-zero entries in the CG multiplier vector. Hence, it is expected that the introduction of a separation routine for denser CG multiplier vectors could improve the lower bounds further. Using the exact separation routine for the CG rank-1 cuts is expected to be time consuming, but for test purposes it is interesting to see how well the column generation reacts to these cuts and also how much the lower bounds are improved.

5.1 Settings

The test instances are the well known benchmarks introduced by Solomon [33]. The benchmarks are divided into two series, both of which are again divided into a C (customers are grouped in larger clusters), an R (customers are distributed randomly), and an RC (a mix of the two previous) series. Of the 56 instances with 100 customers five instances are unsolved at the time of writing. Furthermore, 16 of the solved instances have not yet been solved in the root node of the branch-and-bound tree. We will only consider the R and RC instances, since all C instances can be solved in the root node without cutting planes, see Jepsen et al. [23], Desaulniers et al. [9].

The experiments were performed on a Pentium 4 3.0 GHz with 1 GB RAM. The basic BCP algorithm was developed with the framework COIN, see Lougee-Heimer [27]. The exact MIP-based CG rank-1 separation procedure is a slight modified version of a procedure provided by Hunsaker [19]. The MIPs were solved using CPLEX 9.1 with a maximal running time of 3600 seconds.

An exact separation procedure for a limited set of the SR inequalities have been developed exploiting the SSE2 vector-processing instructions intended for multimedia floating-point purposes which are present in all x86 processors since the introduction of Pentium 4 in 2001. The separation routine is an exact enumeration of SR inequalities with multipliers $u_i \in \{0, \frac{1}{k}, \dots, \frac{k-1}{k}\}$ for $i \in C$ where $\sum_{i \in C} u_i = \frac{n}{k}$, and $0 < k < n \leq |C|$ and k and n are integer, i.e., mod-k cuts with restriction on the sum of the multipliers.

Our implementation of the brute-force evaluation of all sub-multisets of rows of size n, can evaluate the SR inequalities (33) in constant time for each sub-multiset using the vector-processing capabilities. This makes it possible to separate all violated cuts in time $|S|^n/n!$ when $|P| \leq 16$, where S is the set of rows and P is the set of basis columns. Still, the complexity is so high that we cannot expect to separate inequalities with more than seven non-zero coefficients in reasonably time.

Note that our implementation of the BCP algorithm is not competitive with the recent implementation by Desaulniers et al. [9]. Also it is slower than the one used in Jepsen et al. [23] due to the implementation of the more general dominance criteria in the label algorithm. However, the point of our experiments is to study the quality of the lower bounds, i.e., the number of branch nodes, compared to the increase in computational time of the pricing problem by adding various cuts. These conclusions hold for all implementations based on the same decomposition.

5.2 Lower Bounds

Table 1 and 2 show the lower bounds obtained in the root node when different cutting policies are applied.

The cutting policies are:

"no"	No cutting planes
"$n = 3$"	SR cuts with $n = 3$ and $k = 2$
"$n \leq 5$"	Like option $n = 3$ and with $n = 5$ and $k = 2, 3$
"$n \leq 7$"	Like option $n \leq 5$ and with $n = 7$ and $k = 2, 3, 4$
"CG1"	General CG rank-1 cuts

A maximum of 50 cuts violating more than 0.0001 are added in each iteration. No time limit was imposed, but the space limit of 1 GB RAM prevented some instances to run to

Table 1: Lower bound comparison for the 1-series.

Instance	no	$n = 3$	$n \leq 5$	$n \leq 7$	CG1	UB
R101	1631.2	1634.0	1636.3	1636.3	**1637.7**	**1637.7**
R102	**1466.6**	**1466.6**	**1466.6**	**1466.6**	**1466.6**	**1466.6**
R103	1206.8	**1208.7**	**1208.7**	**1208.7**	**1208.7**	**1208.7**
R104	956.9	971.3	**971.5**	**971.5**	**971.5**	**971.5**
R105	1346.2	1355.2	**1355.3**	**1355.3**	**1355.3**	**1355.3**
R106	1227.0	**1234.6**	**1234.6**	**1234.6**	**1234.6**	**1234.6**
R107	1053.3	1064.3	**1064.6**	**1064.6**	**1064.6**	**1064.6**
R108	913.6	**932.1**	**932.1**	**932.1**	**932.1**	**932.1**
R109	1134.3	1144.1	1146.7	**1146.9**	**1146.9**	**1146.9**
R110	1055.6	**1068.0**	**1068.0**	**1068.0**	**1068.0**	**1068.0**
R111	1034.8	1045.9	1047.3	-	-	**1048.7**
R112	926.8	943.5	-	-	-	**948.6**
RC101	1584.1	**1619.8**	**1619.8**	**1619.8**	**1619.8**	**1619.8**
RC102	1406.3	**1457.4**	**1457.4**	**1457.4**	**1457.4**	**1457.4**
RC103	1225.6	1257.7	**1258.0**	**1258.0**	**1258.0**	**1258.0**
RC104	1101.9	1129.9	-	-	-	**1132.3**
RC105	1472.0	**1513.7**	**1513.7**	**1513.7**	**1513.7**	**1513.7**
RC106	1318.8	1367.3	1371.9	**1372.7**	**1372.7**	**1372.7**
RC107	1183.4	**1207.8**	**1207.8**	**1207.8**	**1207.8**	**1207.8**
RC108	1073.5	**1114.2**	**1114.2**	**1114.2**	**1114.2**	**1114.2**

completion.

Upper bounds in the "UB" column are optimal values or best known upper bounds. Entries in tables marked with an asterisk * are from Danna and Le Pape [7], entries marked with a double-asterisk ** are from Desaulniers et al. [9], and entries marked with a triple-asterisk *** are from Jepsen et al. [23]. A dash indicates that our implementation failed due to memory limitation. Entries in bold face indicate optimal integer solution.

Of the 28 solved instances one instance (R102) was solved without adding any cuts. The lower bounds for all remaining instances were considerably improved by adding "$n = 3$" cuts resulting in integer solutions for 15 of the 27 remaining (17 out of 33 when considering the results of Desaulniers et al. [9]). When adding "$n \leq 5$" cuts improvements were present in all but one instance (RC201) resulting in further five integer solutions of the 10 remaining instances that could be solved with this approach. Of the remaining four instances solved with "$n \leq 7$" cuts, two showed no improvement and two resulted in integer solutions. The last two instances were solved to integrality when applying CG rank-1 cuts. Hence, we succeeded in closing the gap between the upper and lower bound for all the instances that we were able to solve within the memory limit.

Tables 1 and 2 also show that the SR inequalities provide almost as good lower bounds as general CG rank-1 cuts. For "$n = 7$" the SR inequalities become time consuming to separate, and hence in practical applications one should confine to "$n = 3$" or "$n \leq 5$".

Table 3 presents an overview of problems solved in the root node as reported in this chapter or by Jepsen et al. [23] or Desaulniers et al. [9]. Column "solved" refers to the number of instances solved to optimality at the time of writing and "total" refers to the total number of instances. Results from the C-series are included for completeness.

As already noted, adding SR inequalities and CG rank-1 cuts greatly strengthens the

Table 2: Lower bound comparison for the 2-series.

Instance	no	$n = 3$	$n \leq 5$	$n \leq 7$	CG1	UB
R201	1140.3	**1143.2**	**1143.2**	**1143.2**	**1143.2**	**1143.2**
R202	1022.3	1027.3	**1029.6**	**1029.6**	**1029.6**	**1029.6**
R203	867.0	**870.8**	**870.8**	**870.8**	**870.8**	**870.8**
R204	-	-	-	-	-	**731.3
R205	939.0	-	-	-	-	**949.8**
R206	866.9	****875.9**	-	-	-	**875.9**
R207	**790.7	****794.0**	-	-	-	**794.0**
R208	-	-	-	-	-	*701.2
R209	841.5	***854.8	-	-	-	**854.8**
R210	889.4	-	-	-	-	**900.5**
R211	-	-	-	-	-	**746.7
RC201	1256.0	1261.7	1261.7	1261.7	**1261.8**	**1261.8**
RC202	1088.1	**1092.3**	**1092.3**	**1092.3**	**1092.3**	**1092.3**
RC203	922.6	**923.7**	**923.7**	**923.7**	**923.7**	**923.7**
RC204	-	-	-	-	-	*783.5
RC205	1147.7	**1154.0**	**1154.0**	**1154.0**	**1154.0**	**1154.0**
RC206	1038.6	**1051.1**	**1051.1**	**1051.1**	**1051.1**	**1051.1**
RC207	947.4	-	-	-	-	**962.9**
RC208	-	-	-	-	-	**776.5

Table 3: Summary of instances solved in the root node.

Instance	no	$n = 3$	$n \leq 5$	$n \leq 7$	CG1	solved	total
C1	9	9	9	9	9	9	9
C2	8	8	8	8	8	8	8
R1	1	5	8	9	10	12	12
R2	0	4	5	5	5	8	11
RC1	0	5	6	7	7	8	8
RC2	0	4	4	4	5	6	8
All	18	35	40	42	44	51	56

lower bounds. Of the 56 instances 35 were previously reported solved in the root node by Jepsen et al. [23], Desaulniers et al. [9]. With our additional cutting planes we were able to solve an additional nine instances in the root node of the remaining 16 previously solved instances. Note that all the instances we were able to solve were solved in the root node. The remaining seven instances, which have previously been solved with "$n = 3$", could not be solved with the current implementation due to hardware limitations. Hence, there exists 12 Solomon instances (seven solved with branching and five unsolved) where CG rank-1 cuts could potentially improve the lower bound in the root node.

5.3 Running Times of the Pricing Problem

Table 4 and 5 contain the results obtained when solving the instances to optimality using different cutting planes. In column "CPU" we report the CPU-time in seconds for solving the last pricing problem, while column "cuts" gives the number of cuts applied. Column "BB" indicates the number of branch-and-bound nodes considered. As before, a dash in the tables indicates that a memory insufficiency had occurred. Entries marked with a double-asterisk ** are from Desaulniers et al. [9].

Table 4: Running time for pricing problem and number of branch-and-bound nodes for the 1-series. 1) Data log-files were lost during machine upgrade.

	no		$n = 3$			$n \leq 5$			CG1		
Instance	CPU	BB	CPU	cuts	BB	CPU	cuts	BB	CPU	cuts	BB
R101	0.1	11	0.1	2	3	0.1	4	3	0.1	15	1
R102	0.2	1	0.2	0	1	0.2	0	1	0.2	0	1
R103	0.4	15	1.3	33	1	1.3	33	1	1.3	33	1
R104	5.8	-	910.5	328	3	-	-	1^1	-	-	1
R105	0.1	55	0.2	52	3	0.2	56	1	0.2	56	1
R106	0.5	147	4.8	114	1	4.8	114	1	4.8	114	1
R107	2.2	-	46.1	224	3	78.4	242	1	78.4	242	1
R108	13.0	-	244.8	192	1	244.8	192	1	244.8	192	1
R109	0.3	-	1.6	127	17	8.7	374	3	10.0	367	1
R110	1.1	-	26.0	269	1	26.0	269	1	26.0	269	1
R111	1.5	-	36.6	175	39	293.7	379	-	-	-	-
R112	35.9	-	-	-	9^1	-	-	-	-	-	-
RC101	0.1	59	0.2	69	1	0.2	69	1	0.2	69	1
RC102	0.3	-	1.4	140	1	1.4	140	1	1.4	140	1
RC103	1.2	-	42.8	276	3	49.1	290	1	49.1	290	1
RC104	15.6	-	569.2	237	7	-	-	-	-	-	-
RC105	0.2	191	0.5	73	1	0.5	73	1	0.5	73	1
RC106	0.3	-	3.5	250	37	16.5	543	5	21.6	572	1
RC107	1.4	-	4.3	85	1	4.3	85	1	4.3	85	1
RC108	9.7	-	86.7	175	1	86.7	175	1	86.7	175	1

The tables show that adding "$n \leq 5$" cuts and "CG1" cuts is relatively cheap with respect to the running time of the pricing problem, while decreasing the number of branch-and-bound nodes significantly e.g., in instances R109, RC106, and R202.

If we had access to "ideal" heuristics for the pricing problem (with low running time and high solution quality) we would only need to solve one pricing problem to optimality in each branch-and-bound node. The running time of the overall algorithm would then be dictated by the running time for optimally solving the pricing (CPU) and the number of branch-and-bound nodes (BB). With the exception of R202 (where massive paging occurred due to lack of memory) the lower bound on the running time "BB $\times$ CPU" is not increasing when n grows and "CG1" cuts are applied. This shows, that good heuristics for the pricing problem can make the addition of SR and CG-1 cuts attractive for the overall running time.

6 Concluding Remarks

We have demonstrated that it is possible to apply general CG rank-1 cuts derived from the master problem formulation in a BCP algorithm for VRPTW. As each cut results in the introduction of a new resource in the pricing problem it was necessary to develop new, tighter dominance rules for use in the pricing algorithm.

Our computational experiments indicate that the addition of CG rank-1 cuts leads to significantly improved lower bounds. In our tests the cuts made it possible to close the gap

Table 5: Running time for pricing problem and number of branch-and-bound nodes for the 2-series.

	no		$n = 3$			$n \leq 5$			CG1		
Instance	CPU	BB	CPU	cuts	BB	CPU	cuts	BB	CPU	cuts	BB
R201	0.2	-	0.4	15	1	0.4	15	1	0.4	15	1
R202	2.9	-	3.0	24	13	419.6	132	1	419.6	132	1
R203	83.2	-	505.6	35	1	505.6	35	1	505.6	35	1
R204	-	-	-	-	-	-	-	-	-	-	-
R205	1.5	-	-	**345	**9	-	-	-	-	-	-
R206	131.7	-	-	**171	**1	-	-	-	-	-	-
R207	-	-	-	**24	**1	-	-	-	-	-	-
R208	-	-	-	-	-	-	-	-	-	-	-
R209	6.5	-	-	**248	**3	-	-	-	-	-	-
R210	-	-	-	**266	**5	-	-	-	-	-	-
R211	-	-	-	-	-	-	-	-	-	-	-
RC201	0.2	-	0.3	25	3	0.3	25	3	0.3	29	1
RC202	0.6	-	1.7	26	1	1.7	26	1	1.7	26	1
RC203	58.8	11	185.2	15	1	185.2	15	1	185.2	15	1
RC204	-	-	-	-	-	-	-	-	-	-	-
RC205	1.0	-	1.8	21	1	1.8	21	1	1.8	21	1
RC206	1.7	-	4.6	23	1	4.6	23	1	4.6	23	1
RC207	-	-	-	**210	**5	-	-	-	-	-	-
RC208	-	-	-	-	-	-	-	-	-	-	-

between the upper and lower bounds in the root node of the branch-and-bound tree for 44 of the 51 currently solvable instances from Solomon's test library. This is an additional 9 instances compared to previous results. The increased complexity of the pricing problem caused by CG rank-1 cuts do affect the running time of the pricing problems but not significantly.

This indicates that CG rank-1 inequalities may be essential when solving larger instances to optimality, as one cannot expect that the branching process will close the gap between the upper and lower bound in reasonable time. Note that one should also take into account the additional time spent in each branch node since the number of LP iterations increases when valid inequalities are added. As for classical branch-and-cut algorithms it will always be a question when to add cuts and when to start branching.

Another important note is the separation time of the CG rank-1 cuts which can indeed be very time consuming. Also the current MIP-based heuristics only finds a limited number of violated cuts as the main effort is put in cut violation quality not violated cut quantity. We suggest that MIP-based heuristics which focus on finding numerous violated CG rank-1 cuts could improve the performance of the BCP algorithm. Fortunately the SR inequalities generally give rise to almost as tight lower bounds as general CG rank-1 cuts, while being easier to handle in the pricing problem (due to integer modulo operations, see Jepsen et al. [23]). For $n = 7$ the separation of SR inequalities takes almost one hour, making them too expensive to separate. For $n \leq 5$ the inequalities can be separated in a couple of minutes. So until more efficient separation methods have been developed, one should only apply SR inequalities for $n \leq 5$.

During our experiments we noticed that specific values of the CG multipliers u occurred more frequently than others. For instance, multiplier vectors $\mathbf{u} \in \{0, \frac{1}{2}\}^{|C|}$ occurred very frequently, showing that it is promising to investigate these inequalities further (note that the SR inequalities for a given n with $k = 2$ are a subset of these inequalities). Knowing the structure of promising CG rank-1 inequalities will make it possible to develop fast, specialized separation heuristics and better handling of these specific inequalities in the pricing problem. Adapting the separation algorithm by Caprara et al. [3] for maximally violated mod-k cuts in the master problem could be an interesting direction of research.

References

[1] R. E. Bixby, S. Ceria, C. M. McZeal, and M. W. P Savelsbergh. An updated mixed integer programming library: MIPLIB 3.0. *Optima*, 58:12–15, 1998.

[2] N. Boland, J. Dethridge, and I. Dumitrescu. Accelerated label setting algorithms for the elementary resource constrained shortest path problem. *Operation Research Letters*, 34 (1):58–68, 2006. doi: 10.1016/j.orl.2004.11.011.

[3] A. Caprara, M. Fischetti, and A. N. Letchford. On the separation of maximally violated mod-k cuts. *Mathematical Programming*, 87(A):37–56, 1999. doi: 10.1007/s101079900107.

[4] A. Chabrier. Vehicle routing problem with elementary shortest path based column generation. *Computers & Operations Research*, 33(10):2972–2990, 2006. doi: 10.1016/j.cor.2005.02.029.

[5] V. Chvatal. Edmonds polytopes and hierarchy of combinatorial problems. *Discrete Mathematics*, 4(4):305–337, 1973. doi: 10.1016/0012-365X(73)90167-2.

[6] W. Cook and J. L. Rich. A parallel cutting plane algorithm for the vehicle routing problem with time windows. Technical Report TR99-04, Computational and Applied Mathematics, Rice University, Houston, Texas, USA, 1999.

[7] E. Danna and C. Le Pape. Branch-and-price heuristics: A case study on the vehicle routing problem with time windows. In G. Desaulniers, J. Desrosiers, and M. M. Solomon, editors, *Column Generation*, chapter 4, pages 99–129. Springer, 2005. doi: 10.1007/0-387-25486-2_4.

[8] G. Desaulniers, J. Desrosiers, J. Ioachim, I. M. Solomon, F. Soumis, and D. Villeneuve. A unified framework for deterministic time constrained vehicle routing and crew scheduling problems. In T. G. Crainic and G. Laporte, editors, *Fleet Management and Logistics*, pages 57–93. Kluwer, 1998.

[9] G. Desaulniers, F. Lessard, and A. Hadjar. Tabu search, partial elementarity, and generalized k-path inequalities for the vehicle routing problem with time windows. *Transportation Science*, 42(3):387–404, 2008. doi: 10.1287/trsc.1070.0223.

[10] M. Desrochers, J. Desrosiers, and M. Solomon. A new optimization algorithm for the vehicle routing problem with time windows. *Operations Research*, 40(2):342–354, 1992. doi: 10.1287/opre.40.2.342.

[11] M. Dror. Note on the complexity of the shortest path models for column generation in VRPTW. *Operations Research*, 42(5):977–979, 1994. doi: 10.1287/opre.42.5.977.

[12] I. Dumitrescu. *Constrained Path and Cycle Problems.* PhD thesis, Department of Mathematics and Statistics, University of Melbourne, Australia, 2002.

[13] F. Eisenbrand. Note - on the membership problem for the elementary closure of a polyhedron. *Combinatorica*, 19(2):297–300, 1999. doi: 10.1007/s004930050057.

[14] D. Feillet, P. Dejax, M. Gendreau, and C. Gueguen. An exact algorithm for the elementary shortest path problem with resource constraints: Application to some vehicle routing problems. *Networks*, 44(3):216–229, 2004. doi: 10.1002/net.v44:3.

[15] M. Fischetti and A. Lodi. Optimizing over the first Chvátal closure. *Mathematical Programming*, 110(1):3–20, 2006. doi: 10.1007/s10107-006-0054-8.

[16] R. Fukasawa, H. Longo, J. Lysgaard, M. Poggi de Aragão, M. Reis, E. Uchoa, and R.F. Werneck. Robust branch-and-cut-and-price for the capacitated vehicle routing problem. *Mathematical Programming*, 106(3):491–511, 2006. doi: 10.1007/s10107-005-0644-x.

[17] R.E. Gomory. Outline of an algorithm for integer solutions to linear programs. *Bulletin of the AMS*, 64:275–278, 1958. doi: 10.1090/S0002-9904-1958-10224-4.

[18] R.E. Gomory. An algorithm for integer solutions to linear programs. In R.L. Graves and P. Wolfe, editors, *Recent Advances in Mathematical Programming*, pages 269–302. McGraw-Hill, New York, 1963.

[19] B. Hunsaker. Cg-rank. http://www.rosemaryroad.org/brady/cg-rank/index.html, 2005.

[20] S. Irnich. Resource extension functions: Properties, inversion, and generalization to segments. *OR Spectrum*, 30(1):113–148, 2008. doi: 10.1007/s00291-007-0083-6.

[21] S. Irnich and G. Desaulniers. Shortest path problems with resource constraints. In G. Desaulniers, Jacques Desrosiers, and M.M. Solomon, editors, *Column Generation*, chapter 2, pages 33–65. Springer, 2005. doi: 10.1007/0-387-25486-2_2.

[22] S. Irnich and D. Villeneuve. The shortest path problem with resource constraints and k-cycle elimination for $k \geq 3$. *INFORMS Journal on Computing*, 18:391–406, 2006. doi: 10.1287/ijoc.1040.0117.

[23] M. 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. doi: 10.1287/opre.1070.0449.

[24] B. Kallehauge, J. Larsen, and O.B.G. Madsen. Lagrangean duality and non-differentiable optimization applied on routing with time windows - experimental results. Technical Report Internal report IMM-REP-2000-8, Department of Mathematical Modelling, Technical University of Denmark, Lyngby, Denmark, 2000.

[25] N. Kohl, J. Desrosiers, O. B. G. Madsen, M. M. Solomon, and F. Soumis. 2-path cuts for the vehicle routing problem with time windows. *Transportation Science*, 33(1):101–116, 1999. doi: 10.1287/trsc.33.1.101.

[26] J. Larsen. *Parallelization of the vehicle routing problem with time windows.* PhD thesis, Department of Mathematical Modelling, Technical University of Denmark, Lyngby, Denmark, 1999.

[27] Robin Lougee-Heimer. The Common Optimization INterface for Operations Research: Promoting open-source software in the operations research community. *IBM Journal of Research and Development*, 47(1):57–66, 2003. doi: 10.1147/rd.471.0057.

[28] G. Nemhauser and S. Park. A polyhedral approach to edge coloring. *Operations Research Letters*, 10(6):315–322, 1991. doi: 10.1016/0167-6377(91)90003-8.

[29] G. Righini and M. Salani. Symmetry helps: bounded bi-directional dynamic programming for the elementary shortest path problem with resource constraints. *Discrete Optimization*, 3(3):255–273, 2006. doi: 10.1016/j.disopt.2006.05.007.

[30] G. Righini and M. Salani. New dynamic programming algorithms for the resource constrained shortest path problem. *Networks*, 51(3):155–170., 2008. doi: 10.1002/net.20212.

[31] M. Salani. *Branch-and-Price Algorithms for Vehicle Routing Problems.* PhD thesis, Universitá Degli Studi Di Milano, Facoltá di Scienza Matematiche, Fisuche e Naturali Dipartimento di Technologie dell'Informazione, Milano, Italy, 2005.

[32] A. Schrijver. On cutting planes. *Annals of Discrete Mathematics*, 9:291–296, 1980. doi: 10.1016/S0167-5060(08)70085-2.

[33] M. M. Solomon. Algorithms for the vehicle routing and scheduling problem with time window constraints. *Operations Research*, 35(2):234–265, 1987. doi: 10.1287/opre.35.2.254.

[34] P. Toth and D. Vigo. An overview of vehicle routing problems. In P. Toth and D. Vigo, editors, *The Vehicle Routing Problem*, chapter 1, pages 1–26. SIAM, 2002.

Chapter 5

Clique Inequalities applied to the Vehicle Routing Problem with Time Windows

Simon Spoorendonk
DIKU Department of Computer Science, University of Copenhagen

Guy Desaulniers
École Polytechnique and GERAD, Montreal, Canada

Abstract

This work presents an exact branch-and-cut-and-price algorithm for the vehicle routing problem with time windows (VRPTW) where the well-known clique inequalities are used as cutting planes defined on the set partitioning master problem variables. It shows how these cutting planes affect the dominance criterion applied in the pricing algorithm, which is a labeling algorithm for solving resource-constrained elementary shortest path problems. The idea of using cutting planes defined on the master problem variables has been recently developed: Chvátal-Gomory rank 1 cuts were applied for the VRPTW. However, to our knowledge, this is a first attempt at incorporating for the VRPTW a set of valid inequalities specialized for the set partitioning polytope. Computational results show that the use of clique inequalities improves the lower bound at the root node of the search tree and reduces the number of nodes in this tree.

1 Introduction

Given an unlimited number of identical vehicles with a specified capacity, and a set of customers, each with a demand to fulfill and a delivery time window, the vehicle routing problem with time windows (VRPTW) consists of finding a set of routes starting and ending at a central depot such that each customer is visited exactly once within its time window, the capacity

Submitted for publication, 2008

of each vehicle is respected, and the total length of the routes is minimized. The VRPTW can be modeled using an arc-flow formulation. Applying the Dantzig-Wolfe decomposition principle on this formulation decomposes the problem into a master problem (a set partitioning problem) and a pricing problem that corresponds to an elementary shortest path problem with resource constraints (ESPPRC), where capacity and time are the constrained resources. Column generation is then used to solve the LP-relaxed master problem and provides a lower bound. Such lower bounds are exploited in a branch-and-bound algorithm to reach integrality. The resulting method is called branch-and-price (BP) or branch-and-cut-and-price (BCP) when cutting planes are also generated.

As shown by the large number of references on exact methods in the survey of Kallehauge et al. [16] and the recent paper of Desaulniers et al. [7], the VRPTW has been extensively studied in the literature. Nevertheless, the most successful exact methods still struggle at solving VRPTW instances of medium size (i.e., involving around 100 customers) when the time windows are relatively wide compared to the average travel time between two customer locations. Consequently, research on exact methods for this problem is still very active.

Since the seminal work of Desrochers et al. [8] in 1992, BP and BCP algorithms have been the leading methodologies for solving the VRPTW. Because the ESPPRC is strongly $\mathcal{NP}$-hard, Desrochers et al. [8] proposed a BP algorithm where the pricing problem is a shortest path problem with resource constraints (SPPRC) that allows cycles (except cycles involving two customers, called 2-cycles) and, thus, yields a relaxation and possibly large integrality gaps. Two research trends have been followed to reduce these gaps. In the first trend, the pricing problem is strengthened. Irnich and Villeneuve [14] developed a k-cycle elimination procedure for the SPPRC for arbitrary integer values of k that forbids the generation of routes containing at least one ℓ-cycle with $\ell \in \{2, 3, \dots, k\}$, where an ℓ-cycle is a subpath composed of ℓ arcs that starts and ends at the same node. In parallel, several researchers devised algorithms for solving the ESPPRC. Feillet et al. [11] were the first to propose a successful label-setting algorithm for the ESPPRC where one resource is required for each customer to impose elementarity. Several improvements of this algorithm were proposed by Chabrier [4], Boland et al. [3], and Righini and Salani [21, 22]. As reported in Desaulniers et al. [7], the use of the ESPPRC as a pricing problem significantly reduces the integrality gap for most instances. However, the ESPPRC remains very difficult to solve and, for some medium-sized instances with wide time windows, it seems impossible to solve the LP-relaxed master problem in a reasonable time.

In the second research trend, cutting planes are generated to tighten the LP-relaxed master problem. Kohl et al. [17] introduced the k-path cuts that force a total flow of at least k vehicles into subsets of customers that cannot be serviced by less than k vehicles. Such cuts can be defined using arc-flow variables and rewritten in terms of the master problem variables after applying the Dantzig-Wolfe decomposition principle. Therefore, their treatment only implies a modification of the arc costs in the pricing problem. Recently, Jepsen et al. [15] proposed subset row inequalities that are directly defined on the master problem variables. These inequalities form a subset of the Chvátal-Gomory rank 1 inequalities for the set partitioning polytope. Later, Petersen et al. [20] generalized this work to include all Chvátal-Gomory rank 1 cuts. These inequalities significantly reduce integrality gaps. However, they complexify the solution of the pricing problem because each cut requires one additional resource and the standard dominance procedure to identify Pareto-optimal labels in the labeling algorithm cannot be used. In a very recent paper, Baldacci et al. [1] used a set partitioning formulation for the capacitated vehicle routing problem and applied clique inequalities to strengthen its linear relaxation. Their solution method consists of first eliminating a large number of variables using a reduced cost criterion before solving the resulting model with a commercial integer programming solver. The application of this criterion requires the a priori computation of a dual solution for the linear relaxation. To do so, they use an additive bounding method that solves in its last step the linear relaxation of a modified set partitioning model using column generation. Based on the observation that the dual values of the clique inequalities are non-positive, the pricing problem is solved by an extensive enumeration scheme based on dynamic programming that disregards these dual values but ensures that all negative reduced cost columns can be found. This procedure performs very well on small, tightly constrained instances, but has memory issues when the number of feasible candidate paths in the pricing problem is too large.

In this paper, we follow this second research trend by studying the use of clique inequalities defined on the master problem variables in the context of the VRPTW. As opposed to the work of Baldacci et al. [1], we propose to consider the dual values of the clique inequalities directly into the labeling algorithm to avoid an extensive enumeration of candidate paths. We believe that such an alternative to extensive enumeration must be sought to derive a scalable BCP algorithm that can solve the most difficult VRPTW instances (e.g., those allowing more than 50 customers per route). To handle the dual values of these inequalities in the labeling algorithm, we introduce an approximate representation of a clique and develop a modified dominance criterion. To assess the usefulness of the clique cuts, we performed computational experiments on a subset of the well-known VRPTW instances of Solomon [24]. The results show that combining the clique inequalities with the subset row inequalities improves the lower bounds and reduces the number of nodes in the search tree.

The paper is organized as follows. In Section 2, the VRPTW is formulated mathematically while, in Section 3, a basic BCP algorithm is described. Section 4 provides an overview on the separation of the clique inequalities and how they may be applied in a BCP algorithm for the VRPTW when the pricing problem is an ESPPRC solved by a labeling algorithm. Computational results are reported in Section 5, and concluding remarks are made in Section 6.

2 Problem Formulations

In this section, we present two formulations for the VRPTW, namely, an arc-flow formulation and a set partitioning formulation. Let C be the set of customers and K the set of available vehicles such that $|K|$ is sufficiently large for not imposing a constraint on the number of vehicles that can be used in a solution (e.g., $|K| = |C|$). The arc-flow formulation is defined on a network $G = (N, A)$, where N and A are the node and arc sets, respectively. The node set N contains one node for each customer $i \in C$ and two nodes, o and o', representing the depot at the start and the end of the routes, respectively. Thus, $N = C \cup \{o, o'\}$. Each customer $i \in C$ has a demand d_i, an associated service time s_i, and a time window $[a_i, b_i]$ in which it should be visited (i.e., service must start within this interval). To simplify the notation, we set $d_o = d_{o'} = s_o = s_{o'} = 0$ and $[a_o, b_o] = [a_{o'}, b_{o'}] = [0, H]$, where H is the length of the planning horizon. The arc set A is given by $A = \{(i,j) : i,j \in N,\ i \neq j, a_i + \tau_{ij} \leq b_j, d_i + d_j \leq D\} \setminus \{(o', o)\}$, where D is the vehicle capacity and τ_{ij} is the travel time between locations i and j plus the service time s_i at i. In this set definition, the last two conditions ensure that the same vehicle can visit consecutively locations i and j according to the time and load restrictions. Denote by c_{ij} the travel cost along arc $(i,j) \in A$.

Two types of variables are used. The binary variable x_{ijk} indicates whether or not vehicle $k \in K$ traverses arc $(i,j) \in A$. The continuous variable t_{ik} provides the start of service time at node $i \in N$ if vehicle $k \in K$ visits node i. Its value is irrelevant otherwise.

Using this notation as well as $\delta^-(i) = \{j \in N \,|\, (j,i) \in A\}$ and $\delta^+(i) = \{j \in N \,|\, (i,j) \in A\}$, the arc-flow model for the VRPTW is given by:

$$\min \sum_{k \in K} \sum_{(i,j) \in A} c_{ij} x_{ijk} \tag{1}$$

$$\text{s.t.} \sum_{k \in K} \sum_{j \in \delta^+(i)} x_{ijk} = 1 \qquad \forall i \in C \tag{2}$$

$$\sum_{j \in \delta^+(o)} x_{ojk} = \sum_{i \in \delta^-(o')} x_{io'k} = 1 \qquad \forall k \in K \tag{3}$$

$$\sum_{j \in \delta^-(i)} x_{jik} - \sum_{j \in \delta^+(i)} x_{ijk} = 0 \qquad \forall i \in C,\ \forall k \in K \tag{4}$$

$$\sum_{(i,j) \in A} d_i x_{ijk} \leq D \qquad \forall k \in K \tag{5}$$

$$x_{ijk}(t_{ik} + \tau_{ij} - t_{jk}) \leq 0 \qquad \forall (i,j) \in A,\ \forall k \in K \tag{6}$$

$$a_i \leq t_{ik} \leq b_i \qquad \forall i \in N,\ \forall k \in K \tag{7}$$

$$x_{ijk} \in \{0,1\} \qquad \forall (i,j) \in A,\ \forall k \in K. \tag{8}$$

The objective function (1) aims at minimizing the total travel costs. Constraints (2) specify that every customer $i \in C$ must be visited by exactly one vehicle. Constraints (3) and flow conservation constraints (4) define a path structure between o and o' for each vehicle k. Constraints (3) also guarantee that each route starts and ends at the depot. Note that a zero-cost variable $x_{oo'k}$ is considered for each vehicle to allow an empty route in the case where not all $|K|$ available vehicles are needed. Constraint set (5) ensures that the capacity of each vehicle is not exceeded. Constraints (6) and (7) impose the time window restrictions. Note that (6) together with the assumption that $\tau_{ij} > 0$ for all $(i,j) \in A \setminus \{o, o'\}$ eliminates all sub-tours. The last constraints (8) define the domain of the arc-flow variables.

Keeping (2) as the linking constraints and applying the Dantzig-Wolfe decomposition principle (Dantzig and Wolfe [5]) on the arc-flow model (1)-(8) leads to the following set partitioning formulation of the VRPTW:

$$\min \sum_{p \in P} \Big(\sum_{(i,j) \in p} c_{ij} \Big) \lambda_p \tag{9}$$

$$\text{s.t.} \quad \sum_{p \in P} \alpha_{ip} \lambda_p = 1 \qquad \forall i \in C \tag{10}$$

$$\lambda_p \geq 0 \qquad \forall p \in P \tag{11}$$

$$\lambda_p \ \text{ binary} \qquad \forall p \in P \tag{12}$$

where P is the set of all feasible $o - o'$ paths (routes) in G, the binary parameter α_{ip} is equal to one if and only if node $i \in N$ is used in route $p \in P$, and the binary variable λ_p indicates whether or not route $p \in P$ is selected in the solution. Set partitioning constraints (10) ensure that each customer $i \in C$ is visited by exactly one vehicle. The definition of set P considers all the other problem constraints that define the feasibility of a route. In general, for a practical VRPTW instance, this set partitioning model contains a huge number of variables and can be solved using a BCP method.

3 Branch-and-Cut-and-Price

As mentioned in the introduction, a BCP method consists of a column generation method embedded in a branch-and-cut method (see Barnhart et al. [2], Desaulniers et al. [6], Lübbecke and Desrosiers [18]). In this context, model (9)-(12) is called the *integer master problem* and its linear relaxation (9)-(11) the *master problem*. Column generation is an iterative method for solving the master problem. At each iteration, it solves a so-called *restricted master problem* (RMP) and a *pricing problem*. The RMP is simply the master problem restricted to a subset of its variables λ_p. The size of this subset increases at each iteration. The RMP is thus a linear program that can be solved by the simplex algorithm. The pricing problem plays the role of determining which columns (variables) should be added to the RMP at each iteration. It aims at finding negative reduced cost columns. When no such columns exist, the current RMP solution is optimal for the master problem and the column generation method stops.

For the VRPTW, each variable λ_p, $p \in P$, is associated with a feasible route in G. The reduced cost $\bar{c}_p$ of such a variable is:

$$\bar{c}_p = \sum_{(i,j) \in p} c_{ij} - \sum_{(i,j) \in p} \pi_j = \sum_{(i,j) \in p} (c_{ij} - \pi_j) \tag{13}$$

where $\pi_j \in \mathbb{R}$ for all $j \in C$ are the dual values of (10) and $\pi_{o'} = 0$. To determine if there exists negative reduced cost variables, one can find the shortest feasible path from o to o' in G, where the cost of each arc $(i,j) \in A$ is $\bar{c}_{ij} = c_{ij} - \pi_j$. Feasibility is enforced by resource constraints (see Irnich and Desaulniers [13]). A resource is a quantity that accumulates along a path and is restricted at each node to take a value within a prespecified resource interval, called a *resource window*. For the VRPTW, the set of resources $\mathcal{R}$ contains a loading resource ($r = load$) to ensure that vehicle capacity is satisfied, a time resource ($r = time$) to respect the customer time windows, and a binary resource $r = cust_i$) for each customer $i \in C$ to ensure path elementarity.

The pricing problem thus corresponds to an ESPPRC. If its optimal value is negative at a given iteration, then the computed shortest path provides a negative reduced cost path variable that can be added to the RMP before starting a new iteration. Additional negative reduced cost paths (when also computed while solving the pricing problem) can also be used to generate more than one column. Otherwise, the column generation algorithm stops.

As shown by Dror [9], the ESPPRC is strongly $\mathcal{NP}$-hard when considering loading and time resources. It can be solved using a labeling algorithm such as those proposed by Feillet et al. [11] and Chabrier [4]. In such a dynamic programming method, labels represent partial paths that are extended (using so-called *extension functions*) in all feasible directions from the source node o. Each label L (a vector with $\mathcal{R}+1$ components) stores the cost of the partial path $T_{cost}(L)$ and the current value $T_r(L)$ of each resource $r \in \mathcal{R}$. For the VRPTW, the extension along an arc (i,j) of a label L representing a partial path ending at node i proceeds as follows to create a new label L' representing a partial path ending by the arc (i,j):

$$T_{cost}(L') = T_{cost}(L) + c_{ij} - \pi_j \tag{14}$$

$$T_{load}(L') = T_{load}(L) + d_j \tag{15}$$

$$T_{time}(L') = \max\{T_{time}(L) + \tau_{ij}, a_j\} \tag{16}$$

$$T_{cust_\ell}(L') = \begin{cases} T_{cust_j}(L) + 1 & \text{if } \ell = j \\ \max\{T_{cust_\ell}(L), U_\ell(L')\} & \text{otherwise} \end{cases} \qquad \forall \ell \in C, \tag{17}$$

where $U_\ell(L')$ indicates whether or not there exist no feasible extensions of label L' reaching node $\ell \in C$ (ℓ is unreachable from L). Assuming that the arc durations satisfy the triangle inequality, $U_\ell(L')$ is equal to 1 if $T_{load}(L') + d_\ell > D$ or $T_{time}(L') + \tau_{j\ell} > b_\ell$, and 0 otherwise. Label L' corresponds to a feasible path if it respects the resource windows, that is, if $T_{load}(L') \in [0, D]$, $T_{time}(L') \in [a_j, b_j]$, and $T_{cust_\ell}(L') \in [0,1]$ for all $\ell \in C$. It is discarded if this is not the case.

To avoid enumerating all feasible paths in G, only Pareto-optimal labels (i.e., labels that are not proven to be dominated by other labels) are kept during the execution of the algorithm. When using non-decreasing extension functions as it is the case for the VRPTW, the label dominance criterion can be stated as follows.

Proposition 1 (Desaulniers et al. [6])**.** Let L and L' be two labels representing partial paths ending at the same node. Label L dominates label L' (which can be discarded) if

$$T_{cost}(L) \leq T_{cost}(L') \tag{18}$$

$$T_r(L) \leq T_r(L') \qquad \forall r \in \mathcal{R}. \tag{19}$$

When equality holds for all label components, one of the two labels must be kept.

Recently, efficient bidirectional labeling algorithms have been proposed by Righini and Salani [21, 22]. These algorithms extend *forward* labels from the source node o and *backward* labels from the sink node o' before joining forward and backward labels to create feasible $o-o'$ paths. In this context, the extension of the backward labels requires backward extension functions that are different from the forward ones (14)-(17). Furthermore, dominance applies only between forward labels or between backward labels.

In a BCP method, cutting planes can be added to tighten the master problem when its computed solution is fractional. When no violated valid inequalities can be identified, branching decisions are imposed, yielding a branch-and-bound search tree. Note that cutting planes can be added throughout this tree. Note also that after adding cutting planes or imposing a branching decision, the master problem and the pricing problem are updated in consequence, and column generation is again performed to re-optimize the modified master problem and compute a new lower bound.

Different types of branching decisions can be imposed either in the master problem or in the pricing problem. In our implementation, we branch (as in Desrochers et al. [8], Desaulniers et al. [7]) on the total flow on an arc $(i, j) \in A$ (with $i \neq o$ and $j \neq o'$) which must be either 0 or 1 in an integer solution. For imposing a flow of 0 on arc (i, j), we simply remove (i, j) from A in the pricing problem, while for imposing a flow of 1 on this arc, we remove all arcs $(i, j') \in A$ such that $j' \neq j$ and all arcs $(i', j) \in A$ such that $i' \neq i$. For both decisions, we remove from the current RMP all columns λ_p such that path p contains a removed arc.

Various families of valid inequalities can also be considered for the VRPTW. In the literature, most of them, such as the 2-path cuts proposed by Kohl et al. [17], can be defined as linear combinations of the arc-flow variables x_{ijk} and rewritten in terms of the master problem variables λ_p. In this case, their treatment does not pose a problem with regard to the complexity of the pricing problem as their dual values simply affect the arc costs.

The subset row (SR) inequalities introduced by Jepsen et al. [15] are, however, a set of valid inequalities directly defined on the master problem variables, which cannot be expressed as linear combinations of the arc-flow variables. These inequalities form a subset of the Chvátal-Gomory (CG) rank 1 cuts for the set partitioning polytope. Given the CG multipliers $u_i^g \geq 0$ for $i \in C$, a CG rank 1 cut (indexed by g) for model (9)-(12) is expressed as:

$$\sum_{p \in P} \left\lfloor \sum_{i \in C} u_i^g \alpha_{ip} \right\rfloor \lambda_p \leq \left\lfloor \sum_{i \in C} u_i^g \right\rfloor . \tag{20}$$

The SR inequalities considered by Jepsen et al. [15] (and also by Desaulniers et al. [7]) are obtained using a CG multiplier equal to $\frac{1}{2}$ for exactly three constraints (10) and a zero multiplier for all the other constraints. As shown by their computational results, the use of these inequalities can significantly improve the lower bound computed at the root node. On the other hand, it increases the difficulty of the ESPPRC pricing problem as each cut requires one additional resource.

Petersen et al. [20] extended this work and showed how any CG rank 1 cut (20) can be applied when using a BCP method for solving model (9)-(12). In this case, the reduced cost of a variable λ_p also depends on the positive coefficients $\lfloor \sum_{i \in C} u_i^g \alpha_{ip} \rfloor$ of the CG cuts and their corresponding dual values. As discussed next, the computation of these coefficients in a labeling algorithm is not straightforward because of their stepwise nature. Let $\mathcal{CG}1$ be the set of CG rank 1 cuts added to the master problem and denote by β_g the non-positive dual variable associated with cut $g \in \mathcal{CG}1$ and by u^g the CG multiplier vector of size $|C|$ defining this cut. For each cut $g \in \mathcal{CG}1$, an additional resource is defined. This resource computes the current fractional part of $\sum_{i \in C} u_i^g \alpha_{ip}$, that is, for a label L representing a partial path that visits the customer node set $C(L)$, the value of this resource is given by $T^g(L) = \sum_{i \in C(L)} u_i^g - \lfloor \sum_{i \in C(L)} u_i^g \rfloor$. It is not restricted by resource windows. In the labeling algorithm, the forward extension of a label L along an arc $(i,j) \in A$ to create a new label L' proceeds as follows for the cost and cut components:

$$T^g(L') = (T^g(L) + u_i^g) \pmod 1 \qquad \forall g \in \mathcal{CG}1 \tag{21}$$

$$T_{cost}(L') = T_{cost}(L) + c_{ij} - \pi_j - \sum_{\substack{g \in \mathcal{CG}1 \\ T^g(L)+u_i^g \geq 1}} \beta_g. \tag{22}$$

The computation of the components $T_r(L')$, $r \in \mathcal{R}$, is performed using the previous extension functions (15)-(17).

The use of the CG rank 1 cuts also requires modifying the dominance criterion as follows.

Proposition 2 (Petersen et al. [20])**.** Let L and L' be two labels representing partial paths ending at the same node. Label L dominates label L' (which can be discarded) if

$$T_{cost}(L) - \sum_{\substack{g \in \mathcal{CG}1 \\ T^g(L) > T^g(L')}} \beta_g \leq T_{cost}(L') \tag{23}$$

$$T_r(L) \leq T_r(L') \qquad \forall r \in \mathcal{R}. \tag{24}$$

This dominance criterion is a generalization of the one proposed by Jepsen et al. [15], but it is equivalent when considering only the SR inequalities.

The separation of the CG rank 1 cuts is an $\mathcal{NP}$-hard problem (see Eisenbrand [10]). As proved by Jepsen et al. [15], the separation of the SR cuts is also $\mathcal{NP}$-hard. Consequently, Jepsen et al. [15] and Desaulniers et al. [7] considered only the SR inequalities involving exactly three constraints and used enumeration to find the violated ones.

4 Clique Inequalities

Clique inequalities are well-known valid inequalities for the set packing and set partitioning polytopes (see Padberg [19], Schrijver [23]). In particular, they are facet defining for the master problem polytope $conv\{\lambda \in \mathbb{R}^{|P|} : \sum_{p \in P} \alpha_{ip}\lambda_p = 1, \forall i \in C, \lambda_p \in \{0,1\}, \forall p \in P\}$. In this section, we discuss how these inequalities can be applied to strengthen the master problem formulation (9)-(11) when solved by column generation.

A clique is defined on a *conflict* graph $\mathcal{G} = (V, E)$ that is undirected. Its vertex set V contains one vertex for each route $p \in P$, that is, $V = P$. Its edge set E contains an edge between two vertices p and q of V if routes p and q have at least one customer in common (p and q are said to be *in conflict* or *conflicting*), that is,

$$E = \{(p,q) : p, q \in P, p \neq q, \exists\, i \in C \text{ such that } \alpha_{ip} = \alpha_{iq} = 1\}.$$

Thus, an edge (p,q) identifies a conflict between p and q for which the variables λ_p and λ_q cannot both take value 1 in a feasible integer solution. A clique W is a maximal subset of vertices of V such that $(p,q) \in E$ for all pairs $p, q \in W$, $p \neq q$.

Given a clique $W \subseteq P$ in $\mathcal{G}$, the corresponding clique inequality is expressed as:

$$\sum_{p \in W} \lambda_p \leq 1. \tag{25}$$

It simply says that at most one of the variables λ_p, $p \in W$, can be set at 1 in a feasible solution of the integer master problem (9)-(12).

Given a fractional-valued solution to the master problem, the separation of the most-violated clique inequality can be done on the conflict graph $\mathcal{G}$, where each vertex is given a weight equal to the value of the corresponding variable. A maximal weight clique in $\mathcal{G}$ provides a most-violated clique inequality. This separation problem is known as the maximum weighted clique problem and is strongly $\mathcal{NP}$-hard (see Schrijver [23]).

Given the complexity of solving this separation problem, we use a separation heuristic that is described in the following subsection. Afterwards, we propose a representation of the clique inequalities which facilitates their treatment in the labeling algorithm used for solving the pricing problem. Next, we introduce the additional resources required in the ESPPRC pricing problem for handling these inequalities and develop a new dominance criterion for the labeling algorithm. Finally, we discuss a special case of clique inequalities.

4.1 Separation Heuristic

As proposed by Hoffman and Padberg [12], a simple heuristic for identifying violated clique inequalities is to look for cliques that are each built from an initially chosen constraint (10). This row is used to determine an initial set of routes (vertices) to put in the clique. Figure 1 provides a pseudo-code of this heuristic procedure. Given a fractional-valued solution to the master problem (9)-(11), let P^F be the index set of the variables λ_p that take a fractional value in this solution. For each row $i \in C$, the heuristic starts by identifying the set $idIn \subseteq P^F$ of routes p with a non-zero coefficient α_{ip} in row i and the set $idOut \subseteq P^F$ of routes with a zero entry in this row (thus, $P^F = idIn \cup idOut$). The routes in $idIn$ will all be part of the clique under construction. Note that the sum of the values of the variables associated with these routes is equal to 1. Then, all routes in $idOut$ that are conflicting with all routes in $idIn$ are added to a candidate set $idCand$ of routes. If this set is empty, then no violated clique inequalities involving all the routes in $idIn$ exist and the procedure proceeds to the next row $i \in C$. Otherwise, there exists at least one such clique that defines a violated inequality. In this case, a greedy procedure, named GREEDY-ADD-ROUTES, is used to add routes of fractional-valued variables to the routes in $idIn$. These additional routes allow to compute by how much the inequality under construction is violated. If this violation exceeds a prespecified parameter value *min_violation*, then the same greedy procedure is called to complete the clique by also considering the routes in $P \setminus P^F$. The identified violated inequalities are stored in the set *idCliques*.

The GREEDY-ADD-ROUTES procedure, detailed in pseudo-code of Figure 2, takes as input two sets of routes: *idClique* and *idCand*. The *idClique* set contains two-by-two conflicting routes that will be part of the clique. The *idCand* set is composed of routes that are candidates to be added to the clique. The procedure starts by sorting the set *idCand* in decreasing order of the routes' density (i.e., the number of customers they contain). Then respecting this order,

```
FIND-CLIQUES
1  for each row $i \in C$
2        do $idCand \leftarrow \emptyset$
3           $idIn \leftarrow$ ROUTES-IN-ROW($i$) $\cap P^F$
4           $idOut \leftarrow$ ROUTE-NOT-IN-ROW($i$) $\cap P^F$
5           for each route $p$ in $idOut$
6                 do if $p$ is conflicting with all routes in $idIn$
7                       then add $p$ to $idCand$
8           if $idCand \neq \emptyset$
9              then $idClique \leftarrow$ GREEDY-ADD-ROUTES($idIn, Cand$)
10                  if COMPUTE-VIOLATION($idClique$) $\geq min\_violation$
11                     then add GREEDY-ADD-ROUTES($idClique, P \setminus P^F$) to $idCliques$
12 return $id$Cliques
```

Figure 1: Pseudo-code for finding clique inequalities.

```
GREEDY-ADD-ROUTES($idClique$, $idCand$)
1 sort $idCand$ in decreasing order of density
2 for each route $p$ in $idCand$
3       do if $p$ is conflicting with all routes in $idClique$
4             then add $p$ to $idClique$
5 return $idClique$
```

Figure 2: Pseudo-code of the greedy procedure for adding routes to the current *idClique* set.

the routes are added to the *idClique* set if they are conflicting with the current routes in this set. Note that, when this procedure is called in Step 9 of the FIND-CLIQUES procedure, there is no need to verify if the routes in *idCand* are conflicting with the columns in the initial *idClique* set as this was already verified in Step 6.

4.2 Clique Representation

When column generation is used for solving the master problem, only a subset of its columns are known at the end of the solution process, namely, those present in the restricted master problem. Clique inequalities are defined using the routes of these known columns. However, a route p generated after adding a cut can conflict with all routes in the corresponding clique and can thus be added to the clique to yield a larger clique and a stronger inequality. Consequently, the reduced cost of the corresponding variable λ_p should also depend on the dual value of the cut. Therefore, this value must be taken into account by the labeling algorithm when solving the pricing problem. This means that, when creating a new label at a node $j \in N$ in the labeling algorithm, one must check, for every clique cut, whether or not the addition of node j yields a partial path (route) that now conflicts with all routes in the clique. If so, one must subtract from the label's reduced cost the dual value of the corresponding cut. This check, called hereafter the CC check (for clique conflict check), can be very time consuming when cliques involve large numbers of routes. In this section, we propose an approximate representation of a clique that, in practice, reduces the time required to perform this CC check. As we will see in the next subsection, this representation also increases the number of dominated labels that can be discarded during the execution of the labeling algorithm. Nevertheless, such an approximate representation does not come for free. Some weaknesses regarding the strength of the computed clique inequalities will be pointed out during the description of this representation.

Let $\boldsymbol{\alpha}_q = \left(\alpha_{iq}\right)_{i \in C}$ for all $q \in P$. Consider a clique W and a newly generated route $p \in P \setminus W$. To verify if p is in conflict with all routes in W using a complete representation of W, one has to check if $\boldsymbol{\alpha}_p$ is orthogonal with $\boldsymbol{\alpha}_q$ for each route q in W. This CC check requires $O(|W||C|)$ operations. To reduce the (practical) complexity of this operation, we rather use an approximate representation that considers only a subset of the rows in C and a subset of the routes in W. To describe this representation, let us start with preliminary notation and definitions. Let $idRows(p) \subseteq C$ be the set of customers visited in route $p \in P$.

Definition 1. Let i be a customer in C, and p and q be two routes in P. A constraint in (10) indexed by i (more briefly, row i) is said to be a *conflicting row for* p *and* q if both routes visit customer i, that is, if $i \in \big(idRows(p) \cap idRows(q)\big)$.

Definition 2. Let W be a clique. The *set* $\chi(W) \subseteq C$ *of conflicting rows of* W is given by

$$\chi(W) = \bigcup_{\substack{p,q \in W \\ p \neq q}} \big(idRows(p) \cap idRows(q)\big).$$

In other words, row $i \in C$ belongs to $\chi(W)$ if i is conflicting for at least one pair of distinct routes p and q in W.

Let W be a clique and denote by $idConfPairs_i(W) \subseteq W \times W$ the set of pairs of routes p and q in W for which row $i \in C$ is conflicting.

Definition 3. Let W be a clique. A *minimal subset* $\chi_{min}(W)$ *of conflicting rows of* W is a subset of $\chi(W)$ such that

$$\bigcup_{i \in \chi_{min}(W)} idConfPairs_i(W) = \{(p,q) : p,q \in W, p \neq q\} \tag{26}$$

and

$$idConfPairs_i(W) \setminus \left(\bigcup_{j \in \chi_{min}(W) \setminus \{i\}} idConfPairs_j(W) \right) \neq \emptyset, \qquad \forall i \in \chi_{min}(W) \tag{27}$$

Condition (26) ensures that all pairs of routes in W is conflicting with respect to at least one row in subset $\chi_{min}(W)$ (that is, W would still be a clique if the set of constraints (10) was restricted to the subset of indices in $\chi_{min}(W)$), while condition (27) guarantees the minimality of this subset (that is, the first condition would not hold if any row is removed from the subset).

```
FIND-MINIMAL-SUBSET($idClique$)
 1  $idCurSet \leftarrow \emptyset$
 2  for each row $i \in C$
 3        do $idConfPairs(i) \leftarrow \emptyset$
 4           for each pair of routes $p$ and $q$ in $idClique$
 5                 do if $i$ is conflicting for $p$ and $q$
 6                       then $idCurSet \leftarrow idCurSet \cup \{i\}$
 7                            add $(p, q)$ to $idConfPairs(i)$
 8  Sort $idCurSet$ in increasing order of the cardinality of $idConfPairs(i)$, $i \in idCurSet$
 9  for each row $i \in idCurSet$
10        do $idRemove \leftarrow 1$
11           for each pair of routes $(p, q) \in idConfPairs(i)$
12                 do if $\nexists j \in idCurSet \setminus \{i\}$ such that $(p, q) \in idConfPairs(j)$
13                       then $idRemove \leftarrow 0$
14           if $idRemove = 1$
15              then remove $i$ from $idCurSet$
16  return $idCurSet$
```

Figure 3: Pseudo-code for finding a minimal subset of conflicting rows of a clique.

The minimal subset of conflicting rows of a given clique is not unique. Finding a minimal subset of minimal cardinality is $\mathcal{NP}$-hard: it corresponds to a set covering problem where each pair of routes in the clique defines a covering constraint, and each constraint in (10) requires a binary variable with a cost coefficient of 1. Given this complexity, for finding a minimal subset for a clique W, we rather resort to a heuristic procedure that starts from the set of conflicting rows $\chi(W)$ and sequentially removes rows from it until satisfying condition (27). The pseudo-code of this procedure, called FIND-MINIMAL-SUBSET, is given in Figure 3. In steps 1 to 7, the set $\chi(idClique)$ of conflicting rows of *idClique* and the set $idConfPairs(i)$ of conflicting pairs of routes for each row $i \in C$ are built. In Step 8, the rows i in the set $idCurSet = \chi(idClique)$ are sorted in increasing order of the cardinality of their set $idConfPairs(i)$. This sort favors finding a minimal subset of low cardinality. Then, in Steps 9 to 15, the rows in *idCurSet* are removed sequentially from this subset if they do not meet condition (27). The set *idCurSet* returned in Step 16 is a minimal set of conflicting rows of *idClique*.

The proposed approximate representation is based on the following observation.

Observation 1. Let W be a clique and $\chi_{min}(W)$ a minimal subset of its conflicting rows. Let p be a newly generated route. If p is in conflict with all routes in W when considering only the rows in $\chi_{min}(W)$, then p is also in conflict with all these routes when all rows $i \in C$ are considered.

Observation 1 provides a sufficient condition to identify a route that can be used to enlarge a clique. This sufficient condition is, however, not a necessary one as shown by the following example.

Example 1. Consider a set of rows (customers) $C = \{1, 2, 3\}$, a clique $W = \{1, 2\}$ and two newly generated routes a and b that are both conflicting with routes 1 and 2. The columns of all these four routes, restricted to the three rows in C, are shown below.

$p =$	1	2	a	b
	1			1
		1	1	1
	1	1	1	

In this case, there exists a single minimal subset of conflicting rows of W, namely, $\chi_{min}(W) = \{3\}$. Considering only this row, route a would be declared in conflict with routes 1 and 2, while route b would not be identified as such.

Given a clique W, the number of conflicting rows in a minimal subset $\chi_{min}(W)$ is at most $\sum_{i=1}^{|W|-1} i = \frac{|W|(|W|-1)}{2}$, i.e., the number of rows in this subset is $O(\min\{|C|, |W|^2\})$. Consequently, when $|C| > |W|^2$, considering only a minimal subset of conflicting rows when performing the CC check can clearly accelerate this operation. This approximate CC check might, however, result in a slightly weaker clique inequality because certain newly generated routes may not become part of the clique even though they are actually conflicting with all routes in this clique (when all rows are considered in the check). On the other hand, note that a stronger inequality defined on a superset of this clique and including these routes can be generated at a later stage. This enlarged clique would yield a different minimal subset of conflicting rows.

An approximate representation of a clique W based on a minimal subset $\chi_{min}(W)$ is now available. It is composed of the vectors $\boldsymbol{v}_p(\chi_{min}(W))$, $\forall p \in W$, where $\boldsymbol{v}_p(\chi_{min}(W))$ is the sub-vector of the column $\boldsymbol{\alpha}_p$ restricted to the rows in $\chi_{min}(W)$. This representation can be made more compact by noticing that a newly generated route is in conflict with a route $p \in W$ if it is also conflicting with a route $q \in W$, $q \neq p$, such that $\boldsymbol{v}_q(\chi_{min}(W)) \leq \boldsymbol{v}_p(\chi_{min}(W))$. It is easy to prove that there are no pairs of routes p and q in W such that $\boldsymbol{v}_q(\chi_{min}(W)) < \boldsymbol{v}_p(\chi_{min}(W))$ (otherwise, $\chi_{min}(W)$ would not be a minimal subset of conflicting rows of W). Consequently, one can eliminate from the representation all vectors $\boldsymbol{v}_p(\chi_{min}(W))$, except one, that are equal. For a clique W, we denote by $\Gamma(W)$ the resulting set of routes used in the representation. This representation, called a key set, is defined as follows.

Definition 4. Given a clique W and a minimal set $\chi_{min}(W)$ of its conflicting rows, the *key set* of W is composed of the sub-vectors $\boldsymbol{v}_p(\chi_{min}(W))$, $\forall p \in \Gamma(W)$.

To accelerate the CC check, one can therefore use the key set of a clique W (based on a minimal subset $\chi_{min}(W)$) instead of its complete representation. However, this approach poses a problem since the key set would grow for each route p added to the clique such that $\boldsymbol{v}_p(\chi_{min}(W)) \not\geq \boldsymbol{v}_q(\chi_{min}(W))$ for at least one route $q \in \Gamma(W)$. Also, two routes generated in the same column generation iteration could exclude one another from entering the clique if they were not conflicting with each other. This is another weakening of the clique representation that can lead to the separation of another almost identical clique with a different key set. To avoid these drawbacks, we propose to add in a clique W only the routes p for which $\boldsymbol{v}_p(\chi_{min}(W)) \geq \boldsymbol{v}_q(\chi_{min}(W))$ for at least one route $q \in \Gamma(W)$. This restriction, called the *clique admissibility rule*, guarantees that all added routes conflict with each other (since they all contain at least one of the key set sub-vectors which all conflict), and allows to always keep the same key set as new routes are added to the clique. A route p satisfying the admissibility rule of a clique W is said to be *admissible to enlarge clique W*.

The usage of this approximate clique representation and the clique admissibility rule is illustrated in the following example.

Example 2. Consider five routes $p = 1, \dots, 5$ that define the key set of a clique W with respect to a minimal subset $\chi_{min}(W)$ and three newly generated routes $p = a, b, c$. Their respective sub-vectors $\boldsymbol{v}_p(\chi_{min}(W))$ are given below.

$p =$	1	2	3	4	5	a	b	c
	1	1				1		
	1		1			1		1
	1			1	1	1	1	1
		1	1	1			1	
		1			1			1
			1		1		1	

With the proposed representation, routes a and b are admissible to enlarge W because $\boldsymbol{v}_a(\chi_{min}(W)) \geq \boldsymbol{v}_1(\chi_{min}(W))$ and $\boldsymbol{v}_b(\chi_{min}(W)) \geq \boldsymbol{v}_4(\chi_{min}(W))$. However, even though route c conflicts with all routes in the key set, it is inadmissible to enlarge W because $\boldsymbol{v}_c(\chi_{min}(W)) \not\geq \boldsymbol{v}_p(\chi_{min}(W))$ for $p = 1, \dots, 5$.

4.3 Modified Pricing Problem and Labeling Algorithm

Let Ω be the set of cliques W defining the inequalities (25) considered in the master problem at a given moment during the execution of the BCP algorithm and let ζ_W be the non-positive dual variable associated with the clique inequality (25) for $W \in \Omega$. The treatment of these inequalities has an impact on the definition of the pricing problem. Indeed, the reduced cost of a variable λ_p now depends on the dual values of these inequalities, that is,

$$\bar{c}_p = \sum_{(i,j) \in p} (c_{ij} - \pi_j) - \sum_{W \in \Omega : p \in W} \zeta_W. \tag{28}$$

Therefore, the labeling algorithm used to solve the pricing problem must be modified to account for these modified reduced costs. With the key set representation of a clique and the clique admissibility rule, it is easy to determine if a route p is admissible to enlarge a clique W when p is completely known. However, this is not an easy task when p is under construction in the labeling algorithm. In this section, we discuss how this can be done.

The first required modification to the labeling algorithm consists of adding new components to the labels that enable to determine if a path is admissible to enlarge a clique. For each clique $W \in \Omega$, $|\chi_{min}(W)| + 1$ binary resources are defined and the corresponding components are added to each label (these components are stored as bitvectors to save memory space). For a label representing a partial path p, the first $|\chi_{min}(W)|$ of these resource values indicate, until it is proven that p can enlarge W, whether or not each customer of the minimal subset $\chi_{min}(W)$ has been visited along p. Thus, they represent the sub-vector $\boldsymbol{v}_p(\chi_{min}(W))$ and are compared in the CC check to the key set vectors of W. When the CC check is positive (that is, p is admissible to enlarge W), these resource values are set to 0 and never changed in the subsequent label extensions. Furthermore, the dual value ζ_W is subtracted from the label (reduced) cost component. For a label L, these resource components are denoted $T^W_{cust_\ell}(L)$, $\ell \in \chi_{min}(W)$. The other additional resource component, denoted $T^W_{inadm}(L)$, simply indicates whether or not p is inadmissible to enlarge the clique: it is equal to 1 if p is inadmissible, and 0 otherwise. Consequently, if $T^W_{inadm}(L) = 0$, then $T^W_{cust_\ell}(L) = 0$, $\forall \ell \in \chi_{min}(W)$, and any complete path obtained by extending L until reaching the sink node provides a route that can be added to the clique W. The set of all these new resources for all cliques in Ω is denoted $\mathcal{Q}$.

With these additional resources, the labeling algorithm can be applied for solving exactly the modified pricing problem if the dominance criterion of Proposition 1 considers the set $\mathcal{R} \cup \mathcal{Q}$ instead of only $\mathcal{R}$. However, given the large number of resources required by each clique inequality, the dominance criterion becomes weaker as clique inequalities are added and the difficulty of solving the pricing problem drastically increases. This was mentioned earlier by Jepsen et al. [15] and Petersen et al. [20] when adding CG cuts in a column generation context. To improve the performance of their labeling algorithms, these authors have exploited the facts that only minimal cost paths are sought and that the CG cuts all have non-positive dual values, i.e., they can be considered as penalties in the pricing problem since they are subtracted. Indeed, a label L can dominate another label L' if the difference between the cost of L' and that of L is at least equal to the maximal sum of the penalties that L can pay when further extended toward the sink node, while L' would not pay them. This observation allows to neglect the resources defined for the inequalities in the dominance criterion (see Proposition 2). A similar trick was used by Chabrier [4] to increase dominance when the set of customers visited by a partial path includes at most two customers that do not belong to the set of customers visited by another (possibly dominating) partial path.

The dominance criterion we propose is very similar to the one developed for the CG inequalities, although some adjustments must be done because clique inequalities do not have as nice properties as CG inequalities. This criterion is stated in the following proposition.

Proposition 3. Let L and L' be two labels representing partial paths ending at the same node. Label L dominates label L' (which can be discarded) if

$$T_{cost}(L) - \sum_{W \in \Omega_{LL'}} \zeta_W \leq T_{cost}(L') \tag{29}$$

$$T_r(L) \leq T_r(L') \qquad \forall r \in \mathcal{R}, \tag{30}$$

where $\Omega_{LL'} = \{W \in \Omega : T^W_{inadm}(L) = 1 \text{and} \big(T^W_{inadm}(L') = 0 \text{or} \exists\, \ell \in \chi_{min}(W) \text{such that} T^W_{cust_\ell}(L) > T^W_{cust_\ell}(L')\big)\}$ is the set of cliques W for which the penalty ζ_W could be paid in a feasible extension of L along an arc sequence, while it would not be paid when extending similarly L'.

Proof. The proof is done for a single clique $W \in \Omega$ and can be trivially extended for the whole clique set Ω. To prove that L dominates L' when (29)–(30) hold, we show that i)

$F(L') \subseteq F(L)$, and ii) $T_{cost}(L) + f(L,\epsilon) \le T_{cost}(L') + f(L',\epsilon)$ for all $\epsilon \in F(L')$, where $F(J)$ is the set of feasible extensions (along arc sequences) of a label J and $f(J,\epsilon)$ is the difference between the cost of label J and that of the path obtained by extending label J using extension ϵ. Observe that, for any label J,

$$f(J,\epsilon) = \begin{cases} f_{arc}(\epsilon) & \text{if } T^W_{inadm}(J) = 0 \text{ or } T^W_{inadm}(J \oplus \epsilon) = 1 \\ f_{arc}(\epsilon) - \zeta_W & \text{otherwise,} \end{cases} \tag{31}$$

where $f_{arc}(\epsilon)$ is the sum of the reduced costs $(c_{ij} - \pi_j)$ of the arcs (i,j) traversed in extension ϵ, and $J \oplus \epsilon$ denotes the label created by extending J using extension ϵ. Note also that $f_{arc}(\epsilon) \le f(J,\epsilon)$ because $\zeta_w \le 0$.

Assume that relations (29)–(30) are satisfied. Given that all extension functions of the resources $r \in \mathcal{R}$ are non-decreasing, relations (30) imply that $F(L') \subseteq F(L)$. Now, to show that $T_{cost}(L) + f(L,\epsilon) \le T_{cost}(L') + f(L',\epsilon)$ for all $\epsilon \in F(L')$, we consider the following four cases.

Case 1: $T^W_{inadm}(L) = 0$.

In this case, $W \notin \Omega_{LL'}$ and $f(L,\epsilon) = f_{arc}(\epsilon)$. Consequently, relation (29) writes as

$$\begin{aligned} & T_{cost}(L) \le T_{cost}(L') \\ \Rightarrow \quad & T_{cost}(L) + f_{arc}(\epsilon) \le T_{cost}(L') + f_{arc}(\epsilon) \\ \Rightarrow \quad & T_{cost}(L) + f(L,\epsilon) \le T_{cost}(L') + f_{arc}(\epsilon) \le T_{cost}(L') + f(L',\epsilon). \end{aligned}$$

Case 2: $T^W_{inadm}(L) = 1$, $T^W_{inadm}(L') = 1$, and $T^W_{cust_\ell}(L) \le T^W_{cust_\ell}(L'), \forall \ell \in \chi_{min}(W)$.

In this case, $W \notin \Omega_{LL'}$. Furthermore, because $T^W_{cust_\ell}(L) \le T^W_{cust_\ell}(L'), \forall \ell \in \chi_{min}(W)$, one can deduce that $T^W_{inadm}(L \oplus \epsilon) \ge T^W_{inadm}(L' \oplus \epsilon)$ and $f(L,\epsilon) \le f(L',\epsilon)$. Consequently, from relation (29), we find that

$$\begin{aligned} & T_{cost}(L) \le T_{cost}(L') \\ \Rightarrow \quad & T_{cost}(L) + f(L,\epsilon) \le T_{cost}(L') + f(L,\epsilon) \le T_{cost}(L') + f(L',\epsilon). \end{aligned}$$

Case 3: $T^W_{inadm}(L) = 1$, $T^W_{inadm}(L') = 1$, and $\exists\, \ell \in \chi_{min}(W)$ such that $T^W_{cust_\ell}(L) > T^W_{cust_\ell}(L')$.

In this case, $W \in \Omega_{LL'}$. Consequently, relation (29) writes as

$$\begin{aligned} & T_{cost}(L) - \zeta_W \le T_{cost}(L') \\ \Rightarrow \quad & T_{cost}(L) + f_{arc}(\epsilon) - \zeta_W \le T_{cost}(L') + f_{arc}(\epsilon) \\ \Rightarrow \quad & T_{cost}(L) + f(L,\epsilon) \le T_{cost}(L') + f_{arc}(\epsilon) \le T_{cost}(L') + f(L',\epsilon). \end{aligned}$$

Case 4: $T^W_{inadm}(L) = 1$ and $T^W_{inadm}(L') = 0$.

In this case, $W \in \Omega_{LL'}$ and the proof is identical to that of the case 3.

□

The dominance criterion of Proposition 3 gives the possibility to dominate more labels than when dominance is done using the criterion of Proposition 1 with the set $\mathcal{R} \cup \mathcal{Q}$. Indeed, the latter criterion allows no dominance in the above cases 3 and 4, whereas the former criterion can identify dominated labels in these cases.

To further increase the number of dominated labels, condition (29) can be relaxed by considering the notion of unreachable customer as defined in Section 3. Indeed, if, for a clique $W \in \Omega$, $T^W_{inadm}(L) = 1$ and $U_\ell(L) = 1$ for all $\ell \in \chi_{min}(W)$ such that $T^W_{cust_\ell}(L) = 0$, then W can be excluded from the set $\Omega_{LL'}$, that is, the penalty ζ_W can be disregarded in (29).

Finally, remark that the use of minimal subsets of conflicting rows (instead of the whole sets of conflicting rows) in the proposed clique representation increases the possibility of dominating labels simply because less clique resources are needed.

4.4 A Special Case of Clique Inequalities

As mentioned in Section 3, the SR inequalities (20) used by Jepsen et al. [15] and Desaulniers et al. [7] for the VRPTW are obtained using a CG multiplier equal to $\frac{1}{2}$ for exactly three constraints in (10) and a zero multiplier for all the other constraints. For a given subset B of three customers in C, such an inequality can be rewritten as:

$$\sum_{p \in P_B} \lambda_p \leq 1, \tag{32}$$

where $P_B \subseteq P$ is the subset of routes visiting at least two customers in B. It is easy to see that this SR inequality is in fact a clique inequality (25) that can be represented with a minimal subset of conflicting rows containing the three rows (10) associated with the customers in B and the key set composed of the three vectors $(1, 1, 0)^\top$, $(1, 0, 1)^\top$, and $(0, 1, 1)^\top$. Hence, the SR inequalities used by Jepsen et al. [15] and Desaulniers et al. [7] are a special case of the clique inequalities. Nevertheless, because these SR inequalities only require a single additional resource each to be treated, it is more advantageous to treat them as SR inequalities than as clique inequalities that would necessitate four extra resources each.

5 Computational Experiments

We conducted computational experiments to assess the use of the clique inequalities (25) when the VRPTW is solved by the BCP method described in Section 3. To do so, we compare the results of two different solution methods. The first method, denoted SR, corresponds to a BCP method in which only subset row cuts are added as cutting planes. For this method, we used the code, the heuristic strategies, and the parameter setting that were used by Desaulniers et al. [7]. In particular, we applied the same heuristics for generating rapidly negative reduced cost columns, avoiding in most column generation iterations the use of an exact (and often highly time-consuming) dynamic programming algorithm for solving the ESPPRC pricing problem. Also, we considered only SR cuts defined with exactly three constraints and did not reoptimize the modified master problem to optimality after adding cuts in a branch-and-bound node (in this case, the lower bound of the node corresponds to the bound computed before adding cuts). The second method, denoted CLIQUE, is the same as the first one except that violated clique inequalities can be added when no violated SR inequalities are found. Because the SR cuts defined with three constraints are also clique cuts as discussed in Section 4.4, this method can be seen as a method applying only clique cuts in which a special case of the clique cuts (the SR cuts) are favored and treated more efficiently. The generation of clique cuts are subject to the following rules which are similar to those applied for the SR cuts. The most violated clique cut must be violated by at least 0.1 before adding any cuts and cuts violated by less than 0.01 are not added. The maximal number of cuts added per iteration is 20. Furthermore, a clique cut can only be added if the minimal subset of conflicting rows of its clique differ by at least 20 % with the minimal subset of conflicting rows of all already generated clique cuts. Finally, we do not consider clique cuts defined for cliques represented by a minimal subset containing more than 31 conflicting rows (always allowing the use of efficient bitwise operations when using a 32-bit processor).

The experiments were conducted on a subset of the well-known VRPTW benchmark instances of Solomon [24]. Solomon created 56 instances, each involving 100 customers. They can be divided into 3 classes according to the spatial distribution of the customers: the customer locations are clustered in the C class, randomly distributed in the R class, and partly clustered and partly randomly distributed in the RC class. Furthermore, each class contains two series of instances. In the 1-series instances (C1, R1, and RC1), the time windows are relatively narrow, while they are much wider in the 2-series instances (C2, R2, and RC2). The 1-series instances are, in general, easier to solve than the 2-series instances. To the best of our knowledge, 5 of these 56 instances (all R2 and RC2 instances) have not yet been solved to optimality (see Desaulniers et al. [7]). For the unsolved instances, even their initial master problems (that is, without any cuts) have not yet been solved when using column generation with the ESPPRC as a pricing problem. Consequently, the clique inequalities presented in this paper cannot be useful for the moment to help solving these instances.

Our tests focused only on the R1 and RC1 instances. The C1 and C2 instances were not considered because they can all be solved in the root node of the search tree without adding any cuts. The 19 R2 and RC2 were not considered for three reasons: i) 5 of them cannot be solved with the proposed methodology; ii) 8 of the 14 others can be solved in the root node by considering only the SR inequalities; and iii) for most of the remaining ones, the computational times are highly dependent on the number of times the ESPPRC pricing problem is solved with the exact dynamic programming algorithm. The instances are identified with the usual notation, that is, for example, RC105 indicates the fifth instance of the 1-series in the RC class.

All computational experiments were performed on a machine with 12 GB of memory equipped with a 64-bit Dual Core AMD Opteron processor 275 clocked at 2.2 GHz and a Linux operating system. The BCP method was implemented using the Gencol library (version 4.5) that is commercialized by Kronos Inc and represents the resource values as 32-bit integers. All restricted master problems were solved using the CPLEX solver, version 10.1.1.

The computational results reported in this section are divided into four parts. First, we investigate the use of the key set representation of a clique compared to its full representation. Second, we compare the root node lower bounds obtained by the SR and CLIQUE methods. Third, we compare the total running times for solving the VRPTW instances with both

	Complete					Key set				
Instance	CPU	LB	cuts	Avg. rows in χ	Avg. cols in clique	CPU	LB	cuts	Avg. rows in χ_{min}	Avg. cols in Γ
R104	4542	971.0	21	29.7	27.3	3163	971.0	20	10.8	14.0
R106	163	1234.4	6	30.0	27.0	122	1234.4	6	11.5	14.3
R108	3710	932.1	4	29.8	24.8	3964	932.1	8	10.9	15.1
R109	211	1144.0	5	28.4	23.8	255	1144.2	9	9.2	12.6
R110	1171	1067.5	45	27.5	23.7	711	1067.7	36	9.4	12.9
R111	544	1045.5	13	25.5	19.8	406	1045.4	6	8.2	9.0
R112	17118	946.0	20	29.3	27.3	16126	946.4	37	13.6	21.9
RC103	1182	1257.7	11	25.3	25.6	751	1257.7	2	5.0	6.0
RC104	9698	1129.3	5	29.6	30.2	8353	1129.3	8	10.5	16.3

Table 1: Complete representation and key set representation results.

methods. Finally, we provide average separation times for the SR and the clique cuts.

5.1 Clique Representation Results

With the first experiments, we want to assess the effectiveness of using the key set clique representation (as proposed in Section 4.2) instead of the complete representation. In these tests, we used, with both representations, the CLIQUE method for solving the master problem augmented with SR and clique cuts at the root node of the search tree. In Table 1, we report the results obtained for the R1 and RC1 instances for which clique inequalities were generated with a set of conflicting rows involving less than 32 rows (that is, $\chi(W) \leq 31$ for a clique W). This condition ensures that the generated clique inequalities are admissible for both representations, allowing a fair comparison.

For each instance and for both representations, this table indicates the total computational time in seconds for solving the master problem at the root node (CPU), the lower bound reached (LB), the total number of clique cuts generated (cuts), the average number of rows in $\chi(W)$ or $\chi_{min}(W)$ for representing a clique W, and the average number of columns in a clique W or in $\Gamma(W)$. From these results, we make the following observations. Firstly, for 3 of the 9 instances, the lower bound is stronger with the key set representation, while it is weaker for 1 instance. This might be surprising because the clique cuts are stronger when the cliques are completely represented, but as we are using a heuristic separation procedure, this is plausible. Secondly, the key set representation yields faster computational times for 7 of the 9 instances. For the other two instances, these times are slightly higher because more cuts were generated, yielding additional master problem reoptimizations. Overall, the average computational time is reduced by 12% when using the key set representation. Finally, we can attribute this gain in computational time to the fact that the key set representation is much more compact than the complete representation. Indeed, the number of rows and columns considered are reduced by averages of 65% and 47%, respectively, when using the key set representation.

In summary, for the tested instances, the key set representation yields an average reduction of 12% of the computational time. It also reduces memory consumption. Most importantly, it allows to generate and handle efficiently clique inequalities for cliques yielding more than 31 conflicting rows but less than 31 rows in a minimal subset of conflicting rows.

5.2 Lower Bound Results

In this section, we provide the lower bounds achieved by three methods (no cuts, SR, and CLIQUE) at the root node of the search tree. As opposed to what was used to obtain the integrality results presented in the next subsection (see the beginning of Section 5), the modified master problem was reoptimized to optimality after adding cuts for computing these lower bounds. Given that, at the root node, the CLIQUE method applies first all SR inequalities used by the SR method, the lower bounds produced by these three methods are always non-decreasing (in the order no cuts, SR, CLIQUE) for each instance. These lower bounds are reported in Table 2. In this table, the UB column gives the optimal value. The LB columns provide the lower bounds for each method, where a value in bold face indicates that the lower bound is equal to the upper bound. In these columns for the SR and CLIQUE methods, "*id*" (for identical) means that it was impossible for the corresponding method to improve the previous lower bound on the same line because it was equal to the upper bound. The "gap cl." columns for the SR and CLIQUE methods gives the percentage of the integrality gap of the no cuts method closed by the cuts ($= 100 \left(\frac{UB-LB_i}{UB-LB_{nocuts}} \right)$, $i = SR, CLIQUE$). Finally, for the CLIQUE method, the "gap cl. SR" column specifies the percentage of the integrality gap of the SR method closed by the clique cuts ($= 100 \left(\frac{UB-LB_{CLIQUE}}{UB-LB_{SR}} \right)$. In these columns, a "-" appears when the corresponding LB column contains "id".

These results indicate that the clique cuts can be useful to improve the lower bound quality at the root node of the search tree for 13 of the 20 instances tested. For the other 7 instances, the lower bounds reached by the SR method were already equal to their corresponding upper bounds. Out of these 13 instances, the clique cuts succeeded to improve the lower bound in 10 cases, totally closing the integrality gap for 2 instances (R105 and R108). In fact, the average integrality gap of the SR method closed by the clique cuts is 38%. This average is significant.

The lower bounds using clique inequalities are not as impressive as those obtained by the CG rank 1 cuts used in Petersen et al. [20]. This can be partially explained by the cut adding rules used in our algorithm (see the beginning of Section 5). Indeed, we do not add any violated inequalities if none of them is violated by at least 0.1, we use a heuristic separation routine for finding violated cliques, and we limit the size of the cliques (number of rows in the minimal subset) defining the cuts. This is in contrast with the work of Petersen et al. [20] where they added all inequalities violated by at most 0.0001 and used an exact separation procedure that was restricted to one hour of computational time at each call. Their goal was to obtain the best possible lower bounds and not necessarily to reduce the overall computational times.

		no cuts	SR		CLIQUE		
Instance	UB	LB	LB	gap cl.	LB	gap cl.	gap cl. SR
R101	1637.7	1631.2	1634.0	43.1 %	1634.0	43.1 %	0 %
R102	1466.6	**1466.6**	***id***	-	***id***	-	-
R103	1208.7	1206.8	**1208.7**	100.0 %	***id***	*id*	-
R104	971.5	956.9	970.5	93.1 %	971.3	98.6 %	80.0 %
R105	1355.3	1346.2	1355.2	98.9 %	**1355.3**	100.0 %	100.0 %
R106	1234.6	1227.0	1234.4	97.4 %	1234.4	97.4 %	0 %
R107	1064.6	1053.3	1063.3	88.5 %	1063.9	93.8 %	46.2 %
R108	932.1	913.6	932.0	99.5 %	**932.1**	100.0 %	100.0 %
R109	1146.9	1134.3	1144.0	77.0 %	1144.2	78.6 %	6.9 %
R110	1068.0	1055.6	1067.0	91.9 %	1067.6	96.8 %	60.0 %
R111	1048.7	1034.8	1045.3	75.5 %	1045.9	80.0 %	17.6 %
R112	948.6	926.8	945.8	87.2 %	946.9	92.2 %	39.3 %
RC101	1619.8	1584.1	**1619.8**	100.0 %	***id***	*id*	-
RC102	1457.4	1406.3	**1457.4**	100.0 %	***id***	*id*	-
RC103	1258.0	1225.6	1257.5	98.5 %	1257.7	99.1 %	40.0 %
RC104	1132.3	1101.9	1129.7	91.4 %	1129.9	92.1 %	7.7 %
RC105	1513.7	1472.0	**1513.7**	100.0 %	***id***	*id*	-
RC106	1372.7	1318.8	1367.5	90.4 %	1367.5	90.4 %	0 %
RC107	1207.8	1183.4	**1207.8**	100.0 %	***id***	*id*	-
RC108	1114.2	1073.5	**1114.2**	100.0 %	***id***	*id*	-

Table 2: Lower bounds achieved by the no cuts, SR and CLIQUE methods.

5.3 Integer Solution Results

Using the SR and CLIQUE methods, we solved to optimality the 20 R1 and RC1 instances. Table 3 presents the statistics of these experiments. For each instance and each method, it provides the total computational time in seconds (CPU), the total number of cuts generated (cuts), and the total number of nodes in the branch-and-bound search tree (BB). For the CLIQUE method, the numbers of SR and clique inequalities are given in parenthesis in the "cuts" column. Again, in the columns for the CLIQUE method, "*id*" means that, for the instance in question, the CLIQUE method coincides with the SR method because the instance could be solved in the root node without generating any clique inequalities.

From these results, we observe that clique inequalities were generated for 12 of the 13 instances which could not be solved in the root node with the SR method. For 8 of these 13 instances, the addition of clique inequalities reduced the number of branch-and-bound nodes as expected with stronger lower bounds, while this number increased for a single instance (R109). However, the computational times are, in general, longer with the CLIQUE method. In fact, they slightly decreased for only 3 instances, while they considerably increased for the instances R109, R111, and R112. For these instances, much more cuts were generated by the CLIQUE method and yielded a large number of additional reoptimizations that were not compensated by the decrease in the number of nodes for the R111 and R112 instances. Nevertheless, we can expect that using the clique inequalities may be competitive when the search trees are large, thereby trading the solution of many pricing problems for fewer but harder ones.

	SR			CLIQUE		
Instance	CPU	cuts	BB	CPU	cuts (SR/clique)	BB
R101	9	16	15	9	16 (16/0)	15
R102	3	0	1	*id*	*id*	*id*
R103	20	82	1	*id*	*id*	*id*
R104	2332	351	11	3990	347 (271/76)	5
R105	28	154	3	35	155 (154/1)	3
R106	88	139	3	104	152 (139/13)	3
R107	347	216	5	479	198 (171/27)	3
R108	1527	265	3	1249	285 (265/20)	1
R109	942	466	47	2373	865 (676/191)	91
R110	420	227	5	533	241 (208/33)	5
R111	5246	972	95	13667	1299 (739/560)	91
R112	17484	576	21	55226	666 (414/252)	11
RC101	28	107	1	*id*	*id*	*id*
RC102	194	194	1	*id*	*id*	*id*
RC103	1055	277	5	961	279 (277/2)	3
RC104	14294	372	25	14343	382 (346/37)	15
RC105	26	59	1	*id*	*id*	*id*
RC106	5240	943	87	4279	668 (616/52)	47
RC107	229	159	1	*id*	*id*	*id*
RC108	2203	192	1	*id*	*id*	*id*

Table 3: Integer solution results for the SR and CLIQUE methods.

5.4 Separation Results

In Table 4, we present results on the procedures used to separate the SR and clique inequalities in the CLIQUE method. For each instance and for each procedure, this table specifies the total number of times that the procedure was called, the total number of cuts generated (cuts), the total time in seconds spent in the procedure, and the average time per call.

Comparing the overall time spent in both separation procedures with the total computational time reported in Table 3 for the CLIQUE method, we observe that the overall separation time is, in general, relatively small, but it can sometimes be quite significant (up to 25% for instance R109). Because the SR separation routine is always invoked first and the clique separation routine is not invoked when violated SR inequalities are found, the number of calls to the first routine is always greater than that of the second one. For certain instances (in particular, R111 and R112), the clique routine is however called several times per branch-and-bound node to generate a large number of clique cuts. Finally, we remark that the average time per call required by the clique routine is, in general, much smaller than that of the SR routine. Thus, this gives room for a more extensive separation procedure for the clique inequalities that could further help improving the quality of the lower bounds.

	SR separation				clique separation			
Instance	called	cuts	total time	avg. time	called	cuts	total time	avg. time
R101	10	16	< 1	< 0.1	7	0	< 1	< 0.1
R102	0	0	0	0.0	0	0	0	0.0
R103	6	82	1	0.2	0	0	0	0.0
R104	33	271	176	5.3	15	76	19	1.3
R105	13	154	5	0.4	2	1	< 1	0.2
R106	14	139	21	1.5	5	13	1	0.2
R107	18	171	75	4.2	8	27	3	0.4
R108	19	265	86	4.5	1	20	4	4.0
R109	189	676	521	2.8	142	191	89	0.6
R110	20	208	88	4.4	7	33	3	0.4
R111	303	739	986	3.3	251	560	1120	4.5
R112	76	414	751	9.9	51	252	310	6.1
RC101	10	107	2	0.2	0	0	0	0.0
RC102	19	194	29	1.5	0	0	0	0.0
RC103	23	277	78	3.4	3	2	< 1	0.1
RC104	47	346	228	4.9	21	37	11	0.5
RC105	6	59	2	0.3	0	0	0	0.0
RC106	107	616	352	3.3	60	52	35	0.6
RC107	13	159	35	2.7	0	0	0	0.0
RC108	17	192	48	2.8	0	0	0	0.0

Table 4: Separation results for the SR and clique inequalities.

6 Conclusion

This paper is a first step towards adding special purpose cuts for the master problem in a BCP algorithm for the VRPTW. It considers clique inequalities defined directly on the master problem variables and shows how to deal with their dual values directly in the labeling algorithm used for solving the ESPPRC pricing problem. In particular, it proposes an approximate representation of a clique and a modified dominance criterion that allows for an efficient treatment of the clique inequalities. The computational results shows that these inequalities are helpful, in general, to reduce the number of branch-and-bound nodes to explore. However, the overall computational times are, in general, slower when applying them due to the number of additional master problem reoptimizations to perform.

We think that the proposed methodology can be improved to yield much better results. For example, a better separation heuristic could be used and the strategy always favoring the generation of SR cuts before that of clique cuts could be revised. Such developments will be the subject of future research.

References

[1] R. Baldacci, N. Christofides, and A. Mingozzi. An exact algorithm for the vehicle routing problem based on the set partitioning formulation with additional cuts. *Mathematical Programming*, 115(2):351–385, 2008. doi: 10.1007/s10107-007-0178-5.

[2] 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(3):316–329, 1998. doi: 10.1287/opre.46.3.316.

[3] N. Boland, J. Dethridge, and I. Dumitrescu. Accelerated label setting algorithms for the elementary resource constrained shortest path problem. *Operation Research Letters*, 34 (1):58–68, 2006. doi: 10.1016/j.orl.2004.11.011.

[4] A. Chabrier. Vehicle routing problem with elementary shortest path based column generation. *Computers & Operations Research*, 33(10):2972–2990, 2006. doi: 10.1016/j.cor.2005.02.029.

[5] G. B. Dantzig and P. Wolfe. Decomposition principle for linear programs. *Operations Research*, 8(1):101–111, 1960. doi: 10.1287/opre.8.1.101.

[6] G. Desaulniers, J. Desrosiers, J. Ioachim, I. M. Solomon, F. Soumis, and D. Villeneuve. A unified framework for deterministic time constrained vehicle routing and crew scheduling problems. In T. G. Crainic and G. Laporte, editors, *Fleet Management and Logistics*, pages 57–93. Kluwer, 1998.

[7] G. Desaulniers, F. Lessard, and A. Hadjar. Tabu search, partial elementarity, and generalized k-path inequalities for the vehicle routing problem with time windows. *Transportation Science*, 42(3):387–404, 2008. doi: 10.1287/trsc.1070.0223.

[8] M. Desrochers, J. Desrosiers, and M. Solomon. A new optimization algorithm for the vehicle routing problem with time windows. *Operations Research*, 40(2):342–354, 1992. doi: 10.1287/opre.40.2.342.

[9] M. Dror. Note on the complexity of the shortest path models for column generation in VRPTW. *Operations Research*, 42(5):977–979, 1994. doi: 10.1287/opre.42.5.977.

[10] F. Eisenbrand. Note - on the membership problem for the elementary closure of a polyhedron. *Combinatorica*, 19(2):297–300, 1999. doi: 10.1007/s004930050057.

[11] D. Feillet, P. Dejax, M. Gendreau, and C. Gueguen. An exact algorithm for the elementary shortest path problem with resource constraints: Application to some vehicle routing problems. *Networks*, 44(3):216–229, 2004. doi: 10.1002/net.v44:3.

[12] K. L. Hoffman and M. Padberg. Solving airline crew scheduling problems by branch-and-cut. *Management Science*, 39(6):657–682, 1993. doi: 10.1287/mnsc.39.6.657.

[13] S. Irnich and G. Desaulniers. Shortest path problems with resource constraints. In G. Desaulniers, Jacques Desrosiers, and M.M. Solomon, editors, *Column Generation*, chapter 2, pages 33–65. Springer, 2005. doi: 10.1007/0-387-25486-2_2.

[14] S. Irnich and D. Villeneuve. The shortest path problem with resource constraints and k-cycle elimination for $k \geq 3$. *INFORMS Journal on Computing*, 18:391–406, 2006. doi: 10.1287/ijoc.1040.0117.

[15] M. 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. doi: 10.1287/opre.1070.0449.

[16] B. Kallehauge, J. Larsen, O.B.G. Madsen, and M.M. Solomon. Vehicle routing problem with time windows. In G. Desaulniers and J. Desrosiers amd M.M. Solomon, editors, *Column Generation*, chapter 3, pages 67–98. Springer, 2005. doi: 10.1007/0-387-25486-2_3.

[17] N. Kohl, J. Desrosiers, O. B. G. Madsen, M. M. Solomon, and F. Soumis. 2-path cuts for the vehicle routing problem with time windows. *Transportation Science*, 33(1):101–116, 1999. doi: 10.1287/trsc.33.1.101.

[18] M. E. Lübbecke and J. Desrosiers. Selected topics in column generation. *Operations Research*, 53(6):1007–1023, 2005. doi: 10.1287/opre.1050.0234.

[19] M. Padberg. On the facial structure of set packing polyhedra. *Mathematical Programming*, 5:199–215, 1973. doi: 10.1007/BF01580121.

[20] B. Petersen, D. Pisinger, and S. Spoorendonk. Chvátal-Gomory rank-1 cuts used in a Dantzig-Wolfe decomposition of the vehicle routing problem with time windows. In B. Golden, R. Raghavan, and E. Wasil, editors, *The Vehicle Routing Problem: Latest Advances and New Challenges*, pages 397–420. Springer, 2008. doi: 10.1007/978-0-387-77778-8_18.

[21] G. Righini and M. Salani. Symmetry helps: bounded bi-directional dynamic programming for the elementary shortest path problem with resource constraints. *Discrete Optimization*, 3(3):255–273, 2006. doi: 10.1016/j.disopt.2006.05.007.

[22] G. Righini and M. Salani. New dynamic programming algorithms for the resource constrained shortest path problem. *Networks*, 51(3):155–170., 2008. doi: 10.1002/net.20212.

[23] A. Schrijver. *Combinatorial Optimization: Polyhedra and Efficiency*. Springer, 2003.

[24] M. M. Solomon. Algorithms for the vehicle routing and scheduling problem with time window constraints. *Operations Research*, 35(2):234–265, 1987. doi: 10.1287/opre.35.2.254.

Chapter 6

A Note on Valid Inequalities for a Dantzig-Wolfe Decomposition of an Integer Program

Simon Spoorendonk
DIKU Department of Computer Science, University of Copenhagen

Guy Desaulniers
École Polytechnique and GERAD, Montreal, Canada

Jacques Desrosiers
École des Hautes Études Commerciales (HEC) and GERAD, Montreal, Canada

Abstract

Given a Dantzig-Wolfe decomposition of an integer linear program, this note presents a general framework for formulating, on the original integer formulation, valid inequalities derived on the integer master problem. It is possible to model these inequalities by adding new variables and constraints to the original formulation. This clarifies the connection between variables in the original formulation and the decomposed formulation and makes it easier to show the correctness of valid inequalities for the master problem, and to show how the additional inequalities may give rise to an augmented subproblem. Examples on how to apply this framework are given for the vehicle routing problem with time windows, the edge coloring problem, and the cutting stock problem.

Keywords: Dantzig-Wolfe decomposition, column generation, cutting planes

1 Introduction

Branch-and-price is already established as the leading solution methodology for many large-scale integer programming problems. The use of cutting planes has received increasing attention, but has mostly been limited to specialized implementations within branch-and-cut-and-price algorithms. This paper presents a general framework on the use of cutting planes that are available on some integer master problem structures in the context of column generation.

It is a generalization and a formalization of the work initially presented in Jepsen et al. [12] (and later used in Desaulniers et al. [7]) in which, given a Dantzig-Wolfe decomposition of the vehicle routing problem with time windows (VRPTW), a subset of the Chvátal-Gomory rank-1 cuts derived from the set partitioning master problem formulation was successfully applied. This work was later extended in Petersen et al. [16] to include all Chvátal-Gomory rank-1 cuts. In both papers it is shown that the additional cutting planes lead to an augmentation of the subproblem. To be more specific, the subproblem which is a resource constrained shortest path problem and additional resources are added for each cut. The augmented subproblem indirectly indicates that there exist an augmented formulation of the original multi-commodity network flow model of the VRPTW that includes the Chvátal-Gomory rank-1 cuts based on the integer master problem formulation.

In Spoorendonk and Desaulniers [18] clique inequalities are applied to the VRPTW and the augmented number of resources due to the the additional cutting planes are handled in the labeling algorithm for the subproblem. Baldacci et al. [1] apply clique inequalities to the capacitated vehicle routing problem, but here the subproblem is solved by an enumeration scheme based on dynamic programming.

In Nemhauser and Park [14] one can find an early example of the use of cutting planes derived on the integer master problem of a column generation solution approach for the edge coloring problem. Inequalities are added in the master problem and modelled in the subproblem which is solved by branch-and-bound. Other examples are found in Belov and Scheithauer [3, 4] for various cutting stock problems. In the first paper, Chvátal-Gomory cuts of any rank are used on the one-dimensional cutting stock problem and the multiple length one-dimensional cutting stock problem. In the second, both Chvátal-Gomory cuts and Gomory mixed integer cuts are used on the one-dimensional cutting stock problem and the two-dimensional two-stage cutting problem. Cuts are added in the master problem and the subproblem is solved with a branch-and-bound algorithm using an approximation algorithm to calculate lower bounds but no connection is made to variables in the original solution space.

The emphasis in this note is solely on how to apply cutting planes derived on integer master problem formulations given a Dantzig-Wolfe decomposition of an integer program. Section 2 recaps the Dantzig-Wolfe decomposition and introduces the notation used throughout the remainder of the note. Section 3 introduces the general framework that describes an augmentation of the original formulation such that it includes the valid inequalities derived from the integer master problem formulation. Examples on how to apply and interpret the presented framework are provided in Section 4.

2 Dantzig-Wolfe Decomposition

Dantzig-Wolfe decomposition for linear programs was introduced by Dantzig and Wolfe [5]. Detailed surveys on column generation, branch-and-price, and branch-and-cut-and-price algorithms for the decomposition of integer linear programs may be found in Barnhart et al. [2], Lübbecke and Desrosiers [13]. Following is the Dantzig-Wolfe decomposition, almost as presented in Lübbecke and Desrosiers [13]. We have specifically chosen the *discretization version*, see Vanderbeck [20], as we want to apply general cuts to an integer master problem formulation. Consider the $|K|$-block-angular linear integer problem (IP):

$$\min \sum_{k\in K} c^k x^k \tag{1}$$

$$\text{s.t.} \sum_{k\in K} A^k x^k \leq b \tag{2}$$

$$D^k x^k \leq d^k \qquad k \in K \tag{3}$$

$$x^k \in \mathbb{Z}_+^{n_k} \qquad k \in K \tag{4}$$

This can be rewritten into the smaller (in the number of constraints) problem:

$$\begin{aligned} \min & \sum_{k\in K} c^k x^k \\ \text{s.t.} & \sum_{k\in K} A^k x^k \leq b \\ & x^k \in X^k, k \in K \end{aligned}$$

containing only the upper matrix blocks A^k for $k \in K$. However, x^k belongs to the domain $X^k = \{x^k \in \mathbb{Z}_+^{n_k} : D^k x^k \leq d^k\}$, that is, x^k must satisfy the constraint matrix block D^k to be part of the smaller problem. In Nemhauser and Wolsey [15] it is shown that, for each $k \in K$, there exits a finite set of integer points $\{x^{kp}\}_{p\in P^k}$ and a finite set of integer rays $\{x^{kr}\}_{r\in R^k}$ such that

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

Substituting this into IP, the following integer master problem (IMP) is obtained:

$$\min \sum_{k\in K}\Big(\sum_{p\in P^k} (c^k x^{kp})\lambda_{kp} + \sum_{r\in R^k} (c^k x^{kr})\lambda_{kr}\Big) \tag{5}$$

$$\text{s.t.} \sum_{k\in K}\Big(\sum_{p\in P^k} (A^k x^{kp})\lambda_{kp} + \sum_{r\in R^k} (A^k x^{kr})\lambda_{kr}\Big) \leq b \tag{6}$$

$$\sum_{p\in P^k} \lambda_{kp} = 1 \qquad k \in K \tag{7}$$

$$\lambda_{kp} \in \{0,1\} \qquad p \in P^k, k \in K \tag{8}$$

$$\lambda_{kr} \in \mathbb{Z}_+ \qquad r \in R^k, k \in K \tag{9}$$

IMP contains a smaller number of constraints than the original IP but it does have an exponential number of variables. The linear relaxation of IMP, denoted LMP, is often solved using column generation where π and $\mu^k, k \in K$ are the vector of dual variables for constraints (6) and (7), respectively. Only a subset of variables in LMP is considered and this gives rise to a restriction of LMP, denoted the restricted linear programming master problem (RLMP). Generating new columns is done by solving a subproblem for each of the domains X^k defined using information from the current solution of the RLMP, more precisely, the values of the dual variables π and μ^k. To find a column with the least reduced cost, the following subproblems for $k \in K$ are solved:

$$\min_{x\in X^k} (c^k - \pi A^k)x - \mu^k \tag{10}$$

If (10) is unbounded the solution is an integer ray in R^k; otherwise it provides a new integer point in P^k if the objective value is negative. The column generation process stops if the objective value is null for all subproblems $k \in K$.

Now consider a valid inequality for the original IP (1)-(4):

$$\sum_{k \in K} \alpha^k x^k \leq \beta$$

that is, a linear combination of the original x^k variables. Decomposed in LMP, the above inequality reads:

$$\sum_{k \in K} \Big(\sum_{p \in P^k} (\alpha^k x^{kp}) \lambda_{kp} + \sum_{r \in R^k} (\alpha^k x^{kr}) \lambda_{kr} \Big) \leq \beta \tag{11}$$

with the associated dual variable ν. Therefore adding valid inequalities directly from the original IP affects the coefficients in the objective function of the subproblems (10) which becomes:

$$\min_{x \in X^k} (c^k - \pi A^k - \nu \alpha^k) x - \mu^k$$

Consider next a valid inequality for the decomposed IMP (5)-(9):

$$\sum_{k \in K} \Big(\sum_{p \in P^k} \alpha^{kp} \lambda_{kp} + \sum_{r \in R^k} \alpha^{kr} \lambda_{kr} \Big) \leq \beta \tag{12}$$

with the dual variable σ in the corresponding linear relaxation. Then negative reduced cost columns are found by solving the following subproblems, for $k \in K$:

$$\min_{x \in X^k} (c^k - \pi A^k) x - \mu^k - \sigma \alpha^{kp} \tag{13}$$

As above, if (13) is unbounded the solution is an integer ray in R^k; otherwise it provides a new integer point in P^k if the objective value is negative. Column generation stops when all subproblems return a zero objective value.

Note, that when $\alpha^k x^{kp} = \alpha^{kp}$ and $\alpha^k x^{kr} = \alpha^{kr}$ then (11) is equivalent to (12). Otherwise the calculation of the last term $-\sigma \alpha^{kp}$ (or $-\sigma \alpha^{kr}$) in the objective function of the above subproblems may not be straightforward since α^{kp} (or α^{kr}) can depend on the constraint matrix A^k and the extreme point x^{kp} (or extreme ray x^{kr}). This is the focus of the remainder of this paper.

3 An Augmented IP Formulation

In the following we present a general method for describing in IP (1)-(4) valid inequalities derived from IMP (5)-(9). This is done through the introduction of new variables and constraints to the formulation of IP that are used to model the valid inequalities for IMP. We denote this the augmented IP formulation. The decomposition of the augmented IP formulation is equivalent with IMP where the valid inequalities are added. Also, the decomposition shows how the subproblems are augmented with regard to variables and constraints (from the additional variables and constraints in the augmented IP formulation).

Let $f(A^k, x^{kp})$ be a function that calculates the coefficient α^{kp} for λ_{kp} (function $f(A^k, x^{kr})$ calculates coefficient α^{kr} for λ_{kr}) given the constraints matrix A^k and an integer point x^{kp} (integer ray x^{kr}) of domain X^k. That is,

$$f(A^k, x^{kp}) = \alpha^{kp} \text{ and } f(A^k, x^{kr}) = \alpha^{kr}.$$

Let y^k be a variable in the augmented formulation of IP such that:

$$y^k = f(A^k, x^k) \qquad k \in K \tag{14}$$

meaning that $y^{kp} = f(A^k, x^{kp})$ and $y^{kr} = f(A^k, x^{kr})$ for extreme points and rays of the polytope of the kth block in the augmented IP. The valid inequality (12) derived in IMP can then be expressed in the augmented IP as:

$$\sum_{k \in K} y^k \leq \beta. \tag{15}$$

Let C be the index set of such valid inequalities derived for the IPM of the form:

$$\sum_{k \in K} \Big(\sum_{p \in P^k} f_i(A^k, x^{kp}) \lambda_{kp} + \sum_{r \in R^k} f_i(A^k, x^{kr}) \lambda_{kr} \Big) \leq \beta_i \qquad i \in C \tag{16}$$

With the use of (14) and (15) the augmented IP formulation is:

$$\min \sum_{k \in K} c^k x^k \tag{17}$$

$$\text{s.t.} \sum_{k \in K} A^k x^k \leq b \tag{18}$$

$$\sum_{k \in K} y_i^k \leq \beta_i \qquad i \in C \tag{19}$$

$$D^k x^k \leq d^k \qquad k \in K \tag{20}$$

$$y_i^k = f_i(A^k, x^k) \qquad i \in C, k \in K \tag{21}$$

$$y_i^k \in \mathbb{R} \qquad i \in C, k \in K \tag{22}$$

$$x^k \in \mathbb{Z}_+^{n_k} \qquad k \in K \tag{23}$$

The above augmented IP still has a $|K|$-block-angular structure with (18) and (19) acting as the linking constraints while the $|K|$ blocks apart from (20) also include new constraints (21) and variables (22). This leads to the following augmented IMP:

$$\min \sum_{k \in K} \Big(\sum_{p \in P^k} (c^k x^{kp}) \lambda_{kp} + \sum_{r \in R^k} (c^k x^{kr}) \lambda_{kr} \Big) \tag{24}$$

$$\text{s.t.} \sum_{k \in K} \Big(\sum_{p \in P^k} (A^k x^{kp}) \lambda_{kp} + \sum_{r \in R^k} (A^k x^{kr}) \lambda_{kr} \Big) \leq b \tag{25}$$

$$\sum_{k \in K} \Big(\sum_{p \in P^k} y_i^{kp} \lambda_{kp} + \sum_{r \in R^k} y_i^{kr} \lambda_{kr} \Big) \leq \beta_i \qquad i \in C \tag{26}$$

$$\sum_{p \in P^k} \lambda_{kp} = 1 \qquad k \in K \tag{27}$$

$$\lambda_{kp} \in \{0, 1\} \qquad p \in P^k, k \in K \tag{28}$$

$$\lambda_{kr} \in \mathbb{Z}_+ \qquad r \in R^k, k \in K \tag{29}$$

where the constraints (26) are a reformulation of the valid inequalities given by (16). Let $\sigma_i, i \in C$ be the dual variables for constraints (26) in the corresponding LMP. Then column generation is performed by solving the following subproblems, for $k \in K$, in replacement of (10):

$$\min_{x \in X^k} (c^k - \pi A^k)x - \mu^k - \sum_{i \in C} \sigma_i y_i \tag{30}$$

$$\text{s.t. } y_i = f_i(A^k, x) \qquad i \in C \tag{31}$$

$$y_i \in \mathbb{R} \qquad i \in C \tag{32}$$

The addition of variables and constraints in the augmented IP, and consequently in the subproblems, may invalidate special purpose algorithms that were previously used for solving (10), see e.g., Section 4.1. However, several examples exist where an augmented IP formulation and a harder subproblem proves worthwhile, e.g., Jepsen et al. [12] and Desaulniers et al. [7]. Note that functions $f_i(A^k, x^k)$ need not be linear in the subproblem, for example if dynamic programming is used to solve it. But for (31) to hold in the linear programming case, it may be necessary to add several constraints and/or variables per subproblem.

In case the new inequalities depend on other inequalities added previously, e.g., in the case of Chvátal-Gomory cuts of rank higher than 1, the suggested framework may be used iteratively. Alternatively the function $f(A^k, x)$ in the subproblem may be reformulated into $f_i(A^k, y_0, \ldots, y_{i-1}, x)$ such that the function includes the y-variables that are the coefficients of the previously added valid inequalities, see Section 4.4.

4 Applications

This section shows some examples of how to interpret and use the theory presented above.

4.1 Chvátal-Gomory rank-1 cuts for the VRPTW

This is an example of how the work presented in Jepsen et al. [12] and Petersen et al. [16] fits into the proposed framework. Note that the subset row inequalities considered in Jepsen et al. [12] (and in Desaulniers et al. [7]) can be written as Chvátal-Gomory rank-1 cuts with exactly three non-zero Chvátal-Gomory multipliers equal to $\frac{1}{2}$.

The VRPTW is defined on a directed graph $G = (V, A)$ with node set V and arc set A. It can be described as finding the set of least cost paths for the set K of vehicles originating from a depot split in a start o and a target o' depot such that the node set $N = V \setminus \{o, o'\}$ is covered exactly once. All nodes must be visited within a given time window $[a_i, b_i]$ for a node i, and a node i have a load d_i that accumulates for a vehicle k and may never exceed the capacity D_k. Vehicles can be left unused by using the zero cost arc (o, o'). Let the binary variable x_{ij}^k indicates the use of arc (i, j) by vehicle k and the variable t_i^k indicates the time of vehicle k at node i if vehicle k visits node i (and is otherwise irrelevant). Let $\delta^-(i)$ and $\delta^+(i)$ be the set of other end nodes for the in- and outgoing arcs of node i. Given the cost c_{ij}^k and the traveling time τ_{ij}^k of arc (i, j) for vehicle k, the three index formulation for the VRPTW with M being a big constant is:

$$\min \sum_{k\in K} \sum_{(i,j)\in A} c_{ij}^k x_{ij}^k \tag{33}$$

$$\text{s.t.} \sum_{k\in K} \sum_{j\in\delta^+(i)} x_{ij}^k = 1 \qquad i \in N \tag{34}$$

$$\sum_{j\in\delta^+(o)} x_{oj}^k = \sum_{i\in\delta^-(o')} x_{io'}^k = 1 \qquad k \in K \tag{35}$$

$$\sum_{j\in\delta^-(i)} x_{ji}^k - \sum_{j\in\delta^+(i)} x_{ij}^k = 0 \qquad i \in N, k \in K \tag{36}$$

$$\sum_{(i,j)\in A} d_i x_{ij}^k \leq D_k \qquad k \in K \tag{37}$$

$$t_i^k + \tau_{ij}^k - M(1 - x_{ij}^k) \leq t_j^k \qquad (i,j) \in A,\ k \in K \tag{38}$$

$$a_i \leq t_i^k \leq b_i \qquad i \in V,\ k \in K \tag{39}$$

$$x_{ij}^k \in \{0,1\} \qquad (i,j) \in A,\ k \in K \tag{40}$$

Where the objective function (33) minimizes the overall travel cost, constraints (34) ensures that nodes are visited exactly once, (35) ensures that all vehicles leaves and enters the depot once, (36) ensures flow conservation at the nodes for all vehicles, (37) ensures that the capacity of the vehicles are not exceeded, (38) and (39) ensures that nodes are visited within their time window, and (40) ensures that are either used or not.

Keeping (34) as the linking constraints and (35)-(40) in the subproblem constraint set X^k (separable for each $k \in K$), the above IP formulation decomposes into the following IMP (with no extreme rays):

$$\min \sum_{k\in K} \sum_{p\in P^k} c^{kp} \lambda_{kp} \tag{41}$$

$$\text{s.t.} \sum_{k\in K} \sum_{p\in P^k} \theta_i^{kp} \lambda_{kp} = 1 \qquad i \in N \tag{42}$$

$$\sum_{p\in P^k} \lambda_{kp} = 1 \qquad k \in K \tag{43}$$

$$\lambda_{kp} \in \{0,1\} \qquad p \in P^k, k \in K \tag{44}$$

where θ_i^{kp} is 1 if node i is on the path p covered by vehicle k and zero otherwise. In LMP, let π and μ be the dual vectors for constraints (42) and (43), respectively. The subproblems are elementary shortest path problems with resource constraints (ESPPRC) where the resources are load and time:

$$\min_{x\in X^k} \sum_{(i,j)\in A} c_{ij}^k x_{ij} - \sum_{i\in N} \pi_i \sum_{(i,j)\in\delta^+(i)} x_{ij} - \mu^k$$

A Chvátal-Gomory rank-1 cut for IMP on constraints (42) is written as:

$$\sum_{k\in K} \sum_{p\in P^k} \left\lfloor \sum_{i\in N} u_i \theta_i^{kp} \right\rfloor \lambda_{kp} \leq \left\lfloor \sum_{i\in N} u_i \right\rfloor$$

where $u_i \in [0,1[$ is the Chvátal-Gomory multiplier of the ith constraint of (42). The coefficient function $f(A^k, x^{kp})$ may be formulated as follow:

$$f(A^k, x^{kp}) = \left\lfloor \sum_{i \in N} u_i \sum_{j \in \delta^-(i)} x_{ji}^{kp} \right\rfloor$$

That is, for a set Ω of Chvátal-Gomory rank-1 cuts, and a small $\epsilon > 0$ (used to model strict less than), this can be modeled by adding the zero cost variables y_h^k for $h \in \Omega$ and $k \in K$ to the original VRPTW formulation (33)-(40) together with the following constraints:

$$\sum_{k \in K} y_h^k \leq \left\lfloor \sum_{i \in N} u_i^h \right\rfloor \qquad h \in \Omega \qquad (45)$$

$$\sum_{i \in N} u_i^h \sum_{j \in \delta^-(i)} x_{ji}^k - (1-\epsilon) \leq y_h^k \leq \sum_{i \in N} u_i^h \sum_{j \in \delta^-(i)} x_{ji}^k \qquad h \in \Omega, k \in K \qquad (46)$$

$$y_h^k \in \mathbb{Z} \qquad h \in \Omega, k \in K \qquad (47)$$

where (46) and the integrality requirement (47) model the floor function $f(A^k, x^{kp})$. In the augmented IMP (and LMP), the added inequalities (45) appears as:

$$\sum_{k \in K} \sum_{p \in P^k} (y_h^{kp}) \lambda_{kp} \leq \left\lfloor \sum_{i \in N} u_i^h \right\rfloor \qquad h \in \Omega$$

with dual vector σ. In the augmented subproblem $k \in K$ we have, besides $x \in X^k$, new variables y_h for each Chvátal-Gomory rank-1 cut with the cost $-\sigma_h$, and new constraints due to (46)-(47):

$$\begin{aligned} \min_{x \in X^k} \; & \sum_{(i,j) \in A} c_{ij}^k x_{ij} - \sum_{i \in N} \pi_i \sum_{(i,j) \in \delta^+(i)} x_{ij} - \mu^k - \sum_{h \in \Omega} \sigma_h y_h \\ \text{s.t.} \; & \sum_{i \in N} u_i^h \sum_{j \in \delta^-(i)} x_{ji} - (1-\epsilon) \leq y_h \leq \sum_{i \in N} u_i^h \sum_{j \in \delta^-(i)} x_{ji} \\ & y \in \mathbb{Z} \end{aligned}$$

In the VRPTW, the new variables y_h can be modelled as a resource such that the subproblem remains an ESPPRC. Dynamic programming algorithms denoted labeling algorithms are used to solve the ESPPRC. Let Ω be the set of Chvátal-Gomory rank-1 cuts and let a partial path be represented by a label $L = (\mathcal{T}_{cost}, (\mathcal{T}_{n_h})_{h \in N}, \mathcal{T}_D, \mathcal{T}_T, (\mathcal{T}_{c_h})_{h \in \Omega})$ consisting of the cost and all the resource consumptions of the partial path, i.e., $(\mathcal{T}_{n_h})_{h \in N}$ is the number of visits to the nodes (in this case 0 or 1 due to elementary), $\mathcal{T}_D$ is the load, $\mathcal{T}_T$ is the time, and $(\mathcal{T}_{c_h})_{h \in \Omega}$ are the Chvátal-Gomory rank-1 cut resources. Labels are extended from node to node via the connecting arc using *resource extension functions*, see e.g., Desaulniers et al. [6], Irnich [10] for a detailed description. In subproblem k extending label L on arc $(i,j) \in A$ to create label $\bar{L}$ (if feasible considering resource limits) is done by the following extension functions:

$$\begin{aligned}
\bar{\mathcal{T}}_{cost} &= \mathcal{T}_{cost} + c^k_{ij} - \pi_j - \sum_{\substack{h\in\Omega \\ \mathcal{T}_{c_h}+u^h_i\geq 1}} \sigma_h \\
\bar{\mathcal{T}}_{n_j} &= \mathcal{T}_{n_j} + 1 \\
\bar{\mathcal{T}}_{n_h} &= \mathcal{T}_{n_h} && h \in N \setminus \{j\} \\
\bar{\mathcal{T}}_{D} &= \mathcal{T}_{D} + d_j \\
\bar{\mathcal{T}}_{T} &= \max\{\mathcal{T}_T + \tau^k_{ij}, a_j\} \\
\bar{\mathcal{T}}_{c_h} &= (\mathcal{T}_{c_h} + u^h_i) \mod 1 && h \in \Omega
\end{aligned}$$

At a given node of the network, a label may dominate other labels using the following dominance criterion:

Proposition 1. (Desaulniers et al. [6]) Let L and $\bar{L}$ be two labels representing partial paths ending at the same node. Label L dominates label $\bar{L}$ (which can be discarded) if

$$\begin{aligned}
\bar{\mathcal{T}}_{cost} &\leq \mathcal{T}_{cost} \\
\bar{\mathcal{T}}_{r} &\leq \mathcal{T}_{r} && r \in \{(n_h)_{h\in N}, D, T, (c_h)_{h\in\Omega}\}
\end{aligned}$$

This dominance criterion has been improved in the case of Chvátal-Gomory rank-1 cut resources:

Proposition 2. (Petersen et al. [16]) Let L and $\bar{L}$ be two labels representing partial paths ending at the same node. Label L dominates label $\bar{L}$ (which can be discarded) if

$$\begin{aligned}
\bar{\mathcal{T}}_{cost} - \sum_{\substack{h\in\Omega \\ \mathcal{T}_{c_h}>\bar{\mathcal{T}}_{c_h}}} \sigma_h &\leq \mathcal{T}_{cost} \\
\bar{\mathcal{T}}_{r} &\leq \mathcal{T}_{r} && r \in \{(n_h)_{h\in N}, D, T\}
\end{aligned}$$

This last dominance criterion is a generalization of the one proposed in Jepsen et al. [12], but is equivalent when considering only the subset row inequalities.

Subproblem running time complexity: For the VRPTW, the subproblem (an ESPPRC) is strongly $\mathcal{NP}$-hard. Hence, adding additional resources does add complexity to the problem but exponential worst case running time is already expected. This is not the case when considering a relaxation of the subproblem such as the non-elementary version, the shortest path problem with resource constraints (SPPRC), that has been widely applied for the VRPTW, see e.g., Irnich and Villeneuve [11]. The SPPRC is weakly $\mathcal{NP}$-hard and can be solved in pseudo-polynomial time, but adding additional resources due to the Chvátal-Gomory rank-1 cuts may indeed increase this pseudo-polynomial factor.

Another example is taken from the multi-depot vehicle scheduling problem, see Riberio and Soumis [17], where the subproblems are shortest path problems in an acyclic graph and are therefore solvable in polynomial time. When cuts are added, the new subproblems become resource constrained, that is, SPPRCs (still on an acyclic graph) and the running time complexity goes from polynomial to pseudo-polynomial.

Special cases: As already indicated at the end of Section 2 the framework presented here includes the formulation of cuts as expressed in (11), that is, in terms of the original variables of IP. Let

$$f(A^k, x^{kp}) = \theta^k x^{kp} \text{ and } f(A^k, x^{kr}) = \theta^k x^{kr}.$$

Then (14) and (15) provides an equivalent formulation of (11). This can easily be verified by substitution of the f-function and the y-variables in (19) and (21). Two examples in the area of vehicle routing are the rounding cut on the total cost and the cut on the total number of vehicles used, see Desrosiers et al. [8]:

$$f_1(A^k, x^{kp}) = \sum_{(i,j)\in A^k} c^k_{ij} x^{kp}_{ij}$$

$$f_2(A^k, x^{kp}) = \sum_{(o,j)\in A^k} x^{kp}_{oj}$$

A third example is the one used in Gamache et al. [9], a cut on the cost of a pilot schedule in the preferential bidding system, i.e., a cut on the cost of a single block k of the diagonal constraint set. The f-function of other indices than the kth returns 0 in this application. Again the y variable can be modeled as an additional resource variable, indeed the actual cost of the generated path (not its reduced cost). Such a cut is formulated with an f-function and a y-variable using (14) and (15) in the IP augmented formulation, transferred into the kth subproblem, and directly controls the cost of the generated columns. This cut is extremely strong and in some cases closes integrality gaps up to 99%.

4.2 Clique inequalities for the VRPTW

In Spoorendonk and Desaulniers [18] it is shown how clique inequalities can be applied. Recall the definition of the VRPTW given in Section 4.1. For a given clique W of paths, the clique inequality defined on constraints (42) of the integer master problem (41)-(44) is:

$$\sum_{k\in K}\sum_{p\in P^k} \theta^{kp}\lambda_{kp} \leq 1$$

where θ^{kp} is 1 if path $p \in W$ and zero otherwise. Spoorendonk and Desaulniers [18] suggest an approximate description of the clique denoted a *key set* that can easily be interpreted in a column generation context. Let $\chi(W) \subseteq N$ be the set of conflicting rows (constraints (42)) with respect to the paths in W, i.e., rows where at least two paths in W have a nonzero coefficient (the paths visit the same customer), and let the *minimal* subset $\chi_{min}(W) \subseteq \chi(W)$ be the subset of conflicting rows such that no row can be disregarded when all columns in the clique must conflict. It is observed that, given a new path p that conflicts with all paths in W when only considering $\chi_{min}(W)$, then p is also in conflict with all paths in W when considering all rows in N. Hence, a new path p can enlarge W if it conflicts with all paths in W with respect to the rows in $\chi_{min}(W)$.

The approximate representation of a clique W based on a minimal subset $\chi_{min}(W)$ is now available. It is composed of the vectors $\boldsymbol{v}_p(\chi_{min}(W))$, $\forall p \in W$, where $\boldsymbol{v}_p(\chi_{min}(W))$ is the subvector of the column resulting from p restricted to the rows in $\chi_{min}(W)$. This representation can be made more compact by noticing that a newly generated path is in conflict with a path $p \in W$ if it is also conflicting with a path $q \in W$, $q \neq p$, such that

$\boldsymbol{v}_q(\chi_{min}(W)) \leq \boldsymbol{v}_p(\chi_{min}(W))$. It is easy to prove that there are no pairs of paths p and q in W such that $\boldsymbol{v}_q(\chi_{min}(W)) < \boldsymbol{v}_p(\chi_{min}(W))$ (otherwise, $\chi_{min}(W)$ would not be a minimal subset of conflicting rows of W). Consequently, one can eliminate from the representation all vectors $\boldsymbol{v}_p(\chi_{min}(W))$, except one, that are equal. For a clique W, we denote by $\Gamma(W)$ the resulting set of paths used in the representation. This representation, called a key set, is defined as follows.

Definition 1. (Spoorendonk and Desaulniers [18]) Given a clique W and a minimal set $\chi_{min}(W)$ of its conflicting rows, the *key set* of W is composed of the subvectors $\boldsymbol{v}_p(\chi_{min}(W))$, $\forall p \in \Gamma(W)$.

A newly generated path p is added in a clique W only if $\boldsymbol{v}_p(\chi_{min}(W)) \geq \boldsymbol{v}_q(\chi_{min}(W))$ for at least one path $q \in \Gamma(W)$. This restriction guarantees that all added paths conflict with each other (since they all contain at least one of the key set subvectors which all conflict), and allows to always keep the same key set as new paths are added to the clique. A path p satisfying this check is said to be *admissible to enlarge W*.

Given the key set of W, the coefficient function $f(A^k, x^{kp})$ may be formulated as:

$$f(A^k, x^{kp}) = \max_{q\in\Gamma(W)} \left\lfloor \sum_{i\in\chi_{min}(W)} \frac{\boldsymbol{v}_{qi}(\chi_{min}(W))}{\sum_{j\in\chi_{min}(W)} \boldsymbol{v}_{qj}(\chi_{min}(W))} \sum_{j\in\delta^-(i)} x^{kp}_{ji} \right\rfloor$$

where $\boldsymbol{v}_{qi}(\chi_{min}(W))$ is the ith entry in subvector $\boldsymbol{v}_q(\chi_{min}(W))$. Let Ω be the set of cliques, then the VRPTW formulation (33)-(40) is augmented with the non-zero cost variables y^K_W for $W \in \Omega$ and $k \in K$ and the constraints:

$$\sum_{k\in K} y^k_W \leq 1 \qquad W \in \Omega \qquad (48)$$

$$\sum_{i\in\chi_{min}(W)} \boldsymbol{v}_{qi}(\chi_{min}(W)) \left(\sum_{j\in\delta^-(i)} x^k_{ji} - 1 \right) + 1 \leq y^k_W \qquad q \in \Gamma(W), W \in \Omega, k \in K \qquad (49)$$

$$y^k_W \in \{0,1\} \qquad W \in \Omega, k \in K \qquad (50)$$

where constraints (49) and (50) model the maximization term of $f(A^k, x^{kp})$. In the IMP (41)-(44) constraints (48) decompose into:

$$\sum_{k\in K} \sum_{p\in P^k} (y^{kp}_W)\lambda_{pk} \leq 1 \qquad W \in \Omega$$

with dual vector σ_W. In the augmented subproblem, we have new variables y_W for the cliques with costs $-\sigma_W$ and new constraints due to (49)-(50):

$$\min_{x\in X^k} \sum_{(i,j)\in A} c^k_{ij} x_{ij} - \sum_{i\in N} \pi_i \sum_{(i,j)\in\delta^+(i)} x_{ij} - \mu^k - \sum_{W\in\Omega} \sigma_W y_W$$

$$\text{s.t.} \sum_{i\in\chi_{min}(W)} \boldsymbol{v}_{qi}(\chi_{min}(W)) \left(\sum_{j\in\delta^-(i)} x^k_{ji} - 1 \right) + 1 \leq y_W \qquad q \in \Gamma(W), W \in \Omega$$

$$y_W \in \{0,1\} \qquad W \in \Omega$$

It is possible to implement the additional constraints using resources in the constrained shortest path subproblem. The first requirement consists of adding new components to the labels that enable to determine if a path is admissible to enlarge a clique. For each clique $W \in \Omega$, define $|\chi_{min}(W)|+1$ binary resources and add the corresponding components to each label. For a label L representing a partial path p, the first $|\chi_{min}(W)|$ of these resource values indicate, until it is proven that p can enlarge W, whether or not each customer of the minimal subset $\chi_{min}(W)$ has been visited along p. Thus, they represent the subvector $\boldsymbol{v}_p(\chi_{min}(W))$ and are compared to the key set vectors of W to determine if p is admissible to enlarge W. If the check is positive, these resource values are set to 0 and never change in the subsequent label extensions. Furthermore, the dual value σ_W is subtracted from the label cost component. For a label L, these resource components are denoted $\mathcal{T}^W_{n_i}$, $i \in \chi_{min}(W)$. The other additional resource component, denoted $\mathcal{T}^W_{inadm}$, simply indicates whether or not p is inadmissible to enlarge the clique: it is equal to 1 if p is inadmissible, and 0 otherwise. The set of all these new resources for all cliques in Ω is denoted $\mathcal{Q}$.

With these additional resources, the labeling algorithm can be applied for solving exactly the subproblem if the dominance criterion of Proposition 1 considers the set $\{(n_h)_{h\in N}, D, T, (c_h)_{h\in C}\} \cup \mathcal{Q}$. However, as for the Chvátal-Gomory rank-1 cuts it is possible to improve on this dominance criterion.

Proposition 3. (Spoorendonk and Desaulniers [18]) Let L and $\bar{L}$ be two labels representing partial paths ending at the same node. Label L dominates label $\bar{L}$ (which can be discarded) if

$$\begin{aligned}
\bar{\mathcal{T}}_{cost} - \sum_{W\in\Omega_{L\bar{L}}} \sigma_W &\le \mathcal{T}_{cost} \\
\bar{\mathcal{T}}_r &\le \mathcal{T}_r \qquad\qquad \forall r \in \{(n_h)_{h\in N}, D, T, (c_h)_{h\in C}\},
\end{aligned}$$

where $\Omega_{L\bar{L}} = \{W \in \Omega : \bar{\mathcal{T}}^W_{inadm} = 1 \text{ and } \big(\bar{\mathcal{T}}^W_{inadm} = 0 \text{ or } \exists\, i \in \chi_{min}(W) \text{ such that } \mathcal{T}^W_{n_i} > \bar{\mathcal{T}}^W_{n_i}\big)\}$ is the set of cliques W for which the penalty σ_W could be paid in a feasible extension of L along an arc sequence, while it would not be paid when extending similarly $\bar{L}$.

Again, as with the Chvátal-Gomory rank-1 cuts, the checking of whether or not a label is in a clique slows down the labeling algorithm.

4.3 Odd circuit constraints in the edge coloring problem

This example is from the paper by Nemhauser and Park [14]. The edge coloring problem on the graph $G = (V, E)$ with the set of colors K is to find the minimum number of colors used to color G such that no adjacent edges have the same color, i.e., no two edges with a common end node can have the same color. A solution can also be seen as a set of matchings with different colors where a matching is a set of edges that do not share any end points.

Let z^k be the binary variable indicating if color k is used by some edges or not, and let x^k_e be the binary variable indicating if edge e is colored by the color k. Let $\delta(i)$ be the set of edges connected to i, and let $E(S)$ be the set of edges with both end points in the node set

$S \subseteq V$. The edge coloring problem on the graph G with the set of colors K is given as:

$$\min \sum_{k \in K} z^k \tag{51}$$

$$\text{s.t.} \sum_{k \in K} x_e^k \geq 1 \qquad e \in E \tag{52}$$

$$\sum_{e \in \delta(i)} x_e^k \leq 1 \qquad i \in V, k \in K \tag{53}$$

$$|E| z^k \geq \sum_{e \in E} x_e^k \qquad k \in K \tag{54}$$

$$z^k \in \{0,1\} \qquad k \in K \tag{55}$$

$$x_e^k \in \{0,1\} \qquad e \in E, k \in K \tag{56}$$

Where the objective function (51) minimizes the number of colors, constraints (52) ensures that all edges are assigned at least one color, (53) ensures that no adjacent edges have the same color, (54) ensures that the color variable is set if any edges are assigned that color, and (55) and (56) define the domain of the variables.

Let (52) be the linking constraints and let (53)-(56) separable by $k \in K$ be in the subproblem constraint set X^k. Then the above IP decomposes into the IMP:

$$\min \sum_{k \in K} \sum_{m \in M^k} (z^{km}) \lambda_{km} \tag{57}$$

$$\text{s.t.} \sum_{k \in K} \sum_{m \in M^k} (x_e^{km}) \lambda_{km} \geq 1 \qquad e \in E \tag{58}$$

$$\sum_{m \in M^k} \lambda_{km} = 1 \qquad k \in K \tag{59}$$

$$\lambda_{km} \in \{0,1\} \qquad m \in M^k, k \in K \tag{60}$$

where M^k is the set of matchings, i.e., integer points, for subproblem k. Given the dual vectors π and μ in LMP for constraints (58) and (59), respectively, the subproblem $k \in K$ is:

$$\min_{x \in X^k} z - \sum_{e \in E} \pi_e x_e - \mu^k \tag{61}$$

For $U \subseteq V$ a matching can cover no more than $\lfloor \frac{1}{2}|U| \rfloor$ edges from $E(U)$. The matching constraints are given as:

$$\sum_{k \in K} \sum_{m \in M^k} \theta^{km} \lambda_{km} \geq \left\lceil \frac{|E'|}{\lfloor \frac{1}{2}|U| \rfloor} \right\rceil \qquad U \subseteq V, E' \subseteq E(U)$$

where

$$\theta^{km} = \begin{cases} 1 & \text{if } E(m) \cap E' \neq \emptyset \\ 0 & \text{otherwise} \end{cases}$$

for the set of edges $E(m)$ of the matching m. For a circuit given as a subgraph $C = (U, E')$ where $|U|$ is odd, the *odd circuit* constraint is given as:

$$\sum_{k \in K} \sum_{m \in M^k} \theta^{km} \lambda_{km} \geq 3$$

Given an odd circuit C we have the coefficient function:

$$f(A^k, z^{km}, x^{km}) = \min \left\{ 1, \sum_{e \in C} x_e^{km} \right\}$$

That is, given a set of odd circuit constraints Ω, the additional constraints to the edge coloring problem (51)-(56) are:

$$\sum_{k \in K} y_C^k \geq 3 \qquad C \in \Omega \qquad (62)$$

$$y_C^k - \sum_{e \in C} x_e^k \leq 0 \qquad C \in \Omega, k \in K \qquad (63)$$

$$y_C^k \in \{0, 1\} \qquad C \in \Omega, k \in K \qquad (64)$$

where constraints (63) and (64) model $f(A^k, z^{km}, x^{km})$. In the IMP (57)-(60), constraints (62) decompose into:

$$\sum_{k \in K} \sum_{m \in M^k} (y_C^{km}) \lambda_{km} \geq 3 \qquad C \in \Omega$$

with dual vector σ. In the integer subproblem (61) we have the additional variables y_C for the circuits with costs $-\sigma_C$ and the new constraints due to (63)-(64):

$$\begin{aligned} \min_{x \in X^k} \ & z - \sum_{e \in E} \pi_e x_e - \mu^k - \sum_{C \in \Omega} \sigma_C y_C \\ \text{s.t.} \ & y_C - \sum_{e \in C} x_e \leq 0 \qquad && C \in \Omega \\ & y_C \in \{0, 1\} && C \in \Omega \end{aligned}$$

As mentioned in the introduction, in Nemhauser and Park [14] the augmented subproblem is solved to optimality with a branch-and-bound algorithm.

4.4 Chvátal-Gomory rank $H \geq 1$ cuts for the one-dimensional cutting stock problem

Belov and Scheithauer [3] have shown how to apply Chvátal-Gomory cuts of rank higher than or equal to one for the one-dimensional cutting stock problem.

The one-dimensional cutting stock problem is to minimize the number of stocks of a given length used to produce a certain amount of small items of a given width. Valério de Carvalho [19] presented an arc flow model for this problem. Let B be the stock length and let I be the set of items, each with a demand b_i and a width w_i. Let $G = (V, A)$ be a directed graph with $B+1$ nodes in $V = \{0, 1, \ldots, B\}$ and arc set $A = \{(i, j) : 0 \leq i \leq j \leq B \text{ and } j - i = w_l, l \in I\}$. Hence, there is an arc between two nodes if there is an item of the corresponding size. Let z be the integer variable determining the number of stocks used, and x_{ij} be an integer variable contributing to the amount of item l with size $j - i = w_l$. Let $\delta^-(i)$ and $\delta^+(i)$ be the other end nodes for the in- and outgoing arcs of node $i \in V$. The arc flow model is given as:

$$\min z \tag{65}$$

$$\text{s.t.} \sum_{(j,j+w_i)\in A} x_{j,j+w_i} = b_i \qquad i \in I \tag{66}$$

$$\sum_{\delta^-(i)} x_{ji} - \sum_{\delta^+(i)} x_{ij} = 0 \qquad i \in V \setminus \{0, B\} \tag{67}$$

$$\sum_{j\in\delta^+(0)} x_{0j} = \sum_{i\in\delta^-(B)} x_{iB} = z \tag{68}$$

$$x_{ij} \in \mathbb{Z}_+ \qquad (i,j) \in A \tag{69}$$

$$z \in \mathbb{Z}_+ \tag{70}$$

where the objective function (65) minimizes the number of stocks, constraints (66) ensures that the demand of each item is met, (67) ensures flow conservation in the nodes meaning that the cutting patterns are connected such that all waste is at the end of stock, (68) ensures that any stock initiated for cutting is also completed (i.e., each integer amount of flow in the graph represents a stock being cut), and (69) and (70) define the domains of the variables.

Leaving (66) as the linking constraints and constraints (67)-(70) as the subproblem constraint set X, the above IP decomposes into the IMP that is the well-known Gilmore-Gomory model:

$$\min \sum_{p\in P} \lambda_p \tag{71}$$

$$\text{s.t.} \sum_{p\in P} \theta_i^p \lambda_p = b_i \qquad i \in I \tag{72}$$

$$\lambda_p \in \mathbb{Z}_+ \qquad p \in P \tag{73}$$

where P is the set of patterns (extreme rays) and $\theta_i^p = \sum_{(j,j+w_i)\in A} x_{j,j+w_i}^p$, i.e., the number of times item i is in pattern $p \in P$. Given the dual vector π in LMP for constraints (72), the subproblem is:

$$\min 1 - \sum_{i\in I} \pi_i \sum_{(j,j+w_i)\in A} x_{j,j+w_i} \tag{74}$$

$$\text{s.t.} \sum_{\delta^-(i)} x_{ji} - \sum_{\delta^+(i)} x_{ij} = 0 \qquad i \in V \setminus \{0, B\} \tag{75}$$

$$\sum_{j\in\delta^+(0)} x_{0j} = \sum_{i\in\delta^-(B)} x_{iB} = 1 \tag{76}$$

$$x_{ij} \in \{0,1\} \qquad (i,j) \in A \tag{77}$$

Note that patterns with zero items are not generated, hence the 1 in the objective function of the subproblem and as the cost coefficient of the λ variables. Also there are no convexity constraint in the IMP (71)-(73) since all patterns with a non-zero number of items are extreme rays of the subproblem. Finally, a single unit of flow is sufficient in (76) to define an extreme ray. Subproblem (74)-(77) is a shortest path problem in the acyclic graph G, but it may also be reformulated into a 0-1 knapsack problem, i.e., let $\theta_i^p = x_i^p$ for the subproblem $\min\{1 - \sum_{i\in I} \pi_i x_i : \sum_{i\in I} w_i x_i \leq B, x_i \in \mathbb{Z}_+, i \in I\}$.

Let Ω be the set of Chvátal-Gomory cuts for the IMP (71)-(73). The cuts of higher rank depends on the lower ranked cuts, hence the coefficient of a variable is calculated as a recursive function. Let $\Omega_h \subset \Omega$ be the cuts of lower rank than cut $h \in \Omega$, let Φ_i^p be the coefficient of variable p for cut $i \in \Omega_h$, and let ζ_i be the right-hand side of cut i. The recursion of Φ_h^p and ζ_h is given as:

$$\Phi_h^p = \left\lfloor \sum_{i\in I} u_i \theta_i^p + \sum_{i\in\Omega_h} u_i \Phi_i^p \right\rfloor , \qquad \zeta_h = \left\lfloor \sum_{i\in I} u_i b_i + \sum_{i\in\Omega_h} u_i \zeta_i \right\rfloor$$

for the Chvátal-Gomory multipliers $-1 < u_i < 1, i \in I \cap \Omega_h$. A Chvátal-Gomory cut $h \in \Omega$ can be formulated as:

$$\sum_{p\in P} \left\lfloor \sum_{i\in I} u_i \theta_i^p + \sum_{i\in\Omega_h} u_i \Phi_i^p \right\rfloor \lambda_p \le \left\lfloor \sum_{i\in I} u_i b_i + \sum_{i\in\Omega_h} u_i \zeta_i \right\rfloor .$$

The coefficient function for the cut $h \in \Omega$ is given as:

$$f_h(A, x^p) = \left\lfloor \sum_{i\in I} u_i \theta_i^p + \sum_{i\in\Omega_h} u_i \Phi_i^p \right\rfloor .$$

Belov and Scheithauer [3] propose an approach where $f_h(A, x^p)$ is substituted into the objective function of the subproblem, leading to the non-linear problem (for the dual vector σ associated with the cuts Ω and constraints (75)-(77) defining the polytope X):

$$\min_{x\in X} 1 - \sum_{i\in I} \pi_i \sum_{(j,j+w_i)\in A} x_{j,j+w_i} - \sum_{h\in\Omega} \sigma_h \Phi_h \tag{78}$$

To solve this problem the non-linear objective function of (78) is approximated by a linear function and used as a lower bound in a branch-and-bound algorithm.

Alternatively, $f_h(A, x^p)$ can be linearized in a similar fashion as for the Chvátal-Gomory rank-1 cuts presented in Section 4.1. Consider a discretized (into $|K|$ blocks) but equivalent model of the arc flow model (65)-(70):

$$\begin{aligned}
\min\ & \sum_{k\in K} z^k && && (79)\\
\text{s.t.}\ & \sum_{k\in K} \sum_{(j,j+w_i)\in A} x_{j,j+w_i}^k = b_i && i \in I && (80)\\
& \sum_{\delta^-(i)} x_{ji}^k - \sum_{\delta^+(i)} x_{ij}^k = 0 && i \in V \setminus \{0, B\}, k \in K && (81)\\
& \sum_{j\in\delta^+(0)} x_{0j}^k = \sum_{i\in\delta^-(B)} x_{iB}^k = z^k && k \in K && (82)\\
& x_{ij}^k \in \{0,1\} && (i,j) \in A, k \in K && (83)\\
& z^k \in \{0,1\} && k \in K && (84)
\end{aligned}$$

The decomposition results in the same IMP (71)-(73) and $k \in K$ identical subproblems that are equal to the subproblem (74)-(77). The set Ω of Chvátal-Gomory cuts is modeled in the discretized arc flow formulation (79)-(84) with the use of a small $\epsilon > 0$ and the variables y_h^k. The coefficient θ_i^k and the recursions of Φ_j^k and ζ_i^k are equivalent with those given for the extreme ray p, but now considering variables in the kth block. The additional constraints are:

$$\sum_{k\in K} y_h^k \leq \left\lfloor \sum_{i\in I} u_i b_i + \sum_{i\in\Omega_h} u_i \zeta_i \right\rfloor \qquad h\in\Omega \qquad (85)$$

$$\sum_{i\in I} u_i\theta_i^k + \sum_{i\in\Omega_h} u_i\Phi_i^k - (1-\epsilon) \leq y_h^k \leq \sum_{i\in I} u_i\theta_i^k + \sum_{i\in\Omega_h} u_i\Phi_i^k \qquad k\in K, h\in\Omega \qquad (86)$$

$$y_h^k \in \mathbb{Z}_+ \qquad h\in\Omega \qquad (87)$$

Note that, since all subproblems are identical (the set of feasible patterns $P^k = P$ for all $k \in K$), only one subproblem needs to be solved, and it suffices to consider variable y_h in the subproblem and its value (y_h^p) for column p. Hence, constraints (85) are formulated in the augmented IMP as:

$$\sum_{p\in P} (y_h^p)\lambda_p \leq \left\lfloor \sum_{i\in I} u_i b_i + \sum_{i\in\Omega_h} u_i\zeta_i \right\rfloor \qquad h\in\Omega$$

with dual vector σ. The subproblem, besides constraints (75)-(77) defining the polytope X, is:

$$\begin{aligned} \min_{x\in X}\ & 1 - \sum_{(j,j+w_i)\in A} \pi_i x_{j,j+w_i} - \sum_{h\in\Omega} \sigma_h y_h \\ \text{s.t.}\ & \sum_{i\in I} u_i\theta_i + \sum_{i\in\Omega_h} u_i\Phi_i - (1-\epsilon) \leq y_h \leq \sum_{i\in I} u_i\theta_i + \sum_{i\in\Omega_h} u_i\Phi_i && h\in\Omega \\ & y_h \in \mathbb{Z} && h\in\Omega \end{aligned}$$

It is possible to model the subproblem as an SPPRC with additional resources for each cut (again in a similar fashion as for the VRPTW). However, it is imperative that the resources are updated in increasing order of rank due to the recursive nature of the coefficients.

5 Final Remarks

The framework presented in this paper provides a fundamental insight on the use of cutting planes in branch-and-cut-and-price algorithms. A detailed description of the connection between cutting planes in the decomposed master problem formulation and the original formulation is given. Such a connection is very important for understanding the use of cutting planes (and branching methods) and makes it possible to verify the correctness of the cutting planes in a reliable fashion.

The authors hope that this framework encourages the development of general purpose cutting planes (such as the Chvátal-Gomory cuts) and combinatorial cuts (such as the clique inequalities) for both the original and the master problem formulations. Indeed we should be delighted that there are two polytopes to derive cuts from. Of course one must keep in mind that there is a trade off by considering an augmented integer formulation that is of higher dimension than the original problem.

References

[1] R. Baldacci, N. Christofides, and A. Mingozzi. An exact algorithm for the vehicle routing problem based on the set partitioning formulation with additional cuts. *Mathematical Programming*, 115(2):351–385, 2008. doi: 10.1007/s10107-007-0178-5.

[2] 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(3):316–329, 1998. doi: 10.1287/opre.46.3.316.

[3] G. Belov and G. Scheithauer. A cutting plane algorithm for the one-dimensional cutting stock problem with multiple lengths. *European Journal of Operations Research*, 141(2): 274–294, 2002. doi: 10.1016/S0377-2217(02)00125-X.

[4] G. Belov and G. Scheithauer. A branch-and-cut-and-price algorithm for one-dimensional stock cutting and two-dimensional two-stage cutting. *European Journal of Operations Research*, 171(1):85–106, 2006. doi: 10.1016/j.ejor.2004.08.036.

[5] G. B. Dantzig and P. Wolfe. Decomposition principle for linear programs. *Operations Research*, 8(1):101–111, 1960. doi: 10.1287/opre.8.1.101.

[6] G. Desaulniers, J. Desrosiers, J. Ioachim, I. M. Solomon, F. Soumis, and D. Villeneuve. A unified framework for deterministic time constrained vehicle routing and crew scheduling problems. In T. G. Crainic and G. Laporte, editors, *Fleet Management and Logistics*, pages 57–93. Kluwer, 1998.

[7] G. Desaulniers, F. Lessard, and A. Hadjar. Tabu search, partial elementarity, and generalized k-path inequalities for the vehicle routing problem with time windows. *Transportation Science*, 42(3):387–404, 2008. doi: 10.1287/trsc.1070.0223.

[8] J. Desrosiers, F. Soumis, and M. Desrochers. Routing with time windows by column generation. *Networks*, 14(4):545–565, 1984. doi: 10.1002/net.3230140406.

[9] M. Gamache, F. Soumis, D. Villeneuve, J. Desrosiers, and E. Gélinas. The preferential bidding system at air canada. *Transportation Science*, 32(3):246–255, 1998. doi: 10.1287/trsc.32.3.246.

[10] S. Irnich. Resource extension functions: Properties, inversion, and generalization to segments. *OR Spectrum*, 30(1):113–148, 2008. doi: 10.1007/s00291-007-0083-6.

[11] S. Irnich and D. Villeneuve. The shortest path problem with resource constraints and k-cycle elimination for $k \geq 3$. *INFORMS Journal on Computing*, 18:391–406, 2006. doi: 10.1287/ijoc.1040.0117.

[12] M. 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. doi: 10.1287/opre.1070.0449.

[13] M. E. Lübbecke and J. Desrosiers. Selected topics in column generation. *Operations Research*, 53(6):1007–1023, 2005. doi: 10.1287/opre.1050.0234.

[14] G. Nemhauser and S. Park. A polyhedral approach to edge coloring. *Operations Research Letters*, 10(6):315–322, 1991. doi: 10.1016/0167-6377(91)90003-8.

[15] G. L. Nemhauser and L. A. Wolsey. *Integer and Combinatorial Optimization.* John Wiley & Sons, Inc., 1988.

[16] B. Petersen, D. Pisinger, and S. Spoorendonk. Chvátal-Gomory rank-1 cuts used in a Dantzig-Wolfe decomposition of the vehicle routing problem with time windows. In B. Golden, R. Raghavan, and E. Wasil, editors, *The Vehicle Routing Problem: Latest Advances and New Challenges*, pages 397–420. Springer, 2008. doi: 10.1007/978-0-387-77778-8_18.

[17] C. Riberio and F. Soumis. A column generation approach to the multiple-depot vehicle scheduling problem. *Operations Research*, 42(1):41–53, 1994. doi: 10.1287/opre.42.1.41.

[18] S. Spoorendonk and G. Desaulniers. Clique inequalities applied to vehicle routing problem with time windows. 2008.

[19] J.M. Valério de Carvalho. Exact solution of the bin-packing problems using column generation and branch-and-bound. *Annals of Operation Research*, 86:629–659, 1999. doi: 10.1023/A:1018952112615.

[20] F. Vanderbeck. On Dantzig-Wolfe decomposition in integer programming and ways to perform branching in a branch-and-price algorithm. *Operation Research*, 48(1):111–128, 2000. doi: 10.1287/opre.48.1.111.12453.

Part III

Conclusion

Chapter 7

Conclusion

Simon Spoorendonk
DIKU Department of Computer Science, University of Copenhagen

1 Summing Up

This thesis has investigated how to use cutting planes in branch-and-cut-and-price (BCP) algorithms. Experimental results are reported for the vehicle routing problem with time windows (VRPTW), where cuts based on the master problem formulation of a Dantzig-Wolfe decomposition have been applied. This includes both the general purpose mixed integer programming cutting planes known as Chv́atal-Gomory cuts and the combinatorial derived clique inequalities that are well known strong valid inequalities for the set partitioning polytope. These investigations spurred a general framework, that in details describe how to apply cutting planes in BCP algorithms, and how cutting planes derived from the master problem may be interpreted in the original problem by augmenting it.

In Chapter 3 and Chapter 4 it was shown how the Chv́atal-Gomory cuts can be applied to a decomposition model of the VRPTW. In the former paper, it was shown how a small subset of the Chv́atal-Gomory cuts, denoted subset-row inequalities, can be applied in the set partitioning master problem, and how to incorporate their dual costs into the pricing problem (the elementary shortest path problem) with the use of additional resources. This algorithm was at the time of publication, the most successful exact solution method for the VRPTW. In the latter paper, these results were extended to include all Chv́atal-Gomory cuts of rank 1. However, a slightly more complicated dominance criterion made the pricing problem much harder to solve in practice when more general Chv́atal-Gomory cuts than the subset-row inequalities were considered. Running times could not be improved compared to the former approach, but on several occasions it was possible to close the optimality gap completely in the root node. This indicates, that there is a potential gain since the number of nodes in the branch-and-bound tree is reduced, meaning that hopefully fewer pricing problems must be solved to optimality.

Chapter 5 extended the work on VRPTW by applying clique inequalities to the master problem. Again, the dual costs of the cutting planes can be handled with additional resources in the pricing problem. However, this time a resource is needed for each of the conflicting rows in the clique when such a cut is applied. In the paper it was shown how a reduced set of conflicting rows can be used to represent a clique. Obviously this leads to a potentially much harder pricing problem, and computational results also indicated that this approach was not competitive with an algorithm considering only the subset-row inequalities.

Chapter 6 summed up the experiences gained in the previous chapters and presents a general framework, that describes the correspondence between cutting planes in the master problem and original formulation. The main insight is, that cutting planes in master problem can be expressed in an augmented original formulation where the addition of variables and constraints are used to model the cuts. A decomposition of the augmented original problem is then shown to be equivalent with a master problem containing the cutting planes, and the new pricing problems are augmentations of the pricing problems without any cuts present. In case the cutting planes can be expressed solely with the use of the variables from the original problem, there is no use for additional variables in the original problem or in the pricing problems. Examples of the application of the framework is given for Chvátal-Gomory rank 1 cuts and clique inequalities for the VRPTW, odd circuits constraints for the edge coloring problem, and Chvátal-Gomory cuts of arbitrary rank for the 1-dimensional cutting stock problem. This framework presents a novel way of interpreting cutting planes in BCP algorithms, and can hopefully be used to ease the development of future cutting planes for BCP algorithms.

When applying Chvátal-Gomory cuts of arbitrary rank, it should be mentioned that several experiments were carried out during this thesis for the 1-dimensional cutting stock problem, the 2-dimensional vector packing problem, and the generalized assignment problem. However, none of the experiments were particularly successful and suggests that some problems are not suitable for cutting planes derived in the master problem. A reason may be, that the objective coefficients in these problems seldom differ much (e.g., all stocks have the cost 1 in the cutting stock problem), resulting in an optimality gap that often is a very small integer value. A valid lower bound is obtained by rounding up the LP solution value, and when applying cuts it may only result in a slight increase in the fractional value and not lead to an increased lower bound. Another noticeable thing is, that although the pricing problem (a knapsack problem) is weakly $\mathcal{NP}$-hard it does not seem difficult enough to justify the increased running time between the normal pricing problem and the augmented pricing problem. That is, the original pricing problem appears to be too easy to justify the extra effort of the cutting planes.

2 Concluding Remarks

Based on the successful work with the VRPTW (with the subset-row inequalities), the less successful work (the Chvátal-Gomory rank1 cuts and clique inequalities), and the above experiments on various packing-like problems, it can be concluded that one needs to be very careful in choosing which cutting planes to include for a given problem. It appears that the pricing problem of the decomposed problem treated should be hard to solve even before cutting planes for the master problem are considered. Most likely, the best results would be achieved if the pricing problem is strongly $\mathcal{NP}$-hard to begin with (as was the case for VRPTW and not for packing-like problems). In this way the effect of the augmentation is less significant (at least complexity wise). Also, for the cutting planes to be effective in the master problem it is preferable to have a large deviation in objective coefficient values and large gaps. The latter is true for any cutting plane algorithm, but since cutting planes for the master problem can be computationally expensive it is truly important to have it in mind if the BCP algorithm is to be successful.

3 Directions for Future Research

There are plenty of known cutting planes to investigate and apply in BCP algorithms. As indicated in Chapter 6 the trick is to formulate the master problem cutting plane in the augmented original formulation. This may need some creative reverse engineering, but based on the work in this thesis it does seem possible. I am suggesting to explore the possibilities within both combinatorial and general mixed integer cuts for master problem formulation.

Based on the success story for VRPTW and my less successful experiments with packing-like problems, it would be interesting to further investigate how hard a problem needs to be (with respect to the decomposed master and subproblems) such that the expensive procedure of using cutting planes derived from the master problem formulation pays off. It may be possible to devise a classification scheme of problems based on some measurement (e.g., running time complexity) to categorize problem formulations. In this perspective, there may be potential in experimenting with decompositions leading to strongly $\mathcal{NP}$-hard pricing problems.

With the link between cutting planes in the master problem and the augmented original formulation described in Chapter 6, it could be interesting to consider a branch-and-cut algorithm for the augmented original formulation. Is it possible to exploit the knowledge from the decomposition or will any possible improvements be lost in the higher dimensional polytope?

Chapter 8

Summary in Danish

Simon Spoorendonk
DIKU Department of Computer Science, University of Copenhagen

1 Resume

Denne afhandling omhandler Dantzig-Wolfe dekomponering af heltalsproblemer med specielt henblik på at tilføje snit i det dekomponerede problem. Dantzig-Wolfe dekomponering benyttes ofte på heltalsproblemer (de *originale* problemer) med strukturelle egenskaber, der gør, at de med fordel kan opdeles i mindre dele (*delproblemer*), hvis løsninger samles i et *master* problem. For at løse master problemet kan man benytte kolonnegenering på den lineære relaksering. Ved kolonnegenering tilføjes løsninger fra delproblemerne (udtrykt som kolonner) undervejs i løsningsprocessen indtil ingen kolonner kan forbedre den lineære løsning af master problemet. På denne måde benyttes kun en brøkdel af de eksponentielt mange mulige kolonner (variable) i master problemet. Løsningen til den lineære relaksering benyttes som grænseværdi i en *branch-and-bound* algoritme til at finde en heltalsløsning.

Et snit er en lovlig ulighed til problemet, som bortskærer en fraktional løsning. Dvs. en løsning til den lineære relaksering bortskæres uden at bortskære en heltalsløsninger. Når s-nit benyttes sammen med kolonnegenering kaldes det en *branch-and-cut-and-price* algoritme. Når et snit tilføjes i master problemet skal den duale omkostning fra snittene medtages i omkostningsberegningerne for de generede kolonner. Hvis et snit kan opskrives som en linear kombination af variable i det originale problem er dette ligetil, da kun omkostningen for de enkelte variable i delproblemerne ændrer sig. Det skal dog nævnes, at selvom kun omkostningen ændrer sig kan delproblemerne godt skifte karakter. Dette bemærkes især, hvis specielt designede algoritmer benyttes til løsning af delproblemerne. Snit udledt fra det originale problem benyttes ofte i branch-and-cut-and-price algoritmer, derimod er det mindre velstuderet at benytte snit defineret direkte på variable i master problemet. En af udfordringerne er, at omkostningsberegningerne for nye kolonner bliver mere komplekse, da det muligvis ikke er muligt at udtrykke den duale omkostning på snittet ved alene at ændre på omkostningen af variablerne i delproblemet. I denne ph.d. afhandling vises det, hvorledes et snit defineret på master problem variable kan beskrives i det originale problem.

Det videnskabelige hovedbidrag præsenteres i fire artikler. Den første artikel omhandler brug af *subset-row* uligheder, som er en delmængde af den generelle familie af Chvátal-Gomory

snit, i ruteplanlægningsproblemet med tidsvinduer (eng.: the vehicle routing problem with time windows). I denne artikel beskrives hvorledes snittene på master problemet kan implementeres i den dynamiske programmeringsalgoritme til delproblemet på en effektiv måde. Resultaterne er meget lovende, og med de nye snit var det muligt at finde optimale løsninger til adskillige hidtil uløste problemer. I den anden artikel bygges videre på dette resultat, og teorien udvides således, at alle Chvátal-Gomory rang-1 snit kan tilføjes til ruteplanlægningsproblemet med tidsvinduer. Resultaterne er ikke nær så imponerende som i den forrige artikel, men det vises, at det er muligt at finde særdeles gode grænseværdier ved tilføjelsen af denne familie af snit.

I den tredje artikel betragtes klike (eng.: clique) uligheder, som er et kombinatorisk snit til master problemet for ruteplanlægningsproblemet med tidsvinduer. I tråd med arbejdet i de forrige artikler, beskrives i denne artikel hvorledes klike ulighederne kan behandles i algoritmen for delproblemet. Eksperimentelle resultater indikerer, at der er en gevinst mht. grænseværdien, men at hastigheden på algoritmen forringes i de fleste tilfælde. I den afsluttende fjerde artikel præsenteres en generel ramme for brug af snit i branch-and-cut-and-price algoritmer. Med rammen beskrives hvorledes snit i det originale og master problemet er forbundet, og specielt vises det, at snit defineret på variable i master problemet kan beskrives ved at betragte et udvidet originalt problem. Artiklen giver en beskrivelse og en indsigt til snit i branch-and-cut-and-price algoritmer, som forhåbentlig kan føre til mere forskning indenfor dette område.

Udover hovedbidraget er der medtaget yderligere tre artikler som er udarbejdet under ph.d.-forløbet. Den første artiklen præsenterer en snitalgoritme for det ressourcebegrænsede korteste vej problem, der ofte ses som delproblem ved dekomponering. Algoritmen er i bestemte tilfælde klart overlegen i forhold til klassiske algoritmer baseret på dynamisk programmering. Den anden artikel præsenterer en kolonnegenereringsalgoritme til et beskyttelsessystem indenfor rutning af telekommunikation i et fibernetværk. Det ønskes at identificere primær- og backup-veje således at alt båndbredde er under den bedste beskyttelse. En kolonnegenereringsmodel, hvor en kolonne definerer parret bestående af en primær- og en backupvej, bliver beskrevet, og udførte tests viser, at dette beskyttelsessystemet er yderst effektivt. Den tredje artikel omhandler indkomststyring i *liner shipping* virksomheder, hvori omfordeling af containere behandles. Da verden er delt i eksport/import zoner (fx. fra Asien til Europa) er omfordelingen af containere et reelt problem, da kapaciteten på skibene ellers er misvisende. Der præsenteres en kolonnegenereringsmodel, hvor en kolonne er en plan for transport af et produkt fra en forsyningshavn til en efterspørgselshavn. Ved brug af denne algoritme er det muligt at løse problemer af et langt større omfang end hidtil.

Part IV

Other Contributions

Chapter 9

A Branch-and-Cut Algorithm for the Elementary Shortest Path Problem with Resource Constraints

Mads Jepsen
DIKU Department of Computer Science, University of Copenhagen

Bjørn Petersen
DIKU Department of Computer Science, University of Copenhagen

Simon Spoorendonk
DIKU Department of Computer Science, University of Copenhagen

Abstract

This paper introduces a branch-and-cut (BAC) algorithm for the elementary shortest path problem with resource constraints (ESPPRC), which commonly appears as a subproblem in column generation based algorithms, e.g., in the classical Dantzig-Wolfe decomposition of the capacitated vehicle routing problem. Specifically, we consider an undirected graph with arbitrary edge costs (i.e., negative cost cycles may appear) and with resources that are equally constrained at all nodes and arcs. A mathematical model and valid inequalities are presented, including a new family of valid inequalities denoted the generalized capacity inequalities. Experimental tests are performed on a set of generated instances with graphs of high edge density and a set of instances from the literature. Traditionally, labeling algorithms have been the dominant solution method for the ESPPRC, but experimental results show that the BAC algorithm is superior on all the tested instances.

Keywords: Branch-and-cut algorithm, elementary shortest path problem with resource constraints

In revision. Preliminary version published as a technical report, 2008

1 Introduction

The elementary shortest path problem with resource constraints (ESPPRC) can informally be stated as the problem of finding a shortest path between two nodes in a graph where resources are accumulated along the path, and where the amount of resources are constrained.

In this paper, we consider the case where the graph is undirected and edge costs are allowed to take on any value. Furthermore, we demand that the path is simple such that no nodes are visited more than once. The resources considered in this paper are all bounded such that the lower and upper bound of the amount of a resource that are accumulated along the path is equal for all nodes and edges. We assume, that the resource lower bounds are zero and that the accumulations are monotone increasing and only performed at the nodes. This type of *globally constrained* resource compares to the vehicle capacity known from the capacitated vehicle routing problem, where the resource accumulates a positive value (demand) at each node and the upper bound (capacity of the vehicle) may not be exceeded.

It is now possible to give a more formal statement of the ESPPRC. Let $G = (V, E)$ be an undirected graph with nodes V and edges E. Let a cost c_e be associated with each edge $e \in E$, let d_i^r be a positive resource accumulation associated to each node $i \in V$ for each resource $r \in R$, and let Q^r be the upper bound on the resource r. Then given a source node $s \in V$ and a target node $t \in V$; find a path between s and t with minimum cost satisfying that the sum of the resource r from at each of the visited nodes is not more than Q^r for all $r \in R$.

The ESPPRC defined as above is $\mathcal{NP}$-hard in the strong sense. This is easily verified by reduction from the longest path problem. The definition of the ESPPRC varies in the literature, especially with regard to edge costs, resource bounds, and resource accumulations.

Beasley and Christofides [7] presented a mathematical model (very similar to the one used in this paper) and performed experimental tests using a branch-and-bound algorithm based on Lagrangian dual bounds. Dumitrescu and Boland [15] presented a labeling algorithm that was improved by preprocessing based on resource availability. Carlyle et al. [10] proposed a Lagrangian relaxation algorithm where paths with cost between the Lagrangian bound and the current upper bound are found using the k-shortest path algorithm by Carlyle and Wood [9]. Common for these approaches are that they all assume that the graph have no negative cost cycles. This makes it easier to ensure simplicity of the path, since it cannot pay off to visit a node more than once. The ESPPRC in this form is weakly $\mathcal{NP}$-hard, and results of the algorithms presented above are therefore not directly comparable to the results in this paper.

Another common definition is to consider resource bounds individually for each node (or edge). In this case, it is often necessary to consider an undirected graph, because the direction of the path determines the correct resource accumulation at a given node. Such resources compare to the time in the vehicle routing problem with time windows, where the resource (time) accumulates for each edge and the nodes must be visited within a resource window (a time window defined by a minimum and a maximum arrival time for a node). Such resources are said to be *locally constrained.* Dror [14] proved that the ESPPRC with a single globally constrained resource and a single locally constrained resource is $\mathcal{NP}$-hard in the strong sense. Feillet et al. [16] presented a labeling algorithm where the simplicity of the path is ensured with the use of an additional globally constrained resource per node. Chabrier [11] improved on the labeling algorithm by applying various bounding procedures to avoid extending unwanted paths. Righini and Salani [24] proposed a bi-directional labeling algorithm where paths are extended from both the source node and the target node until a given *middle* of a monotone increasing resource is reached, e.g., when half the time was consumed on the path. The partial paths are then combined to construct a full path. Independently, Boland et al. [8] and Righini and Salani [25] proposed to extend the labeling algorithm by relaxing the node resources and adding them incrementally until the path is simple. In the former paper, this is referred to as a *state space augmentation* algorithm, and in the latter, it is denoted a *decremental state space relaxation* algorithm. Furthermore, Righini and Salani [25] propose to use the result of the relaxed problem in a branch-and-bound algorithm.

The algorithms presented above are mainly labeling algorithms. As mentioned in Beasley and Christofides [7], even the algorithms based on Lagrangian relaxation make use of a dynamic programming algorithm if negative costs cycles are allowed. The strength of the labeling algorithms is, that the locally constrained resources are easily implemented, since the paths are build piece by piece such that resource limits can be checked at every step. In fact, non-linear functions for accumulation of resources can be handled easily, see e.g., Desaulniers et al. [12]. Generally, labeling algorithms are assumed to perform well on a sparse graphs with tightly constrained resources, since this yields a very reduced solution space to search, i.e., few states in the dynamic programming table needs to be searched. However, when the graph is dense and the resources are loosely constrained, the labeling algorithms get closer to a full enumeration of all paths.

Modeling of resources (accumulation and bounds) is limited in branch-and-cut (BAC) algorithms that are based on linear programming (which is the case in this paper). Globally constrained resources with positive accumulation can be modeled as single knapsack constraints (and remain simple to model with negative accumulation). Locally constrained resources with positive accumulation can be modeled for a directed graph with the use of the Miller-Tucker-Zemlin (MTZ) constraints, see Miller et al. [22]. This gives rise to $|E|$ additional constraints and $|V|$ variables per resource. Another modeling approach gives rise to $|V|$ constraints and $|E|$ variables per resource, see e.g., Ascheuer et al. [1, 2]. A different approach is to relax the resource constraints and, in a cutting plane fashion, make use of the infeasible path inequalities which cuts of any path (or partial path) that violates a resource bound. In Ascheuer et al. [2] a BAC algorithm for the traveling salesman problem with time windows makes use of the three modeling approaches described above. Results indicate that the infeasible path inequalities are to be preferred.

When considering the ESPPRC as a subproblem in a column generation context, another issue comes up. Recent branch-and-cut-and-price algorithms, see e.g., Jepsen et al. [20], Petersen et al. [23], Desaulniers et al. [13], Spoorendonk and Desaulniers [27], Baldacci et al. [5], make use of cutting planes where the dual values are not directly subtractable from the edge costs, which has previously been the preferred approach, see e.g., Fukasawa et al. [18]. The subtraction of such dual values depend on the complete path and can be very cumbersome to overcome in labeling algorithms. However, when following the ideas in Spoorendonk et al. [28] it is clarified how to model the additional costs in the subproblem, whereupon the BAC algorithm can be applied.

Results by Ascheuer et al. [2] for the traveling salesman problem with time windows indicate, that it is expensive (in running time) in a BAC algorithm, to use either of the modeling approaches for locally constrained resources, i.e., the time windows. However, when only globally constrained resources are considered, it seem likely that a BAC algorithm can be competitive with labeling algorithms. So, although locally constrained resources can be modeled in a BAC algorithm, it is not within the scope of this paper to investigate that approach. The reason for considering an undirected graph in this paper is mainly for simplicity. The BAC algorithm can easily be extended to the directed case by doubling the number of variables in the mathematical formulation. Neither of the separation routines are affected by this (except for the doubling of variables).

The main contribution of this paper is the introduction of a BAC algorithm for solving the ESPPRC. This includes a 2-index mathematical model and a presentation of valid inequalities with emphasis on the introduction of the generalized capacity inequalities. The computational results indicate that the BAC algorithm is competitive with labeling algorithms when considering dense graphs, and even more so when the resources are loosely constrained.

The paper is outlined as follows: Section 2 presents work on BAC algorithms for problems that are related to the ESPPRC and Section 3 contains a formal integer programming model of the ESPPRC. Section 4 describes the cutting planes used in the BAC algorithm and the computational results are found in Section 5. Section 6 holds concluding remarks and suggestions for further research.

2 Related Work

Bauer et al. [6] suggested to solve the ESPPRC by a BAC algorithm, but to our knowledge nothing further has been published in the literature, although several BAC algorithms exist for problems related to the ESPPRC. Bauer et al. [6] consider the knapsack constrained circuit problem (KCCP) where a minimal capacitated cycle in a graph is sought. This is equivalent to the ESPPRC if one node is fixed in the KCCP, since this node can be spilt into a source and a target node in the ESPPRC. A BAC algorithm was implemented to solve the KCCP where the demand of the nodes was given with unit weights. This variant is denoted the cardinality constraint circuit problem. The instances considered by Bauer et al. [6] have positive edge costs, but negative cost cycles would not affect the algorithm.

In the prize collecting traveling salesman problem (PCTSP), see e.g., Balas [3, 4], a prize is collected at each visited node and a minimum amount of accumulated prizes must be collected on the tour. That is, the edge costs are positive but the prizes may yield an overall negative solution value. The difference with this variant of the TSP and the ESPPRC is, that in the PCTSP a minimum amount of prizes need to be collected, which forces some of the intermediate nodes to be visited. This is not the case for the ESPPRC as defined in this paper.

In the orienteering problem, see e.g., Fischetti et al. [17], the profit of visiting the nodes is maximized and the length of the tour is bounded by a maximum length. The only difference compared to the definition of the ESPPRC of this paper is, that the resource accumulation is on the edges instead of in the nodes. The instances considered by Fischetti et al. [17] have positive edge costs, but again negative cost cycles would not affect the algorithm.

3 Mathematical Models

This section presents a flow model for the ESPPRC in the undirected graph G. Recall the resource demand d_i^r for nodes $i \in V$, and the resource upper bound Q^r for resource $r \in R$. Let the binary variable x_e indicate the flow on edge $e \in E$. When describing the model some shorthand notation will be used. For a set of nodes $S \subseteq V$ let the set of edges $\delta(S) = \{(i,j) : i \in S \wedge j \in V \setminus S\}$ denote the edges between S and $V \setminus S$ where $\delta(i)$ is shorthand for $\delta(\{i\})$ when the node set S consists of a single node $i \in V$. Let $E(S) = \{(i,j) : i \in S \wedge j \in S\}$ be the set of edges between the nodes in S. Let the short-hand notation

$$x(T) = \sum_{e \in T} x_e$$

indicate the flow in the edge set T. Let the shorthand notation $y_i = \sum_{e \in \delta(i)} x_e/2$ indicate the flow in node $i \in V \setminus \{s,t\}$, and for a set of nodes $S \subseteq V$ let

$$y(S) = \sum_{i \in S} y_i$$

be the flow in that node set. The mathematical model of the ESPPRC is then:

$$\begin{aligned}
\min \; & \sum_{e \in E} c_e x_e && && (1)\\
\text{s.t.} \; & x(\delta(s)) = 1 && && (2)\\
& x(\delta(t)) = 1 && && (3)\\
& x(\delta(i)) = 2y_i && i \in V \setminus \{s,t\} && (4)\\
& \sum_{i \in V} d_i^r y_i \leq Q^r && r \in R && (5)\\
& x(E(S)) \leq y(S) - y_i && i \in S, S \subset V, |S| \geq 2 && (6)\\
& x_e \in \{0,1\} && e \in E && (7)
\end{aligned}$$

The objective function (1) minimizes the overall edge cost. Constraints (2) and (3) ensure that the source node and the target node are end points of the path. Constraints (4) are the flow conservation constraints. Constraints (5) impose the resource constraints. Constraints (6) impose connectivity and subtour elimination. Finally, constraints (7) define the domain of the variables. Note, that $y_i \in \{0,1\}$ due to (2), (3), (6), and (7).

This model has $|E|+|V-2|$ variables and an exponential number of constraints due to (6). In a BAC algorithm, these constraints will be relaxed and separated when violated to ensure feasibility. That is, when disregarding constraints (6) the model have $|V| + |R|$ constraints.

4 Cutting Planes

This section presents the inequalities used in the BAC algorithm: The generalized subtour elimination constraints (constraints (6) the mathematical model), the 0-1 knapsack cover inequalities, and the generalized capacity inequalities for the ESPPRC.

4.1 Generalized Subtour Elimination Constraints

These constraints are generalizations of the subtour elimination constraints known from the traveling salesman problem, which are also valid for ESPPRC on the form:

$$x(E(S)) \leq |S| - 1 \qquad \forall S \subset V \qquad (8)$$

Restricting the constants on the right-hand side to reflect the actual node flow provides a tighter inequality, since $y_i \leq 1$ for all $i \in V \setminus \{s,t\}$. The generalized subtour elimination constraints can be written on either of the forms:

$$x(E(S)) \leq y(S) - y_i \qquad \forall i \in S, \forall S \subset V \tag{9}$$

$$x(\delta(S)) \geq 2y_i \qquad \forall i \in S, \forall S \subset V \setminus \{s,t\} \tag{10}$$

Separation of (9) and (10) can be done by solving a minimum cut problem from each node $i \in V \setminus \{s,t\}$ to the target node t (or the source node s) on the induced graph of the LP solution $(x^\star, y^\star)$ with edge weights w_e given as:

$$w_e = \begin{cases} x_e^\star & e \in E \setminus \{(s,t)\} \\ M & e = (s,t) \end{cases}$$

where M is a sufficiently large constant to ensure that s and t are on the same side of the cut, see Wolsey [30].

4.2 0-1 Knapsack Cover Inequalities

A 0-1 knapsack cover inequality for a set of nodes $S \subseteq V$ where $\sum_{i \in S} d_i^r > Q^r$ for some $r \in R$ is given as:

$$y(S) \leq |S| - 1 \tag{11}$$

The inequality states, that if a set of nodes violates the upper bound on the resource r, then not all nodes in the set can be visited by the path. The 0-1 knapsack cover inequality (11) can be rewritten as

$$\sum_{i \in S} (1 - y_i) \geq 1 \tag{12}$$

Given the LP solution (x^*, y^*), the separation problem becomes finding a cover S, i.e, a set $S \subseteq V$ satisfying $\sum_{i \in S} d_i^r > Q^r$ for some $r \in R$ such that

$$\sum_{i \in S} (1 - y_i^*) < 1 \tag{13}$$

in which case the corresponding 0-1 knapsack cover inequality (11) is violated. The most violating (11) is identified by minimizing the left-hand side of (13) for all $r \in R$, i.e., by solving:

$$\zeta = \min_{r \in R} \left\{ \min_{S \subseteq V} \left\{ \sum_{i \in S} (1 - y_i^*) z_i : \sum_{i \in S} d_i^r z_i > Q^r, z \in \{0,1\}^{|V|} \right\} \right\}$$

If $\zeta \geq 1$, no cover that violates (11) exists. The separation problem consists of $|R|$ minimization versions of the well known 0-1 knapsack problem, see Kellerer et al. [21], Wolsey [30].

Jepsen and Spoorendonk [19] suggested to exploit the fact that, since $y_i \leq 1$ for all $i \in V \setminus \{s,t,\}$, the flow through a set of nodes S can be less than 2 in an LP solution. That is, scaling the right-hand side of (11) with half the flow $x(\delta(S))$ yields

$$y(S) \leq \frac{1}{2}(|S| - 1)x(\delta(S)) \tag{14}$$

When $x(\delta(S)) < 2$, there are cases where the inequality (14) is violated and the normal 0-1 knapsack cover inequality (11) is not. Jepsen and Spoorendonk [19] suggested an enumeration scheme to separate the inequalities. Their results indicated, that (14) did improve the lower bound in the root node, but had a negative effect on the convergence of the BAC algorithm. Therefore, this family of inequalities are not pursued further in this paper.

4.3 Generalized Capacity Inequalities

This subsection introduces a family of inequalities inspired by the fractional capacity inequalities of the capacitated vehicle routing problem (CVRP), see Toth and Vigo [29]. The generalized capacity inequalities are given as:

$$\frac{1}{2}Q^r x(\delta(S)) \geq \sum_{i \in S} d_i^r y_i \qquad S \subseteq V \setminus \{s,t\}, r \in R \tag{15}$$

The inequalities ensure that a set S of nodes are visited according to their demand, e.g., if 2/3 of the resource is consumed in S, then the flow in and out of S should be at least 4/3. An example of a violated (15) can be seen in Figure 4.3.

The validity of (15) is proved in the following proposition:

Proposition 1. The generalized capacity inequalities (15) are valid for the ESPPRC.

Proof. If $y(S) = 0$ then $x(\delta(S)) = 0$, therefore both the left-hand side and the right hand side evaluate to 0. If $y(S) \geq 1$ then $x(\delta(S)) \geq 2$ and due to the resource constraint (5) for resource r, the right-hand side can never evaluate to more than Q^r which will be the minimal value of the left-hand side, i.e., in this case the resource constraint (5) for resource r dominates the generalized capacity inequality. □

Given an LP solution $(x^\star, y^\star)$ the separation problem of (15) is the problem of finding a set $S \subseteq V \setminus \{s,t\}$ for a resource $r \in R$ such that

$$\begin{aligned}
& \frac{1}{2}Q^r x^\star(\delta(S)) < \sum_{i \in S} d_i^r y_i^\star \\
\Leftrightarrow\ & \frac{1}{2}Q^r x^\star(\delta(S)) - \sum_{i \in S} d_i^r y_i^\star + \sum_{i \in V} d_i^r < \sum_{i \in V} d_i^r \\
\Leftrightarrow\ & \frac{1}{2}Q^r x^\star(\delta(S)) + \sum_{i \in S} d_i^r (1 - y_i^\star) + \sum_{i \in V \setminus S} d_i^r < \sum_{i \in V} d_i^r
\end{aligned}$$

Separating (15) for an be done by solving $|R|(|V| - 2)$ different minimum cut problems one from each node $h \in V \setminus \{s,t\}$ to the target node t for each resource $r \in R$. The problems are solved as maxflow problems using the same procedure as for separating (9) and (10). The maxflow problem for each h is solved on a directed graph induced from the LP solution $(x^\star, y^\star)$, i.e., edges are split into opposite directed arcs, and the arcs into h are disregarded. The edge weights e_{ij} are given as:

$$w_{ij} = \begin{cases} \frac{1}{2}Q^r x_{hj}^\star + d_j^r & i = h,\ j \in V \setminus \{h,t\} \\ \frac{1}{2}Q^r x_{it}^\star + d_i^r(1 - y_i^\star) & i \in V \setminus \{s,t\},\ j = t \\ \frac{1}{2}Q^r x_{ij}^\star & i \in V \setminus \{h,t\},\ j \in V \setminus \{h,t\} \\ M & i = s, j = t \end{cases}$$

Consider the fractional solution given by the graph to the right with different fractional edge values indicated by the dotted and dashed lines. The nodes are numbered $0, \ldots, 5$ where a path is sought from node 0 to 0. For a single resource, the resource demands are given as $d = \{0, 2, 2, 2, 2, 1\}$ and the resource upper bound Q is 5.

Consider a generalized capacity inequality (15) covering the node set $S = \{1, 2, 3\}$ resulting in a fractional flow $x^\star(\delta(S)) = x^\star_{01} + x^\star_{03} = \frac{4}{3}$ through the node set. The corresponding (15) is violated since

$$\frac{1}{2} Q x^\star(\delta(S)) = \frac{10}{3} \ngeq \sum_{i \in S} d_i y^\star_i = \frac{12}{3}$$

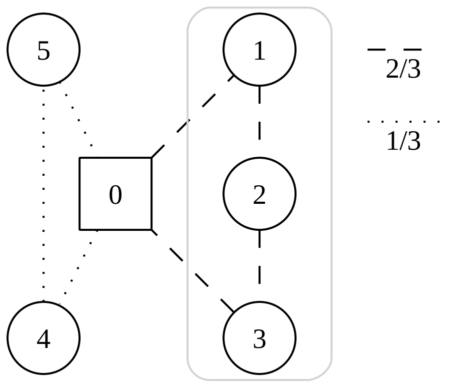


Figure 1: A violated generalized capacity inequality (15).

where M is a sufficiently large constant to ensure that s and t are on the same side of the cut. The induced graph is denser than the induced graph used for separating (9) and (10), therefore the separation of (15) is expected to be slower.

5 Computational Results

The experiments begin with an investigation of the impact of the parameter settings for the cut generation of the generalized subtour elimination constraints (9). Next, the impact of the generalized capacity inequalities (15) are investigated. For the parameter test, we consider 10 of the harder problems of the generated instances. This is followed by a lower bound comparison on the generated instances using different separation strategies. Last is a comparison of the BAC algorithm and a labeling algorithm. We use a labeling algorithm, that is implemented as described in Righini and Salani [24]. For the known instances, the comparison is made with the results obtained in Righini and Salani [25]. The mathematical model for the ESPPRC presented in this paper contains an exponential number of constraints, so it is not possible to input it directly into a general purpose mixed integer solver such as ILOG's CPLEX. However, it is possible to model the globally constrained resources in a similar way as the locally constrained resources, e.g., with the MTZ constraints. Such a model can be plugged into CPLEX and solved directly, but preliminary results indicate that this approach is always significantly slower than using the BAC algorithm proposed in this paper.

All experiments are performed on a 2.66 GHz Intel(R) Xeon(R) X5355 machine with 8 GB memory using CPLEX 10.2. The BAC algorithm is implemented using callback functions for the cut generation, which is available in the CPLEX callable library. The tests are performed using the default CPLEX parameters. This includes the generation of cuts for general mixed-integer programs such as Chvátal-Gomory, mixed-integer rounding, and disjunctive cuts.

Also, the 0-1 knapsack covers are included in the CPLEX default settings and preliminary tests indicated, that the separation time nor the change in lower bounds were much affected by the cuts. Therefore, we have not performed any further tests of the 0-1 knapsack covers but rely on the CPLEX default settings.

5.1 The Benchmark Instances

A set of benchmarks derived from the CVRP instances (divided in series A, B, E, G, M, and P) available at `http://www.branchandcut.org` has been generated. Here, the source and target nodes are chosen by splitting the node representing the depot in two. To identify sufficiently hard instances of the ESPPRC, we have used the BAC algorithm for the ESPPRC in a simple column generation algorithm for the CVRP, see e.g., Baldacci et al. [5] for the details on mathematical models. We have not included results for the CVRP, since it is not in the scope of this paper. Note, that for all the generated instances there is a valid upper bound of 0, since they are constructed from a column generation algorithm. The instances are named from the derived CVRP instances, which are given as letter indicating the series followed by the number of nodes and vehicles (the latter is not used for the ESPPRC). At the end a number, indicating the final iteration number of our column generation algorithm, is added, e.g., the instance P-n50-k7-92 is from the P-series and consists of 50 nodes (where 7 vehicles are used for the CVRP), and is from iteration 92. The ESPPRC instances are gathered in the SPPRCLIB available at `http://www.diku.dk/~spooren/spprclib.htm`.

Beside the generated instances, we consider the instances used in Feillet et al. [16], Righini and Salani [24, 25] with 100 nodes and a single globally constrained resource (the capacity resource). These instances are derived from the benchmarks by Solomon [26] for the vehicle routing problem with time windows, where the time constraints have been discarded. For the c101, r101, and rc101, three different distributions of nodes are chosen, and ten instances have been created for each distribution, where the resource bounds (capacity) range from 10 to 100 in steps of 10. We consider only instances with bounds of 60 and above. Additionally, we have extended the set of instances by setting bounds to 200, 500, 700, and 1000. A larger resource bound results in loosely constrained instances, that are expected to be harder to solve to optimality. The instances are named according to the series and a tenth of the capacity, e.g., c_100_09 is from the c101 instance, with capacity 90.

5.2 Impact of the Parameters for the Generalized Subtour Elimination Constraints

The setting of the parameters for the generation of violated generalized subtour elimination constraints (6) can have a huge influence on the computation time of the BAC algorithm. A low threshold on violation will result in good lower bounds and fewer branch nodes, but a slower convergence in each node, while the opposite is true for a high threshold. Also, the number of violated cuts added in each iteration can influence the convergence and the time spent when reoptimizing the LP-problem.

Figure 2 shows a plot with two axes given as the violation threshold and number of cuts to add per iteration. The requirement of violation is ranging from 0.1 to 1 in steps of 0.1, and the number of cuts to add is starting at 1 and then from 10 to 100 in steps of 10. The vertical axis indicates the average time spent. The time for each instance is scaled to the interval $]0, 1]$ where 1 is the maximum time given for all the parameter settings for that instance.

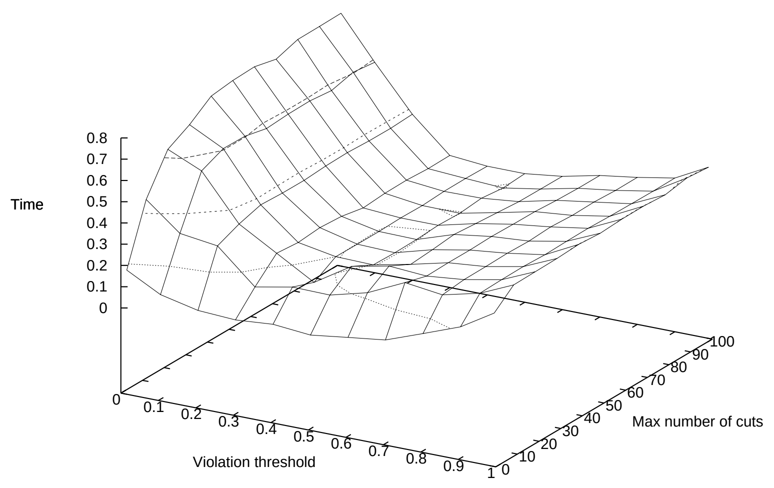


Figure 2: Parameter test for the generalized subtour elimination constraints (9). Above is a plot of the average time given the violation threshold and the number of cuts to add.

From Figure 2, it is observed that the best parameter setting appears to be to add 1 cut per iteration with a violation of at least 0.4. This indicates that the cut separation time is insignificant compared to solving the LPs.

5.3 Investigating the Generalized Capacity Inequalities

Note, that the generalized capacity inequalities (15) can substitute the generalized subtour elimination constraints (9) in the model (1)-(7), since any infeasible integer solution will be violated by some generalized capacity inequality. However, due to the computational expensive separation routine for constraints (15), a cut policy was chosen such that constraints (15) are only separated (and possible added) whenever no violated constraints (9) are separated (using the default parameters found above). Preliminary tests indicated, that due to a computational expensive separation routine for constraints (15), the cuts were not worth the effort. A slow separation was expected since the max-flow calculations are done on very dense graphs compared to the very sparse graph used in the separation of constraints (9). However, we believe that constraints (15) may become useful, e.g., with the use of a faster heuristic separation routine.

Figure 3 shows, as before, a plot of the violation threshold, number of cuts to add per iteration, and average time. The time is calculated without the separation time of constraints (15), and therefore only indicates if the convergence of the BAC is improved or not, when constraints (15) are added. Figure 3 indicates that a large violation threshold (≥ 0.8) is preferred for constraints (15) and that, the convergence of the BAC algorithm is faster when few of the constraints (15) are added. Figure 4 substantiate this result, as it can be seen that almost no cuts are added with violation thresholds 0.8 and higher. Although the generalized capacity inequalities (15) are a theoretically interesting set of inequalities, our tests have shown that in their current form and with the proposed exact separation routine, the inequalities do not appear to be computationally competitive.

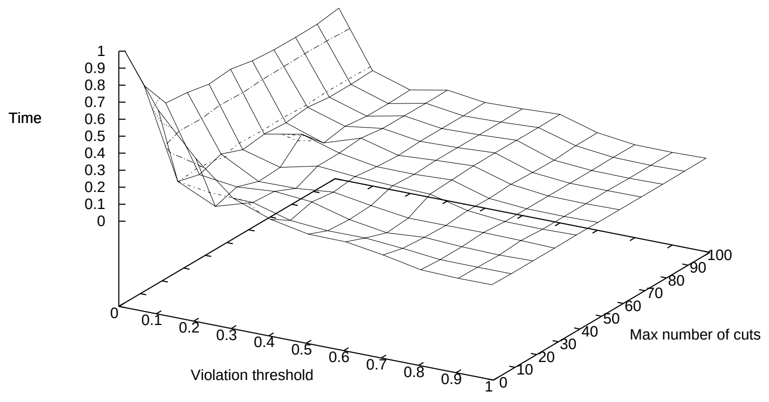


Figure 3: Parameter test for the generalized capacity inequalities (15). Above is a plot of the average time given the violation threshold and the number of cuts to add.

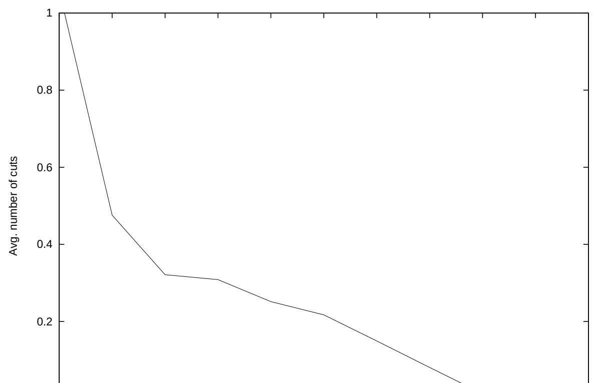


Figure 4: Parameter test for the generalized capacity inequalities (15). Above is given the average scaled number of generalized capacity inequalities added with different violation thresholds when solving the instances, i.e., with a violation threshold of 0.1 the number of cuts are decreased by about 50 % compared to the setting with a violation threshold of 0.01.

5.4 Lower Bound Comparison

Table 1 sums up the root lower bounds (root) and the number of branch nodes (nodes) for three different cut separation parameter settings. A '-' entry in the branch node columns indicates that the BAC algorithm timed out at 600 seconds. The three parameter settings tested are:

- `GSEC` is the BAC algorithm where at most 1 violated generalized subtour elimination constraint (9) with a minimum violation of 0.01 is added per iteration.
- `GCI` is the BAC algorithm with the `GSEC` parameter setting and when no violated (9) are found then at most 1 violated generalized capacity inequality (15) with a minimum violation of 0.01 is added.
- `default` is the BAC algorithm where at most 1 violated generalized subtour elimination constraint (9) with a minimum violation of 0.4 is added per iteration.

The optimal solution is given in the rightmost column.

When comparing the parameter settings `GSEC` and `GCI`, it is obvious that the generalized capacity inequalities (15) improve the lower bounds considerably. The average gap is decreased by 63% when comparing the two settings, this includes the instances that timed out and potentially could have improved the lower bound further. Surprisingly, the number of branch nodes does not decrease proportionally with the size of the gap. That is, for the instances that did not time out, the average gap is closed by 76% but with only 7% fewer branch nodes. In several cases, the number of branch nodes actually increases considerably (A-n63-k9-157, B-n45-k6-54, P-n50-k10-24, P-n55-k10-44). This indicates that (15) complicates the branch decisions. The comparison of the settings `GSEC` and `default` is more as expected: A worse lower bound with the `default` setting leads to more branch nodes. However, the previous test for the generalized subtour elimination (9) constraints showed, that this setting was the fastest on average.

5.5 Comparison with a Labeling Algorithm

Table 2 shows the running time of the BAC algorithm (BAC time (s)) with default parameters compared to the running time of our implementation of a labeling algorithm (LA) (LA time (s)) for the generated instances. The time limit was set to two hours and a timeout is indicated with a '-' in the table. The rightmost column presents the speed up if both algorithms finished. The BAC algorithm clearly outperforms the labeling algorithm. That is, in all 45 instances. However, it is worth noting that when the solution is near 0 (which is and upper bound for all instances since they are generated as pricing problems in a column generation algorithm) then the labeling algorithm performs much better than on the instances that contains much negativity. That is, the label algorithm is faster when there are less negativity in the problem whereas the BAC algorithm appears to be more robust. It should be noted that the implementation of our labeling algorithm may be improved, but it is doubtful, that it will be competitive with the BAC algorithm for the instances with a speed up of more than 100.

	GSEC		GCI		default		
Name	nodes	root	nodes	root	nodes	root	solution
A-n54-k7-149	231	-90877	-	-41213	280	-109018	-12492
A-n60-k9-57	1641	-98206	-	-64557	3071	-118437	-1000
A-n61-k9-80	205	-63534	92	-41032	462	-73397	-23549
A-n62-k8-99	133	-103839	-	-47340	301	-122973	-35969
A-n63-k9-157	122	-63082	492	-38929	113	-78190	-24189
A-n63-k10-44	127	-76475	149	-51035	280	-80765	-32561
A-n64-k9-45	358	-92812	157	-65209	425	-104686	-50550
A-n65-k9-10	152	-93117	129	-58526	189	-103936	-42835
A-n69-k9-42	72	-56453	-	-53299	179	-60410	-43290
A-n80-k10-14	84	-121510	45	-112483	120	-128508	-105283
B-n45-k6-54	277	-95588	497	-88761	502	-103214	-74278
B-n50-k8-40	166	-105497	-	-41212	237	-128488	-12832
B-n52-k7-15	25	-85997	22	-79129	59	-90278	-74998
B-n57-k7-20	12	-876421	19	-876421	328	-882924	-867154
B-n66-k9-50	239	-81006	28	-38097	1195	-94120	-26520
B-n67-k10-26	184	-55180	178	-26808	343	-63086	-21924
B-n68-k9-65	150	-88375	-	-55175	342	-99383	-31001
B-n78-k10-70	344	-91021	-	-54330	480	-101516	-44333
E-n76-k7-44	117	-30127	115	-25885	338	-32038	-22214
E-n76-k10-72	239	-36569	164	-31404	138	-38613	-25241
E-n76-k14-102	3163	-28126	-	-16153	3992	-31018	-1
E-n76-k15-40	3747	-25752	-	-17526	4993	-28675	-1
E-n101-k8-291	48	-8296	-	-7398	197	-9472	-4266
E-n101-k14-158	1468	-25748	-	-22729	1350	-30882	-3590
G-n262-k25-316	669	-1434843	-	-1434843	1510	-1434883	-1426535
M-n101-k10-97	40	-35323	37	-34758	76	-37825	-32628
M-n121-k7-260	89	-162680	-	-161424	147	-164742	-160097
M-n151-k12-15	338	-87899	-	-85488	822	-92880	-79996
M-n200-k16-143	6	-199411	4	-199411	118	-201772	-198792
M-n200-k17-12	4	-121506	1	-121210	7	-121506	-121210
P-n50-k7-92	950	-18594	1152	-12245	1319	-21516	-2
P-n50-k8-19	160	-89868	40	-89848	207	-90606	-83307
P-n50-k10-24	197	-19811	608	-11971	443	-21975	-2965
P-n51-k10-30	1028	-23812	-	-18488	1588	-27061	-2
P-n55-k7-116	84	-27065	36	-22945	105	-28094	-17824
P-n55-k8-260	101	-18839	145	-11377	167	-22237	-3573
P-n55-k10-44	913	-21448	2197	-11798	1192	-25131	-1090
P-n55-k15-88	5971	-26723	-	-20135	4781	-28128	-2
P-n60-k10-24	242	-26948	137	-21183	301	-29289	-15001
P-n60-k15-8	1495	-21889	1889	-13812	1507	-24674	-534
P-n65-k10-102	2390	-18923	-	-10975	2532	-21424	-3
P-n70-k10-12	2	-72264	1	-70317	21	-73460	-70317
P-n76-k4-41	1	-88276	1	-88276	1	-88276	-88276
P-n76-k5-16	6	-108884	10	-108884	24	-108884	-107633
P-n101-k4-174	174	-19656	165	-19041	395	-19887	-17702

Table 1: Comparison of the number of branch nodes and lower bounds for the generated instances using three different cut separation strategies.

Name	BAC time (s)	LA time (s)	speed up
A-n54-k7-149	6.96	1735.23	249.3
A-n60-k9-57	36.55	242.64	6.6
A-n61-k9-80	4.44	-	∞
A-n62-k8-99	17.94	-	∞
A-n63-k9-157	3.16	-	∞
A-n63-k10-44	2.12	693.80	327.3
A-n64-k9-45	14.57	-	∞
A-n65-k9-10	4.43	-	∞
A-n69-k9-42	1.76	3246.72	1844.7
A-n80-k10-14	12.14	-	∞
B-n45-k6-54	1.32	-	∞
B-n50-k8-40	11.01	-	∞
B-n52-k7-15	1.00	-	∞
B-n57-k7-20	1.74	-	∞
B-n66-k9-50	66.93	-	∞
B-n67-k10-26	4.62	-	∞
B-n68-k9-65	11.88	-	∞
B-n78-k10-70	24.30	-	∞
E-n76-k7-44	6.02	-	∞
E-n76-k10-72	1.19	-	∞
E-n76-k14-102	14.77	45.19	3.1
E-n76-k15-40	19.59	151.59	7.7
E-n101-k8-291	8.08	-	∞
E-n101-k14-158	37.84	-	∞
G-n262-k25-316	53.00	-	∞
M-n101-k10-97	3.12	-	∞
M-n121-k7-260	34.46	-	∞
M-n151-k12-15	78.03	-	∞
M-n200-k16-143	3.18	-	∞
M-n200-k17-12	17.75	-	∞
P-n50-k7-92	2.42	104.22	43.1
P-n50-k8-19	0.36	-	∞
P-n50-k10-24	0.72	2.91	4.0
P-n51-k10-30	2.18	4.06	1.9
P-n55-k7-116	0.58	2275.07	3922.5
P-n55-k8-260	1.20	133.45	111.2
P-n55-k10-44	2.14	14.69	6.9
P-n55-k15-88	3.97	44.73	11.3
P-n60-k10-24	1.04	110.20	106.0
P-n60-k15-8	1.95	2.50	1.3
P-n65-k10-102	6.65	163.48	24.6
P-n70-k10-12	0.24	-	∞
P-n76-k4-41	1.85	-	∞
P-n76-k5-16	0.57	-	∞
P-n101-k4-174	11.25	-	∞
Best	45	0	

Table 2: Time comparison of the BAC algorithm and the labeling algorithm.

Name	BAC time (s)	DSSR time (s)
c_100_06	0.36	0.21
c_100_07	0.38	0.18
c_100_08	0.53	1.34
c_100_09	0.62	2.02
c_100_10	1.14	7.68
c_100_20	0.82	n.a.
c_100_50	3.07	n.a.
c_100_70	2.70	n.a.
c_100_100	4.43	n.a.
r_100_06	0.75	34.64
r_100_07	0.85	143.63
r_100_08	1.35	281.62
r_100_09	1.04	1002.30
r_100_10	0.80	-
r_100_20	2.09	n.a.
r_100_50	26.96	n.a.
r_100_70	16.25	n.a.
r_100_100	1.76	n.a.
rc_100_06	0.23	0.35
rc_100_07	0.66	0.92
rc_100_08	0.90	1.77
rc_100_09	0.36	1.40
rc_100_10	0.77	7.33
rc_100_20	1.08	n.a.
rc_100_50	4.10	n.a.
rc_100_70	4.17	n.a.
rc_100_100	6.47	n.a.
Best	28 (13)	2

Table 3: Time comparison of the BAC algorithm and the labeling algorithm (Righini and Salani [25]).

In Table 3 the BAC algorithm is compared to the results obtained with the decremental state-space relaxation (DSSR) algorithm by Righini and Salani [25] (recall from Section 1 that this a specialized labeling algorithm). The running times for the two algorithms are given in the columns (BAC time (s)) and (DSSR time (s)). Since Righini and Salani [25] performed their tests on a 1.6 GHz Intel (R) Pentium 4(R) with 512 MB memory, and an exact time comparison with our machine is hard, so we have not included the speed up factor. '-' indicates that the algorithm timed out after one hour, the 'n.a.' entry indicates that no result is available for that instance.

Although the DSSR algorithm is faster on two instances out of the 15 comparable cases, it is only marginally better (even when taken their slower machine into account). There is a clear tendency, that when the capacity increases (i.e., when the ESPPRC becomes more loosely constrained) the running times of the DSSR algorithm increase significantly. The running times are also generally increasing for the BAC algorithm when the capacity increases (except for r_100_100), but not as drastically as for the DSSR algorithm. Results are not available for the DSSR algorithm for the extended instances (with capacity from 200 and above), but if the tendency from the smaller instances continues, then the DSSR algorithm will probably not be able to solve the larger instances within the time limit. The BAC algorithm is clearly superior for the loosely constrained instances.

6 Concluding Remarks

This paper introduces a BAC algorithm for solving the ESPPRC. The algorithm clearly outperformed the labeling algorithms (our own implementation of the one describes in Righini and Salani [24] as well as the one by Righini and Salani [25]) for the tested instances. Labeling algorithms have been the preferred solution approach up until now, but the experimental results presented in this paper suggest otherwise. Furthermore, the generalized capacity inequalities were introduced as a set of valid inequalities for the ESPPRC. It can be concluded that the inequalities improve the lower bounds significantly. However, this comes at a cost of complicating the branch decision, and leads to a large amount of additional branch nodes. Also, the exact separation routine takes a considerable amount of time. This is due to solving a maxflow problem on an almost complete graph. That is, the generalized capacity inequalities improve the lower bound, but lead to increased running times.

Future research could include the adaption of more valid inequalities known from related problems, e.g., two-matching inequalities, comb inequalities, and infeasible path inequalities. Another interesting direction is the conditional cuts by Fischetti et al. [17]. Such cuts resemble a specialized branch rule, as they cut off some of the branch tree after solving a subproblem that finds the optimal solution for the subtree. Another natural extension of the work presented in this paper is to extend the BAC algorithm to include locally constrained resources. This would lead to a larger mathematical formulation and will most definitely pose a serious challenge for future research.

References

[1] N. Ascheuer, M. Fischetti, and M. Grötschel. A polyhedral study of the asymmetric travelling salesman problem with time windows. *Networks*, 36(2):69–79, 2000. doi: 10.1002/1097-0037(200009)36:2⟨69::AID-NET1⟩3.0.CO;2-Q.

[2] N. Ascheuer, M. Fischetti, and M. Grötschel. Solving the asymmetric travelling salesman problem with time windows by branch-and-cut. *Mathematical Programming*, 90(3):475–506, 2001. doi: 10.1007/PL00011432.

[3] E. Balas. The prize collecting traveling salesman problem. *Networks*, 19(6):621–636, 1989. doi: 10.1002/net.3230190602.

[4] E. Balas. The prize collecting traveling salesman problem: Ii. polyhedral results. *Networks*, 25(4):199–216, 1995. doi: 10.1002/net.3230250406.

[5] R. Baldacci, N. Christofides, and A. Mingozzi. An exact algorithm for the vehicle routing problem based on the set partitioning formulation with additional cuts. *Mathematical Programming*, 115(2):351–385, 2008. doi: 10.1007/s10107-007-0178-5.

[6] P. Bauer, J. T. Linderoth, and M. W. P. Savelsbergh. A branch and cut approach to the cardinality constrained circuit problem. *Mathematical Programming*, 91(2):307–348, 2002. doi: 10.1007/s101070100209.

[7] J.E. Beasley and N. Christofides. An algorithm for the resource constrained shortest path problem. *Networks*, 19:379–394, 1989. doi: 10.1002/net.3230190402.

[8] N. Boland, J. Dethridge, and I. Dumitrescu. Accelerated label setting algorithms for the elementary resource constrained shortest path problem. *Operation Research Letters*, 34 (1):58–68, 2006. doi: 10.1016/j.orl.2004.11.011.

[9] W. M. Carlyle and R. K. Wood. Near-shortest and k-shortest simple paths. *Networks*, 46(2):98–109, 2005. ISSN 0028-3045. doi: 10.1002/net.v46:2.

[10] W.M. Carlyle, J.O. Royset, and R.K. Wood. Lagrangian relaxation and enumeration for solving constrained shortest-path problems. *Networks*, 51(3):155–170, 2008. doi: 10.1002/net.20212.

[11] A. Chabrier. Vehicle routing problem with elementary shortest path based column generation. *Computers & Operations Research*, 33(10):2972–2990, 2006. doi: 10.1016/j.cor. 2005.02.029.

[12] G. Desaulniers, J. Desrosiers, J. Ioachim, I. M. Solomon, F. Soumis, and D. Villeneuve. A unified framework for deterministic time constrained vehicle routing and crew scheduling problems. In T. G. Crainic and G. Laporte, editors, *Fleet Management and Logistics*, pages 57–93. Kluwer, 1998.

[13] G. Desaulniers, F. Lessard, and A. Hadjar. Tabu search, partial elementarity, and generalized k-path inequalities for the vehicle routing problem with time windows. *Transportation Science*, 42(3):387–404, 2008. doi: 10.1287/trsc.1070.0223.

[14] M. Dror. Note on the complexity of the shortest path models for column generation in VRPTW. *Operations Research*, 42(5):977–979, 1994. doi: 10.1287/opre.42.5.977.

[15] I. Dumitrescu and N. Boland. Improved preprocessing, labeling and scaling algorithms for the weight-constrained shortest path problem. *Networks*, 42(3):135–153, 2003. doi: 10.1002/net.10090.

[16] D. Feillet, P. Dejax, M. Gendreau, and C. Gueguen. An exact algorithm for the elementary shortest path problem with resource constraints: Application to some vehicle routing problems. *Networks*, 44(3):216–229, 2004. doi: 10.1002/net.v44:3.

[17] M. Fischetti, J. J. S. Gonzalez, and P. Toth. Solving the orienteering problem through branch-and-cut. *INFORMS Journal on Computing*, 10(2):133–148, 1998. ISSN 1526-5528. doi: 10.1287/ijoc.10.2.133.

[18] R. Fukasawa, H. Longo, J. Lysgaard, M. Poggi de Aragão, M. Reis, E. Uchoa, and R.F. Werneck. Robust branch-and-cut-and-price for the capacitated vehicle routing problem. *Mathematical Programming*, 106(3):491–511, 2006. doi: 10.1007/s10107-005-0644-x.

[19] M. Jepsen and S. Spoorendonk. A note on the flow extended 0-1 knapsack cover inequalities for the elementary shortest path problem with a capacity constraint. Technical Report 08-02, DIKU Department of Computer Science, University of Copenhagen, Denmark, 2008.

[20] M. 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. doi: 10.1287/opre.1070.0449.

[21] H. Kellerer, U. Pferschy, and D. Pisinger. *Knapsack Problems.* Springer, Berlin, Germany, 2004.

[22] C. E. Miller, A. W. Tucker, and R. A. Zemlin. Integer programming formulation of traveling salesman problems. *Journal of the ACM*, 7(4):326–329, 1960. doi: 10.1145/321043.321046.

[23] B. Petersen, D. Pisinger, and S. Spoorendonk. Chvátal-Gomory rank-1 cuts used in a Dantzig-Wolfe decomposition of the vehicle routing problem with time windows. In B. Golden, R. Raghavan, and E. Wasil, editors, *The Vehicle Routing Problem: Latest Advances and New Challenges*, pages 397–420. Springer, 2008. doi: 10.1007/978-0-387-77778-8_18.

[24] G. Righini and M. Salani. Symmetry helps: bounded bi-directional dynamic programming for the elementary shortest path problem with resource constraints. *Discrete Optimization*, 3(3):255–273, 2006. doi: 10.1016/j.disopt.2006.05.007.

[25] G. Righini and M. Salani. New dynamic programming algorithms for the resource constrained shortest path problem. *Networks*, 51(3):155–170., 2008. doi: 10.1002/net.20212.

[26] M. M. Solomon. Algorithms for the vehicle routing and scheduling problem with time window constraints. *Operations Research*, 35(2):234–265, 1987. doi: 10.1287/opre.35.2.254.

[27] S. Spoorendonk and G. Desaulniers. Clique inequalities applied to vehicle routing problem with time windows. 2008.

[28] S. Spoorendonk, G. Desaulniers, and J. Desrosiers. A note on cutting planes in Dantzig-Wolfe decompositions of integer programs. 2008.

[29] P. Toth and D. Vigo, editors. *The Vehicle Routing Problem.* SIAM, 2002.

[30] L. A. Wolsey. *Integer Programming.* John Wiley & Sons, Inc., 1998.

Chapter 10

Optimal Routing with Failure Independent Path Protection

Thomas Stidsen
DTU Management, Technical University of Denmark

Bjørn Petersen
DIKU Department of Computer Science, University of Copenhagen

Kasper Bonne Rasmussen
Department of Computer Science, ETH Zurich, Switzerland

Simon Spoorendonk
DIKU Department of Computer Science, University of Copenhagen

Martin Zachariasen
DIKU Department of Computer Science, University of Copenhagen

Abstract

Reliable communication has become crucial in today's information society. Modern communication networks are required to deliver reliable communication to their customers. Unfortunately, protection against network failures significantly hampers efficient utilization of network investments, because the associated routing problems become much harder. In this paper we present a rigorous mathematical analysis of one of the most promising protection methods: Failure independent path protection. We present an LP model which is solved by column generation. The subproblem is proven to be strongly $\mathcal{NP}$-hard, but still solvable for medium sized networks through the use of specialized dynamic programming algorithms. This enables us to evaluate the performance of failure independent path protection for 8 networks with up to 37 nodes and 57 links. The results indicate that only between 3% and 8% extra network capacity is necessary when compared to the capacity required by complete rerouting (which is the absolute lower bound

In revision. Preliminary conference paper published in the proceedings of the International Network Optimization Conference (INOC), 2007

for single link failure protection).

Keywords: Failure independent path protection, column generation, protection capacity minimization

1 Introduction

Today's information society relies increasingly on advanced communication networks. This has led to massive investments in increased communication network capacity. In order to utilize these investments the network operators perform *t*raffic engineering, i.e., they route communication to maximize the utilization of the capital invested in the communication network.

Most of the backbone networks which today carry long distance communication traffic use path based routing, i.e., a communication connection between two points in the network is established along one or more fixed paths. Despite the huge success of the packet switched Internet, path based routed network technology will continue to be the dominant technique of backbone networks, because traffic engineering can be performed much more efficiently than in packet switched networks. Examples of such path switched network technologies are SDH/SONET or DWDM networks or circuit switched network technologies like PSTN/ISDN. Furthermore, the new Multi Path Label Switching (MPLS) [34] protocol enables packets to be routed on fixed paths.

The standard model of a path switched communication network is a directed graph $G = (V, A)$ consisting of a set of nodes V and a set of arcs A. The nodes correspond to telecommunication switches. The telecommunication switches route the communication signals through cables. We will assume that all cables enable bidirectional communication and therefore we will model one cable using two arcs, one each way between the end nodes. We assume that a static communication connection demand is given which requires one-way communication between an origin node o_k and a terminating node d_k of volume ρ_k for a set of demands $k \in K$. For each demand k we should construct a single *primary* (or working) path from o_k to d_k, and all the required volume of traffic ρ_k should be sent over this primary path (i.e., traffic should be non-bifurcated).

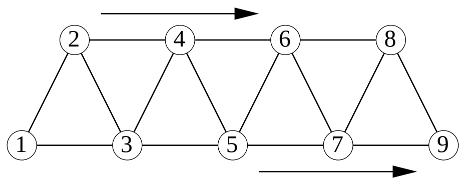
Communication networks are increasingly required to be *reliable*. If we cannot trust our messages to reach the receiver, the use of a communication network is limited. Communication networks are prone to failures and many different types of failures can occur. Switches (nodes) can lose power, experience software and hardware failures, etc. Cables (arcs) can be cut by entrepreneurs or by natural disasters. For simplicity, in this paper we will only consider single cable failures, i.e., simultaneous failure of the two arcs which correspond to a cable. This is a well-known and widely used simplification [15, 26].

Multiple cable failures can occur in networks, but are less probable. Several cables can fail, if, e.g., a switch fails or a single cable failure in a lower network layer may result in multiple failures in the upper layers. These kind of network errors are of increasing importance but they also make network protection significantly harder, e.g., the problem of finding failure independent paths is $\mathcal{NP}$-complete in the face of multiple cable failures [17].

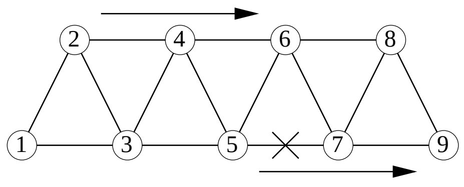
When a cable fails, the network operator either has to repair the cable or *re-route* the failed paths around the failure. Because repairing a cable can take considerable time, rerouting is an interesting alternative. The main problem with rerouting is that enough capacity needs to be available on the remaining non-failed cables to enable rerouting. Traffic engineering which takes into account the possibility of a cable failure becomes significantly more complex, but is again important in order to utilize network investments.

In this paper we assume that traffic which is routed along one primary paths is rerouted along the same *backup* (or protection) path. Hence rerouted traffic is non-bifurcated. The cost function is simple: We assume that a linear cost term c_a for using capacity on arc a has to be paid. The required capacity of an arc is the maximum capacity required for all failure situations (the network should be able to accommodate necessary rerouting). The total cost of the network is the sum of costs over all arcs. It should be noted that in our model arcs have no capacity bounds — in contrast to the well-known multi-commodity flow model [1].

In Figure 1(a) two paths are established, from node 2 to node 6 and from node 5 to node 9, both with a volume of 1, that is, $(o_1, d_1, \rho_1) = (2, 6, 1)$ and $(o_2, d_2, \rho_2) = (5, 9, 1)$. In Figure 1(a) — and all the other figures in this paper — we have only drawn the bidirectional cables, and *not* the two corresponding arcs for each cable, in order not to complicate the figures unnecessarily. The necessary capacity of a cable corresponds to the sum of the necessary arc capacities for that cable. Given the paths chosen in Figure 1(a) an arc capacity of 1 is then required on the arcs $(2, 4)$, $(4, 6)$, $(5, 7)$ and $(7, 9)$, resulting in a total required Non-Failure (NF) network capacity of 4. In Figure 1(b) the cable between node 5 and node 7 fails resulting in the failure of arc $(5, 7)$ and arc $(7, 5)$. This results in a communication breakdown for the path from node 5 to node 9.



(a) Two paths



(b) A cable break

Figure 1: Path switched routing.

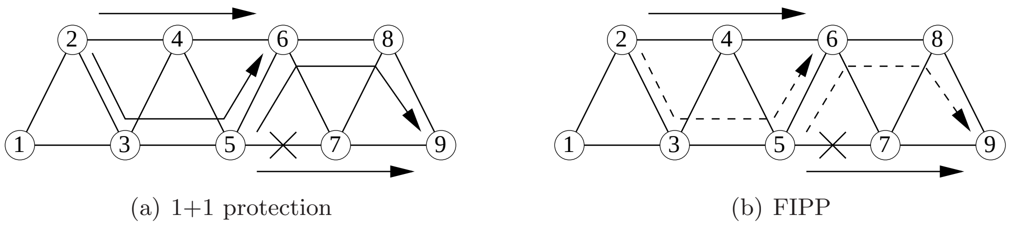
In order to protect communication against a cable failure, a rerouting strategy needs to be planned for each possible cable failure, i.e., a protection method needs to be installed. (Because rerouting methods protect against failures, we will use rerouting methods and protection methods interchangeably.) The importance of network reliability and the importance of minimizing network investments have resulted in a large number of rerouting methods. It is beyond the scope of this paper to review these and we refer the reader to [15] for a recent and comprehensive survey. One of the promising methods is p-cycle protection. This is a clever extension of the well-known ring protection scheme, which significantly improves the capacity requirements necessary for protection [15, 31]. Furthermore, the use of p-cycles enable fast protection of communication, as provided by ring protection. Despite these promising features, p-cycles have not (yet) achieved widespread application.

In this paper we will consider traffic engineering optimization methods for the Failure Independent Path Protection (FIPP) method for path switched networks. In this protection method the backup path for a given demand is independent of the failure related to the primary path, i.e., independent of which of the cables in the primary path have failed. This protection method is also called Shared Backup Path Protection in [15] or Global Backup Path Protection in [6].

The outline of the paper is as follows. In Section 2 we give a brief description of different path protection methods. This leads us to focus on the FIPP method for which we give a mathematical model in Section 3. In the same section we also present a column generation algorithm to solve a relaxed model and discuss the computational complexity of the subproblem. In Section 4 we then present and discuss the results when applying the column generation algorithm to a number of test cases. In Section 5 we discuss possible extensions and in Section 6 we draw some conclusions.

2 Path protection method

The classic path protection method employed in path switched networks is 1+1 protection. Figure 2(a) shows how the 1+1 protection method can be used to protect the path connections from Figure 1. In 1+1 protection, two cable disjoint paths (and hence arc disjoint paths) are established and actively used. If an arc fails on one path, the other path will survive and enable the receiving node to restore communication by just switching to the other incoming signal. This method is simple, there are well-defined standards, but the required network capacity is always at least twice the required non-failure network capacity. The total network capacity required in the example in Figure 2(a), assuming the same demands, is 10. Notice in particular that a capacity of 2 is required on arc $(5, 6)$.



(a) 1+1 protection (b) FIPP

Figure 2: Capacity sharing illustrated.

2.1 Comparing path protection methods

We now define two measures: Restoration Over Build (ROB) network capacity and Relative Restoration Over Build (RROB) network capacity.

ROB: The extra network capacity necessary to ensure protection, i.e., the network capacity for both routing and protection minus the NF network capacity, assuming shortest path routing. In the example from Figure 2(a), $ROB = 10 - 4 = 6$.

RROB: The relative extra network capacity necessary to ensure protection, i.e., the ROB network capacity divided by the NF network capacity. In the example from Figure 2(a), $RROB = \frac{10-4}{4} = 1.5$, meaning that 1+1 protection in this case costs 150% extra network capacity compared to the necessary non-failure network capacity.

The FIPP method is a slight variation of 1+1 protection: Instead of actively sending data packets on both paths, one path is designated the primary path and only when that path fails will the data packets be sent along the backup path. In Figure 2(b) the same two protected connections as in Figure 2(a) are shown, but now there is a primary path (full line) and a backup path (dashed line) for each path. But the required network capacity has decreased.

The arc $(5, 6)$ now only needs a capacity of 1, because the backup paths are not being used at the same time. This concept is called *sharing* and is possible because we only guarantee protection against single cable failures and because the two primary paths are cable disjoint. For the FIPP method, the NF network capacity is again 4, but the ROB network capacity is now 9, which leads to an RROB network capacity of 1.25.

In order to utilize the path protection methods traffic engineering has to be performed in order to minimize the RROB network capacity. When working with 1+1 protection this is a well-studied problem for which there exist polynomial-time algorithms [4, 33]. This is *not* the case for the FIPP method. Because of the possibility of sharing the capacity for the backup paths, the best choice of primary path and backup path for each end-to-end demand node pair becomes interdependent.

A practical solution to the FIPP traffic engineering problem is studied in [23]. In order to simplify the problem, the dependency between different protected communication connections is ignored in [23]. Instead, the focus is on algorithms which can find pairs of disjoint paths, where the cost of backup paths is assumed to be some constant factor cheaper than the primary paths. Because of the sharing possibility it is reasonable that the capacity costs for each arc of the backup path are less than the capacity costs for each arc of the primary path. Even this simplified problem is $\mathcal{NP}$-hard [23] and a number of different heuristics are suggested to find good, though not optimal, solutions to the problem. This line of research is continued in [22]. It should be emphasized that the cost model for backup paths used in [22, 23] is approximate. We quantify the *exact* relationship between costs for primary and backup paths in Section 3.1 and prove that the resulting optimization problem is strongly $\mathcal{NP}$-hard.

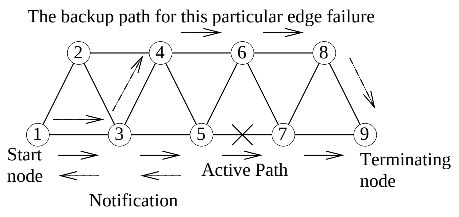
In [26] the full FIPP traffic engineering problem is considered. A column generation approach, similar to the approach in this paper, is considered. The same mathematical model for the column generation master problem is formulated, but the subproblem is *not* formulated. This means that if an optimal solution is required, the full set of disjoint paths has to be pre-generated, and this is only feasible for small networks.

2.2 Different path protection methods

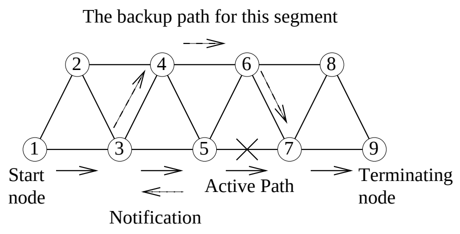
The Failure Independent Path Protection method is just one example of a path protection scheme, and there are a number of other methods. The different path protection methods all use one primary path, but protect the primary path in different ways. In Figure 3, which is (partly) taken from [6], six path protection methods are presented. If the path protection methods are only allowed to choose the backup path based on the failed cable, this list is complete, but a number of additional variations exists, some of which are described in Section 2.3.

Full Backup Path Protection (FBPP)

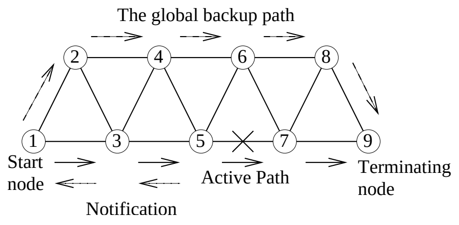
Theoretically FBPP [24], see Figure 3(a), is the most efficient path protection method. (This method is not included in [6].) Given a primary path, each cable which can fail on the primary path is protected by a unique backup path. There are no limitations regarding these backup paths, except they are, obviously, not allowed to use any of the two failed arcs in the cable which they protect. This gives the highest possible freedom in choosing the cheapest protection paths and all the other path protection methods are more restrictive in the choice



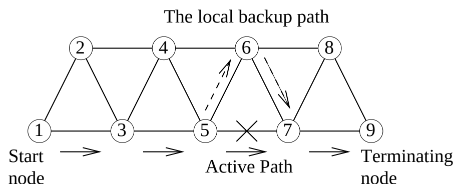
(a) Full Backup Path Protection



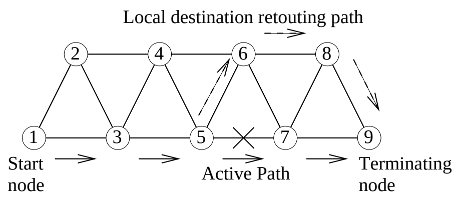
(b) Segment Backup Path Protection



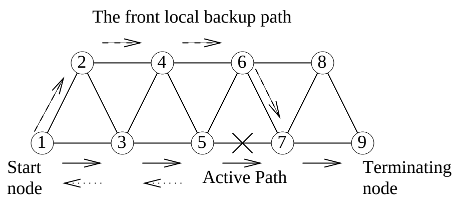
(c) Failure Independent Path Protection



(d) Local Backup Path Protection



(e) Local Destination Rerouting



(f) Front Dynamic Backup Path Protection

Figure 3: Different path protection schemes.

of backup paths and hence more costly.

Segment Backup Path Protection (SEBPP)

SEBPP, see Figure 3(b), protects segments (sets of cables) of the primary path with the same backup path. Hence several cables in the same segment are forced to share backup paths.

Failure Independent Path Protection (FIPP)

FIPP, see Figure 3(c), limits the choice of backup path even further, such that only one backup path is allowed. This forces the backup path to be cable disjoint with the primary path.

Local Backup Path Protection (LBPP)

LBPP [24], see Figure 3(d), performs a local protection, i.e., the rerouting paths are required to lead from one node of the failed cable to the other node of the failed cable. This resembles the classical span protection, but in this case different reroute paths may be chosen for each connection.

Local Destination Rerouting (LDR)

LDR [2], see Figure 3(e), is a variation of local protection, where the connection paths are rerouted directly to the end node of the connection. LDR preserves the fast rerouting time of Local Backup Path Protection, but is more efficient regarding ROB network capacity.

Front Dynamic Backup Path Protection (FDBPP)

FDBPP, see Figure 3(f), is another variation of local protection, where the connection path is rerouted from the start node to the end node of the failed cable. To the best of our knowledge this type of protection has not been suggested anywhere else and is only included to make the list of path protection methods complete. We do not expect the FDBPP method to be implemented anywhere.

2.3 Further variations

The description of the different path protection schemes is very simplified and a number of variations can be added. Here we briefly mention two of these.

Stub-release is a technique which can be applied to further lower the required network capacity. The idea is that in case of a failed cable, the unharmed parts of the primary path, which are not in use any longer, are released and can be used for protection [25]. Stub-release can improve the capacity efficiency of each method, with the exception of the Local Backup Path Protection method, at the price of a more complicated protection scheme.

To speed up the recovery process, Hashkin protection can be applied [16]. The idea is to loop-back the communication signals at the switch just before the failed cable, to where the backup path starts. Hashkin protection minimize packet-loss, but requires more network capacity and cannot be used in Local Backup Path Protection and Local Dynamic Backup Path Protection.

2.4 Motivation for FIPP

Out of the 6 different types of path protection described in Section 2.2, we only consider the FIPP method in this paper.

FIPP is the only path protection method for which the protection *action* does not depend on which cable actually fails — it is *failure independent*. This makes FIPP the simplest of the path protection methods. Furthermore, the complex switching schemes take place at the start node of the connection path, which may be an advantage in future networks. It is *not* the most capacity efficient path protection method. The most efficient method is FBPP, but FBPP requires administration of a large number of backup paths. Furthermore, in Section 4 we demonstrate that the FIPP method is indeed a *very* efficient protection method, when optimal routing of the primary path and the backup path is performed.

The main disadvantage with the FIPP method is the relatively long restoration time, i.e., the time it takes to restore communication. This is because of the notification time – which is the backward communication time between the node which observes the failure and the node from which the connection paths originates. We have illustrated the notification time by dotted arrows in Figure 3 for the path protection methods for which this is necessary. For a more complete discussion of restoration time, we refer to [6].

3 LP model and column generation approach

In this section we start by defining the FIPP optimization problem formally. Then we present an LP model for a relaxed version of the FIPP optimization problem, the so-called *fractional* FIPP optimization problem. The LP model has an exponential number of variables, and hence we solve it using column generation. In Section 3.1 we describe the associated pricing problem (or subproblem). A MIP model for solving the subproblem is given in Section 3.2, and in Section 3.3 we show that the subproblem is in fact strongly $\mathcal{NP}$-hard. Finally, in Section 3.4 we give a labeling algorithm for solving the subproblem, and summarize our column generation algorithm in Section 3.5.

Given, as previously defined, a directed graph $G = (V, A)$ with nodes V and arcs A. For each failure situation $s \in S$ we have a set of failed arcs $F_s \subseteq A$. There is a cost c_a for using one unit of capacity of an arc a. We further assume to know a static set of demand node pairs for which protected connections using the FIPP method should be established. A directed connection between an origin node o_k and a terminating node d_k with a volume of ρ_k should be established for each demand $k \in K$. The optimization objective is to minimize the cost of the required capacity when applying the FIPP method to protect the established connections. This means that for each demand a *pair* of directed failure disjoint paths needs to be found: A primary path p^{pri} and a backup path p^{bac}, both connecting node o_k to node d_k. Such a pair of failure disjoint paths is denoted a *path pair* $\pi = (p^{pri}, p^{bac})$. The objective in the FIPP problem is to find a path pair for each demand $k \in K$, such that the total cost of the capacity required is minimized. Note that the capacity required by an arc is the maximum capacity required taken over all failure situations.

Given these definitions we are ready to present an LP model for the fractional FIPP optimization problem. In this problem we allow more than one path pair to accommodate the flow required by a demand. Let P_k be the set of path pairs that can satisfy demand k, that is, the set of primary/backup paths that connect origin node o_k with terminating node d_k. Let $P_k(a) \subseteq P_k$ be the subset of path pairs for which the *primary* path uses arc $a \in A$. Similarly, let $P_k(a, s) \subseteq P_k$ be the set of path pairs for which the *primary* path fails and the *backup* path uses arc $a \in A \setminus F_s$ in failure situation $s \in S$. Finally, let variable λ^k_π denote the amount of communication flow through path pair $\pi \in P_k$, and let variable θ_a denote the capacity required for arc $a \in A$.

FIPP

minimize:

$$\sum_{a \in A} c_a \cdot \theta_a \tag{1}$$

subject to:

$$\sum_{\pi \in P_k} \lambda^k_\pi \quad \geq \quad \rho_k \quad \forall\, k \in K \tag{2}$$

$$\sum_{k \in K} \sum_{\pi \in P_k(a)} \lambda^k_\pi + \sum_{k \in K} \sum_{\pi \in P_k(a,s)} \lambda^k_\pi \quad \leq \quad \theta_a \quad \forall\, s \in S,\ a \in A \setminus F_s \tag{3}$$

$$\lambda^k_\pi, \theta_a \in R_+$$

The objective function is given by (1) and it is the cost of the summed network capacity. The demand constraint (2) ensures that enough capacity is established on the path pairs. The capacity constraint (3) ensures that enough capacity is allocated to route the communication on each arc a in each failure situation s which does not disrupt the arc.

The problem with this LP-model is that the number of path pairs grows exponentially with the network size, and hence the complete model can only be solved for small network sizes. Instead, we will use a column generation algorithm such that only a subset of the path pairs is generated. The optimization subproblem to generate new path pairs with negative reduced costs is given in Section 3.1, and in Section 3.5 the column generation algorithm is given.

It is clear that the fractional FIPP optimization problem is a relaxation of the original FIPP optimization problem which is $\mathcal{NP}$-hard [32]. The hardness of the fractional FIPP optimization problem on the other hand is still an open problem. The LP model can therefore be used for lower bounding in a branch-and-price algorithm for the FIPP optimization problem. The bound can however be weak, because the bound of the *relaxed* FIPP model is equivalent to the bound of the relaxed FBPP model, if the primary paths consists of one link. For primary paths of one link, each of the backup paths for the FBPP model can be constructed by generating path pairs, i.e., the one hop primary path and different backup paths. For primary paths which are not one hop however, the relaxed FIPP model and the relaxed FBPP model are not equivalent, because in the FIPP model the feasible backup paths are more limited than the feasible backup paths for the FBPP model. In other words, it will depend on the network and the communication demand how good a bound the relaxed FIPP model can deliver compared to the bound of the FBPP model.

3.1 Subproblem: Quadratic Cost Disjoint Path Problem

For the master problem for FIPP optimization problem let $\alpha_k \geq 0$, $k \in K$, be the dual variables associated with the (negated version of) constraint (2), and let $\beta_a^s \geq 0$, $s \in S$, $a \in A \setminus F_s$, be the dual variables associated with constraint (3). Our task is to decide if there exists a pair of primary and backup paths $\pi = (p^{pri}, p^{bac})$ from some origin node o_k to some terminating node d_k with negative reduced cost for some $k \in K$.

The reduced cost of a pair of paths (p^{pri}, p^{bac}) is computed as follows. The cost of an arc $a \in p^{pri}$ is $\sum_{s\in S} \beta_a^s$, while the cost of an arc $a \in p^{bac}$ is $\sum_{s\in S: F_s \cap p^{pri} \neq \emptyset} \beta_a^s$. Note the asymmetry in the definition of arc costs in primary and secondary paths: For an arc on the primary path the cost is the sum taken over *all* failure situations, while for an arc on the backup path the sum is only taken over the failure situations that affect an arc on the *primary* path. The total reduced cost of (p^{pri}, p^{bac}) is now

$$-\alpha_k \ + \ \overbrace{\sum_{a\in p^{pri}} \sum_{s\in S} \beta_a^s}^{primary\ path\ cost} \ + \ \overbrace{\sum_{a\in p^{bac}} \ \sum_{s\in S: F_s \cap p^{pri} \neq \emptyset} \beta_a^s}^{backup\ path\ cost}$$

The *Quadratic Cost Disjoint Path Problem (QCDPP)* is to compute a pair of paths $\pi = (p^{pri}, p^{bac})$ with minimum total cost. The name of the problem comes from the fact there is a pairwise (or quadratic) dependence on the cost of the backup path as a function of the primary path. Since the dual variables β_a^s are non-negative, there clearly exists an optimal solution where both the primary path p^{pri} and the backup path p^{bac} are simple. Hence in the following we require that the paths p^{pri} and p^{bac} are simple and arc disjoint.

3.2 MIP model for QCDPP

A primary path is defined by the binary variables x_a for all $a \in A$ and a backup path is defined by the binary variables y_a for all $a \in A$. We define the sets $\delta^+(i)$ as the arcs going out of node $i \in V$ and $\delta^-(i)$ as the set of arcs going into node $i \in V$. We again use the set of failed arcs F_s and define the cardinality of the set as $|F_s|$, i.e., the number of arcs which fails in situation $s \in S$. The binary variables u_s for all $s \in S$ detect whether the primary path is interrupted by failure s and the binary variables v_s for all $s \in S$ detect whether the backup path is interrupted by failure s. Furthermore, the auxiliary variables z_s^a for all $s \in S$ and all $a \in A$ detect if the primary path is interrupted by failure s at the same time as the backup path use arc a.

QCDPP

minimize:

$$c^k_{reduced} = -\alpha_k \; + \; \overbrace{\sum_{a \in A} \sum_{s \in S} \beta_a^s \cdot x_a}^{primary\ path\ cost} \; + \; \overbrace{\sum_{a \in A} \sum_{s \in S} \beta_a^s \cdot z_s^a}^{backup\ path\ cost} \tag{4}$$

subject to:

$$\sum_{a \in \delta^+(i)} x_a - \sum_{a \in \delta^-(i)} x_a = \begin{cases} 1 & i = o_k \\ -1 & i = d_k \\ 0 & \text{otherwise} \end{cases} \quad \forall\, i \in V \tag{5}$$

$$\sum_{a \in \delta^+(i)} y_a - \sum_{a \in \delta^-(i)} y_a = \begin{cases} 1 & i = o_k \\ -1 & i = d_k \\ 0 & \text{otherwise} \end{cases} \quad \forall\, i \in V \tag{6}$$

$$|F_s| \cdot u_s \geq \sum_{a \in F_s} x_a \quad \forall\, s \in S \tag{7}$$

$$|F_s| \cdot v_s \geq \sum_{a \in F_s} y_a \quad \forall\, s \in S \tag{8}$$

$$u_s + v_s \leq 1 \quad \forall\, s \in S \tag{9}$$

$$z_s^a \geq u_s + y_a - 1 \quad \forall\, s \in S,\ a \in A \tag{10}$$

$$x_a, y_a, u_s, v_s \in \{0,1\}, \qquad z_a^s \in [0,1] \tag{11}$$

The objective function (4) is the reduced cost $c^k_{reduced}$ of the two disjoint paths. The first double sum calculates the costs for the primary path. The second double sum then calculates the cost for the backup paths. Notice that each arc a in the backup path only costs β_a^s in situation s if the primary path is disrupted in failure situation s. This is detected by the variable z_s^a. Finally the dual value α_k from constraint (2) is subtracted to calculate the corresponding reduced cost. Both the primary path variables x and the backup path variables y are constrained to form paths by constraint (5) and (6), respectively. The path disruption variables, u for the primary path and v for the backup path, are set by constraint (7) and (8) respectively. Variables u and v are then used in constraint (9) to ensure failure disjointness of the paths. In constraint (10) the auxiliary variable z_s^a is forced to the value 1 if the primary path is disrupted in situation s and the backup path uses the arc a. Finally the domains of the variables are given by constraint (11).

We consider two variants of failure situations: In the *single arc* failure variant there is one failure situation for each arc in A. In the *single link* failure variant there is one failure situation for each pair of opposite arcs, i.e., when the corresponding undirected edge is broken.

In Section 3.3 it is proved that the sub-problem above is $\mathcal{NP}$-hard. However, if instead the primary paths were pre-calculated and the task was to find the best usage of the primary paths, at the same time finding the best backup paths, the sub-problem would be a simple shortest path problem (with links of the primary path removed from the network).

3.3 $\mathcal{NP}$-hardness of QCDPP

We now prove that QCDPP is strongly $\mathcal{NP}$-hard for the single arc and single link failure variants. First we present the proof for the single arc variant and then we indicate how this leads to an $\mathcal{NP}$-hardness proof for the single link variant. In the single arc variant the set of failure situations S is identical to the set of arcs A. The decision version of QCDPP with single arc failures is formally defined as follows (where the constant term $-\alpha_k$ in the objective function of QCDPP is ignored).

INSTANCE: Directed graph $G = (V, A)$, pairwise (integer and non-negative) costs β_a^f for all ordered pairs of arcs $(a, f) \in A \times A$, origin node $o_k \in V$, terminating node $d_k \in V$ and integer C.

QUESTION: Does there exist a pair of simple arc disjoint paths $\pi = (p^{pri}, p^{bac})$ from o_k to d_k in G such that

$$\sum_{a \in p^{pri}} \sum_{f \in A} \beta_a^f + \sum_{a \in p^{bac}} \sum_{f \in p^{pri}} \beta_a^f \leq C \quad ?$$

We prove that this problem is $\mathcal{NP}$-complete by reduction from 3-SATISFIABILITY (3SAT) [14]. It is obvious that the decision version of QCDPP is in $\mathcal{NP}$, since given $\pi = (p^{pri}, p^{bac})$ we can compute the corresponding cost and compare it to C in polynomial time.

Let (U, C) be an instance of 3SAT, where $U = (x_1, x_2, \ldots, x_n)$ is a finite set of n variables and $C = (c_1, c_2, \ldots, c_m)$ is a set of clauses where $|c_i| = 3$, $i = 1, \ldots, m$. We assume without loss of generality that each variable appears in at least one clause.

Based on the 3SAT instance we create an instance of the QCDPP with the structure illustrated in Figure 4. The graph consists of two chains of arcs – the so-called top chain and the bottom chain. Two node disjoint paths from o_k to d_k must necessarily have the property that one of the paths travels through the top chain while the other travels through the bottom chain. By assigning costs appropriately, we will force the primary path to use the bottom chain and the backup path to use the top chain.

We will first assume that we seek two *node* disjoint paths from o_k to d_k in this graph. Later we describe how we can modify the graph so that the paths become *arc* disjoint. Furthermore, the graph that is shown is a directed multigraph, and later we also describe how this graph can be transformed into an ordinary directed graph.

The arcs in the top chain are denoted *variable* arcs, while the arcs in the bottom chain are denoted *clause* arcs. For each clause $c_i \in C$ we have 8 parallel arcs, one for each combination of assignments for the three literals; these assignments are denoted 000, 001, 010 etc. As

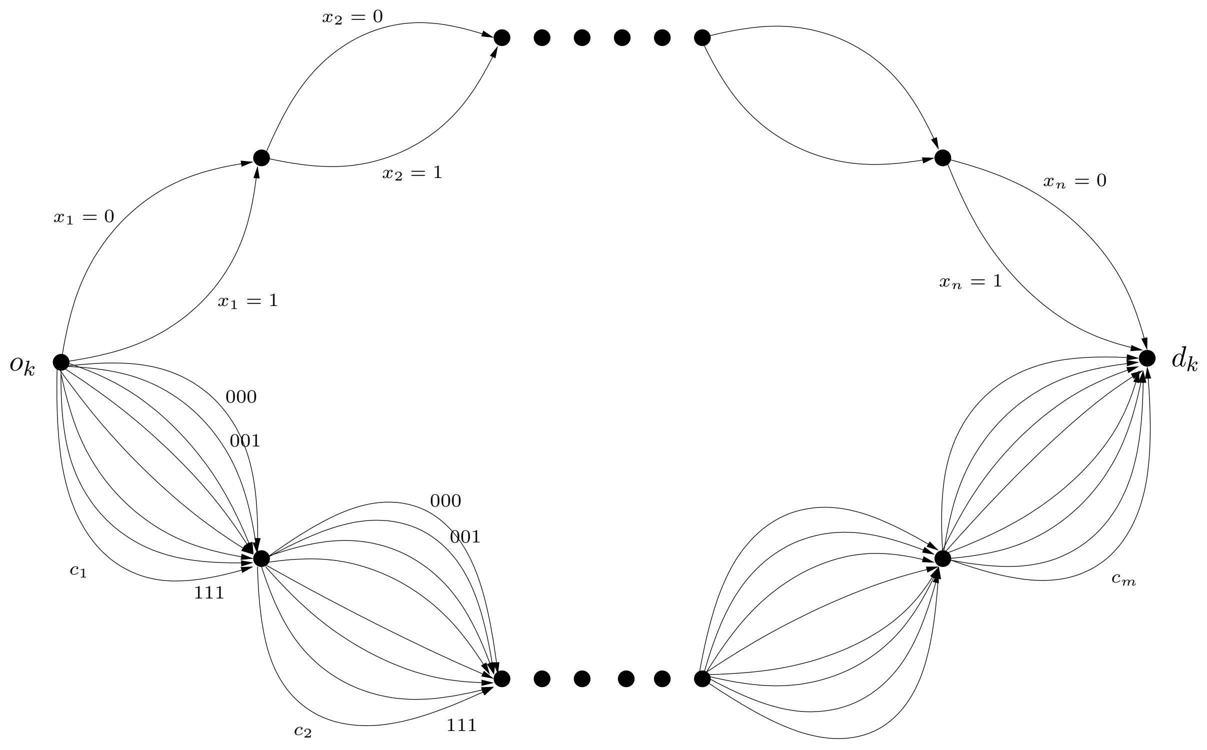


Figure 4: Graph construction for $\mathcal{NP}$-completeness proof.

an example, for the clause $(x_1 \vee x_2 \vee \bar{x}_3)$ the assignment 011 means that $x_1 = 0$, $x_2 = 1$ and $x_3 = 0$. Note that an assignment different from 000 corresponds to a satisfied clause. Similarly, we have two variable arcs for each variable x_j, one arc for $x_j = 0$ and one arc for $x_j = 1$.

We will now assign pairwise costs β_a^f for all ordered pairs of arcs $(a, f) \in A \times A$. We set $\beta_a^f = 0$ for all $(a, f) \in A \times A$ *except from* the following pairs:

- For a *clause* arc a corresponding to the assignment 000 we have $\beta_a^{f'} = 1$ for one arbitrary variable arc f' (say, the arc corresponding to $x_1 = 0$). This means that if the arc a is used by a primary path from o_k to d_k then the cost of a is $\sum_{f \in A} \beta_a^f = 1$.
- For a *variable* arc a and clause arc f, if the variable assignment given by arc a does *not* match the clause assignment given by arc f, then $\beta_a^f = 1$. As an example, the variable arc a corresponding to $x_3 = 1$ has $\beta_a^f = 1$ for the arc f corresponding to the clause $(x_1 \vee x_2 \vee \bar{x}_3)$ with assignment 011. In Table 1 an extended example on how costs are assigned for variable arcs is given.

  Since we assume that each variable appears in at least one clause, each variable edge a has cost at least 1 as a primary edge, since there will be at least one clause assignment that does not match with the variable assignment given by a.

Finally, we set $C = 0$ in the QCDPP instance. Now we prove that we have YES-instance for QCDPP if and only if we have a YES-instance for 3SAT.

Consider a YES-instance for QCDPP, that is, an instance with zero cost. Such an instance must have a primary path p^{pri} following the *clause* arcs from o_k to d_k, since the variable arcs

Assignment	$x_1 = 0$	$x_1 = 1$	$x_2 = 0$	$x_2 = 1$	$x_3 = 0$	$x_3 = 1$
000	0	1	0	1	1	0
001	0	1	0	1	0	1
010	0	1	1	0	1	0
011	0	1	1	0	0	1
100	1	0	0	1	1	0
101	1	0	0	1	0	1
110	1	0	1	0	1	0
111	1	0	1	0	0	1

Table 1: Costs β_a^f associated with variable arcs a for clause f being equal to $(x_1 \vee x_2 \vee \bar{x}_3)$.

have positive costs as primary path arcs. Consequently, the backup path p^{bac} must follow the *variable* arcs from o_k to d_k. Since the total cost of the solution $\pi = (p^{pri}, p^{bac})$ is zero, all arcs of the path p^{pri} correspond to clauses being satisfied (i.e., are different from the clause assignments 000 which have cost 1 as primary path arcs). Also, since the total cost of $\pi = (p^{pri}, p^{bac})$ is zero, the variable arcs followed by p^{bac} match the assignments in the clause arcs. Therefore, assigning the variables x_j, $j = 1, \ldots, n$, to the values indicated by the path p^{bac} gives a satisfying assignment for the 3SAT-instance.

For the other direction, consider a YES-instance for 3SAT. By letting p^{bac} follow the variable arcs in the QCDPP instance as given by a satisfying 3SAT-assignment, and letting p^{pri} follow the clause arcs corresponding to the 3SAT-assignment, we obtain a solution to QCDPP of total cost zero.

By splitting each node in the graph (apart from o_k and d_k) – that is, replacing the node with an arc (u, v), and connecting all in-coming arcs to u and all out-going arcs to v – we force the paths to be *edge* disjoint. Furthermore, the multigraph can be transformed into an ordinary directed graph G by replacing each arc in the multigraph by a sequence of two arcs, and assigning pairwise costs appropriately. Thus we have the following:

Theorem 1 *The decision version of QCDPP when reduced to single* arc *failures is $\mathcal{NP}$-complete even when all pairwise costs are 0 or 1 (and only distinct pairs of arcs can have non-zero costs).*

Consider the directed graph G resulting from the above construction. If, for each arc in G, we add an arc in the opposite direction we obtain a graph G', where bidirectional communication is feasible for each underlying link. Consider the single *link* failure variant of QCDPP for the graph G', where the costs are assigned as in the construction above, but where the β_a^f costs are replaced with β_a^l costs (where l corresponds to a link). Since the primary and backup paths in G' should be simple, no backward arcs in G' will ever be used, and therefore we obtain the following:

Theorem 2 *The decision version of QCDPP when reduced to single* link *failures is $\mathcal{NP}$-complete even when all pairwise costs are 0 or 1 (and non-overlapping pairs of arcs and links can have non-zero costs).*

3.4 Labeling algorithm for the QCDPP

The QCDPP can be formulated as a Shortest-Path Problem with Resource Constraints (SP-PRC). The SPPRC is a common subproblem in many graph based problems when using a column generation based algorithm, e.g., the Vehicle Routing Problem with Time Windows [20, 21] and the Crew Pairing Problem [8]. In the following we will shortly define the SPPRC, discuss complexity issues and the application of recent developments within this area, and describe the basic labeling algorithm. Last we will present the reformulation of the QCDPP into an SPPRC.

The SPPRC can be stated as: Given a weighted directed graph $G' = (V', A')$ with nodes V' and arcs A', and a set of resources R. For each node $i \in V'$ and arc $(i, j) \in A'$ there is a weight of each resource $r \in R$ that is determined by a (not necessarily linear but often constant) function, as well as a lower and upper limit on r. For a sub-path in G' there is a resource accumulation of resource $r \in R$ when visiting node i or traversing arc (i, j), i.e., an amount of resource r is accumulated on the path. The total amount of r must respect the lower and upper limits of r in when arriving at node $i \in V'$ or when using arc $(i, j) \in A$. The increase in resource consumption and cost of a path when extended along an arc is defined by a function, that are sometimes denoted *resource extension functions*, see [18]. The objective is to find a minimum cost path from an origin node $o \in V'$ to an destination node $d \in V'$, where the resources satisfy the limits for all resources $r \in R$. In many cases it suffices to have the limits of the resources only at the nodes; in these cases the limits on the edges can be made non-binding.

The SPPRC is $\mathcal{NP}$-hard in the weak sense when the number of resources is a constant and can be solved with dynamic programming based labeling algorithms in pseudo-polynomial time. An extension of the SPPRC is the node elementary version; the elementary shortest path problem with resource constraints (ESPPRC) where paths must be simple. The elementarity constraint can be enforced with the use of a binary resource for each node to indicate if the node is visited on the path and solved as an SPPRC. The ESPPRC is strongly $\mathcal{NP}$-hard, see [11]. However, if G' does not contain negative weight cycles the additional resources can be disregarded since a least weight path that is simple will always exist, hence the problem can be solved in pseudo-polynomial time. Although the reformulation (see details below) of the QCDPP into a SPPRC leads to a graph with no negative weight cycles, the number of resources amounts to one binary resource per failure scenario, i.e., one per two arcs in G for the single link failure case in the QCDPP. That is, the number of resources in the SPPRC depends on the input of the QCDPP, hence the complexity of the labeling algorithm is exponential when regarding the reformulation of the QCDPP. Also, it is important to note that the reformulation of the QCDPP into a SPPRC results in a non-constant extension function where the weight of the arcs on the backup path depend on the failure scenarios that are affected by the primary path.

A comprehensive overview of work related to SPPRC is outside the scope of this article, but we will briefly discuss some recent results. For further details on mathematical models and solution methods we refer the reader to the survey of Irnich and Desaulniers [19]. Dynamic programming based methods denoted *labeling algorithms* are to date the most dominant approach to solving the SPPRC. However, recently Carlyle et al. [7] present a Lagrangian relaxation based method. The approach is applicable for problems with no negative weight cycles and shows good results when few resources are considered. However, due to the nature of the non-constant extension function on the arc weights in our reformulation this approach is not directly applicable; also we consider a large number of resources which may limit the effect of the Lagrangian relaxation.

Dumitrescu and Boland [12] present an improved preprocessing for the SPPRC (with no negative cost cycles) and embed it into a labeling algorithm. They present resource lower bound calculations using Lagrangian relaxation, hence solving a shortest path problems. Again this approach is not applicable in our case due to the arc weight extension function in our reformulation. Furthermore, this approach have very limited use when only considering binary resources, which is indeed the case for our reformulation, since the resource bounds are already very tight. Feillet et al. [13] address the ESPPRC and propose to consider unreachable nodes instead of visited nodes with the binary resources. The unreachability of a node is determined based on limits on other resources. In our context this would correspond to deciding if a failure scenario cannot be triggered. However, this is difficult to decide without actually visiting the arcs of the scenario, since triggering a scenario does not directly depend on other resources but on the topology of the graph. Therefore the unreachability concept cannot readily be used in our case.

A very successful labeling algorithm by Righini and Salani [27] showed how a significant speedup can be gained by using a bi-directional approach. That is, based on a monotone resource (e.g., the number of nodes on the path) a breaking point is chosen (e.g., when half the nodes have been visited) and the labeling algorithm is run from both sides. By splicing paths starting at the origin node o with a reverse path coming from the destination node d one can construct a full path. For this method to work all extension functions must be reversible which unfortunately is not the case for our objective function. Boland et al. [5] and Righini and Salani [28] independently proposed to relax the state-space of the labeling algorithm such that only a subset of resources are considered to begin with. Any violated resource is then added iteratively until a feasible path has been found. By construction of the graph and the definition of the objective function used in our reformulation, it is doubtful that this approach would perform satisfactory since relaxing resources would yield zero weight arcs in the associated backup path, making it necessary to add resources until all feasible backup paths are covered.

In a labeling algorithm the labels represent partial paths that are extended (using the extension functions) in all feasible directions from the origin node o. Each label L (a vector with $R+1$ components) stores the cost of the partial path $T_{cost}(L)$ and the current value $T_r(L)$ of each resource $r \in R$. To avoid enumerating all feasible paths in G', only Pareto-optimal labels (i.e., labels that are not proved to be dominated by other labels) are kept during the execution of the algorithm. When using non-decreasing extension functions (which is the case for the reformulation of QCDPP), the label dominance criterion can be stated as follows.

Proposition 1 ([9]) *Let L and L' be two labels representing partial paths ending at the same node. Label L dominates label L' (which can be discarded) if*

$$\begin{aligned} &T_{cost}(L) \leq T_{cost}(L') \\ &T_r(L) \leq T_r(L') && \forall r \in R. \end{aligned}$$

When equality holds for all label components, one of the two labels must be kept. Figure 5 summarizes the concept of a labeling algorithm. The initial state is represented by the label L_o at the starting node. This label is enqueued on a priority queue Q that keeps track of all unprocessed labels. The algorithm runs until all labels have been processed. In each iteration the next label L from Q is dequeued. The set of nodes (FEASIBLE EXTENSION(L)) that

```
Initialize label $L_o$
ENQUEUE($Q, L_o$)
**while** $Q$ is not empty
    $L :=$ DEQUEUE($Q$)
    **for** each node $i \in$ FEASIBLE EXTENSION($L$)
        $L_i :=$ EXTEND LABEL($L, i$)
        **if** $i = d$
            **then** ENQUEUE($S, L_i$)
            **else** ENQUEUE($Q, L_i$)
    REMOVE DOMINATED($Q$)
**return** $S$
```

Figure 5: Pseudo-code for labeling algorithm.

are feasible extensions of the partial path represented by L, with regard to connectivity and resource limits, is determined. L is extended to these nodes using the resource extension functions (implemented in EXTEND LABEL(L, i)) to create the new label L_i for node i. If the extended label L_i is extended to the end node d it is stored as a solution in the queue S otherwise L_i is enqueued on Q for future processing. Last Q is cleaned for dominated labels so only Pareto-optimal labels remain.

Next, we consider the transformation of the QCDPP stated as (4)-(11) into a SPPRC Recall the graph $G = (V, A)$ for the QCDPP where a minimum cost primary and backup path pair must be found from $o_k \in V$ to $d_k \in V$ over all $k \in K$. Let $V' = \{i' : i \in V\}$ be a copy of all nodes in V and let $A' = \{(j', i') : (i, j) \in A, i', j' \in V'\}$ be a reversed version of all arcs in A connecting the nodes in V', and let $A''_k = \{(d_k, d'_k) : d_k \in A, d'_k \in A'\}$ be the arc connecting the two node and arc sets for demand pair k. The transformed graph for the kth demand pair is then $G'_k = (V \cup V', A \cup A' \cup A''_k)$ where a primary path will be sought in the first part of the graph with nodes V, then by the arc (d_k, d'_k) the search is switched to the other part of the graph consisting of the nodes V' where a reverse backup path is found. G'_k is illustrated on Figure 6. For each failure situation $s \in S$ it must be ensured that no arcs from F_s is used on the backup path if any of the arcs in F_s was used on the primary path. A binary resource is added for each failure situation $s \in S$. Hence, the set of resources have size $|S|$. Let a label L consist of $1 + |S|$ components, $T_{cost}(L)$ to store the cost of the path and $T_s(L)$ for $s \in S$ to store the bit value of the failure situation resources. $T_s(L)$ will be set to one if the failure scenario s is triggered on the primary path, and resource limits are enforced on the arcs when extending labels. The upper bound for resource $s \in S$ when extending a label on arc a' are given as 0 for $a' \in A' \wedge a \in F_s$ and 1 otherwise. That is, a label L cannot be extended on arc $(i', j') \in A'$ with $(j, i) \in F_s$ for $s \in S$ on the backup path if arc $a \in F_s$ is used on the primary path, i.e., the resource value $T_s(L) = 1$ and the upper bound for s on (i', j') is 0. Hence, in Figure 5 the end node of a is not in the set FEASIBLE EXTENSION(L). Recall that the cost of the backup path depends on the arcs used on the primary path and that $\beta^s_a \geq 0$ and $\alpha_k \leq 0$. The extension along an arc a of a label L (implemented in EXTEND

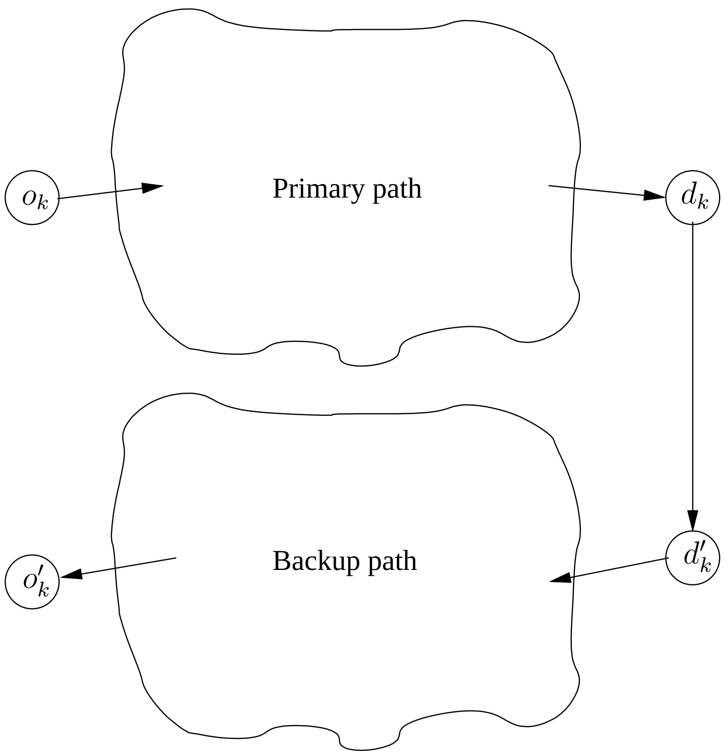


Figure 6: The transformed graph for the kth demand pair. The backup path part of the graph is a reversion of the primary path part, i.e., the path found is a forward directed primary path and a reversed backup path.

LABEL(L, i)) proceeds as follows to create a new label L':

$$T_{cost}(L') = T_{cost}(L) + \begin{cases} \sum_{s\in S} \beta_a^s & a \in A \\ \sum_{s\in S: T_s(L)=1} \beta_{(j,i)}^s & a = (i', j') \in A' \\ -\alpha_k & a \in A''_k \end{cases}$$

$$T_s(L') = \begin{cases} 1 & a \in F_s \\ \max\{T_s(L), 0\} & \text{otherwise} \end{cases} \qquad s \in S$$

Both extension functions are non-decreasing, hence the dominance criterion of Proposition 1 can be applied in the labeling algorithm. For the kth pricing problem; a path represented by label L ending in o'_k have the cost:

$$c_{reduced}^k = -\alpha_k + \overbrace{\sum_{a\in A(L)} \sum_{s\in S} \beta_a^s}^{primary\ path\ cost} + \overbrace{\sum_{a=(i',j')\in A'(L)} \sum_{s\in S: T_s(L)=1} \beta_{(j,i)}^s}^{backup\ path\ cost} \tag{12}$$

where $A(L)$ and $A'(L)$ are the set of arcs used in A and A' respectively. Minimizing expression (12) is equivalent to the objective function stated in (4) and the path found by the labeling algorithm can trivially be split into a primary and a backup path.

```
Initialize $\alpha_k$ , $\beta_a^s$
$k = 1$
**do**
    $k' := k$
    **do**
        SOLVE QCDPP($k$, $\alpha_k$ , $\beta_a^s$)
        $k := k + 1$
    **while** $c_{reduced}^{p,k} \geq 0$ **and** $k' \neq k$
    Update set of path pairs
    SOLVE FIPP with new set of path pairs
    Update $\alpha_k$ , $\beta_a^s$
**while** $k' \neq k$
```

Figure 7: Column Generation algorithm.

3.5 Column Generation Algorithm

Given the LP model in Section 3 we can now apply column generation to solve the model, where the subproblem described in Section 3.2 is either solved using a MIP solver or the labeling algorithm described in Section 3.4. Below we briefly describe the column generation algorithm (Figure 7).

In the column generation algorithm in Figure 7 we first initialize α_k and β_a^s with artificial values: $\alpha_k = \sum_{a \in A} c_a$ and $\beta_a^s = \frac{c_a}{|S|}$ (where S is the set of failure situations). This means that it is always profitable to include a path pair of primary and backup paths for each demand k. After entering the main loop, promising path pairs are found based on the current values of α_k and β_a^s. The resulting paths are then added to the set of path pairs and the master problem is solved with the new set of path pairs. This process continues until no negative reduced-cost path pair for any demand can be found.

4 Results

In this section the efficiency of the FIPP protection method is tested on 8 different networks. Basic network data for the 8 networks is given in Table 2. We have chosen to use the simple demand matrix $D^{kl} = 1$ for each pair of nodes.

In Table 3 and Table 4 we compare the computation times when the QCDPP subproblem is solved using the SPPRC labeling algorithm and a standard MIP solver, respectively.

It can be seen from Table 3 and Table 4 that the SPPRC labeling algorithm is significantly faster on all tested networks. Furthermore, two of the networks, Norway and Ta1, cannot be solved using the MIP solver due to excessive memory consumption.

Given the column generation algorithm, we are now able to calculate the optimal protection capacity required for *relaxed* FIPP protection (Table 5). We find the results in Table 5 interesting because it shows how efficient the relaxed FIPP method is. The FIPP method use at most 8% extra network capacity compared to the theoretical lower bound achieved using Complete Rerouting [30] and on average only 4% extra network capacity. We acknowledge that this is only part of the story and that the moment the demands are required to be integer,

Network	Nodes	Edges	Avg. Node Degree	No. Demands
Cost239 [3]	11	26	4.73	55
Europe	13	21	3.23	78
Newyork [29]	16	49	6.12	120
Ta1 [29]	24	51	4.25	276
FranceSND [29]	25	45	3.60	300
Norway [29]	27	51	3.77	351
USA [10]	28	45	3.21	378
Cost266 [29]	37	57	3.08	666

Table 2: Tested networks and their characteristics.

Network	Rows	Columns				Iter	Time		
		Initial	Final	PerIt	PerDem		Total	CG	CGPct
Cost239 [3]	705	81	1451	42.81	0.78	32	11	1	4.56
Europe [29]	498	99	470	46.38	0.59	8	1	1	36.36
Newyork [29]	2472	169	5292	47.44	0.40	108	2438	1875	76.94
Ta1 [29]	2826	327	4013	43.88	0.16	84	17612	17385	98.71
FranceSND [29]	2280	345	2944	57.76	0.19	45	235	191	81.29
Norway [29]	2901	402	3704	58.96	0.17	56	1177	967	82.22
USA [10]	2358	423	3076	60.30	0.16	44	156	77	49.65
Cost266 [29]	3858	723	6516	62.29	0.09	93	2050	1051	51.29

Table 3: SPPRC labeling algorithm results. Rows: Number of rows in LP. Initial: Initial number of master problem columns. Final: Final number of master problem columns. PerIt: Number of columns added per iteration. PerDem: Number of columns added per iteration per demand. Iter: Number of column generation iterations. Total: Total running time in seconds. CG: Total column generation running time in seconds. CGPct: Column generation (label) solve time as percentage of total time.

i.e., that for each demand the entire communication flow is routed on the same primary path and the same backup path, the ROBB is going to increase.

5 Future Research

The mathematical model we on which we base our results is by choice constructed to be as simple as possible. A number of additional model features can be incorporated into the model and some of these may certainly change the above conclusions. In this section we will briefly describe the two model refinements which we regard as the most important.

Firstly, in the current model we consider the demands as a volume of communication ρ_k to be established between two nodes in the network. In the fractional FIPP problem this volume may be divided between a number of path pairs and this is probably not desirable for the communication customers. Instead, each *customer* should be offered one path pair with a certain volume of traffic — corresponding to the original FIPP problem. For the model

Network	Rows	Columns				Iter	Time		
		Initial	Final	PerIt	PerDem		Total	CG	CGPct
Cost239 [3]	705	81	677	1.00	0.02	597	154	145	93.77
Europe [29]	498	99	307	1.00	0.01	209	31	31	98.36
Newyork [29]	2472	169	2328	1.00	0.01	2160	6491	5943	91.56
Ta1 [29]	2826	-	-	-	-	-	-	-	-
FranceSND [29]	2280	345	1408	1.00	0.00	1064	9434	9356	99.17
Norway [29]	2901	-	-	-	-	-	-	-	-
USA [10]	2358	423	1532	1.00	0.00	1110	2406	2304	95.77
Cost266 [29]	3858	-	-	-	-	-	-	-	-

Table 4: MIP results. Rows: Number of rows in LP. Initial: Initial number of master problem columns. Final: Final number of master problem columns. PerIt: Number of columns added per iteration. PerDem: Number of columns added per iteration per demand. Iter: Number of column generation iterations. Total: Total running time in seconds. CG: Total column generation running time in seconds. CGPct: Column generation (MIP) solve time as percentage of total time.

Network	NP capacity	CR RROB	FIPP RROB	Difference
Cost239	86	0.13	0.19	0.06
Europe	158	0.57	0.65	0.08
Newyork	412	0.19	0.24	0.05
Ta1	733	0.76	0.78	0.02
FranceSND	9825	0.66	0.67	0.01
Norway	61	0.59	0.61	0.02
USA	1273	0.50	0.55	0.05
Cost266	14587	0.62	0.64	0.02
Avg.		0.50	0.54	0.04

Table 5: FIPP protection method comparison. NP capacity: Non-Protected required network capacity. CR RROB: Complete Rerouting [30] required network capacity relative to NP capacity. FIPP RROB: FIPP required network capacity relative to NP capacity. Difference: Absolute difference between RROB for CR and FIPP.

presented in Section 3, this results in more variables, and furthermore, these variables have to be *binary* variables. Hence, to solve this model to optimality, a branch-and-price optimization algorithm is necessary.

Secondly, in the current model there is no bound on the capacity θ_a of an arc $a \in A$. In real-life applications, capacities are acquired in *modular* amounts and economies of scale can be modeled. Modular capacities can be included into the model by changing the right hand side of constraint (3) to a sum of integer variables, as shown in the modified constraint (13) below:

$$\sum_{k\in K}\sum_{\pi\in P_k(a)} \lambda^k_\pi + \sum_{k\in K}\sum_{\pi\in P_k(a,s)} \lambda^k_\pi \;\leq\; \sum_m C_m \cdot \theta_{a,m} \quad \forall\, s \in S,\; a \in A \setminus F_s$$

Here the capacity variables $\theta_{a,m} \in Z^+$ correspond to different types of connections, each possessing a capacity C_m. The objective function is then modified to include different prices for each type of technology. The price pr. capacity unit reflect the economies of scale.

6 Conclusion

In this paper we presented an LP model for the fractional Failure Independent Path Protection (FIPP) optimization problem. The LP model was solved using column generation. We analyzed the subproblem, proved it to be strongly $\mathcal{NP}$-hard and devised a labeling algorithm for solving the subproblem more efficiently. Finally, we evaluated the capacity efficiency of the FIPP method on a number of network instances. The results indicate that the FIPP method appears to be a very efficient protection method — on average only requiring 4% more network capacity than complete rerouting, the absolute lower bound for single link failure protection.

References

[1] R. K. Ahuja, J. B. Orlin, and T. L. Magnanti. *Network Flows: Theory, Algorithms, and Applications*. Prentice Hall, 1993.

[2] J. Anderson, B.T. Doshi, S. Dravida, and P. Harshavardhana. Fast restoration of atm networks. *IEEE Journal on Selected Areas in Communications*, 12(1):128–138, 1994. ISSN 07338716.

[3] P. Batchelor, B. Daino, P. Heinzmann, D. R. Hjelme, P. Leuthold, R. Inkret, G. De Marchis, H. A. Jager, F. Matera, M. Joindot, B. Mikac, A. Kuchar, H.-P. Nolting, E. Coquil, J. Spath, F. Tillerot, B. Caenegem, N. Wauters, and C. Weinert. Study on the implementation of optical transparent transport networks in the european environment-results of the research project COST 239. *Photonic Network Communication*, 2(1):15–32, 2000. ISSN 1387974X. doi: 10.1023/A:1010050906938.

[4] R Bhandari. *Survivable Networks - Algorithms for Diverse Routing*. Kluwer, 1999.

[5] N. Boland, J. Dethridge, and I. Dumitrescu. Accelerated label setting algorithms for the elementary resource constrained shortest path problem. *Operation Research Letters*, 34 (1):58–68, 2006. doi: 10.1016/j.orl.2004.11.011.

[6] E. Calle, J. L. Marzo, and A. Urra. Protection performance components in MPLS networks. *Computer Communications*, 27(12):1220–1228, 2004. ISSN 01403664. doi: 10.1016/j.comcom.2004.02.025.

[7] W.M. Carlyle, J.O. Royset, and R.K. Wood. Lagrangian relaxation and enumeration for solving constrained shortest-path problems. *Networks*, 51(3):155–170, 2008. doi: 10.1002/net.20212.

[8] G. Desaulniers, J. Desroisers, Y. Dumas, S. Marc, B. Rioux, M. M. Solomon, and F. Soumis. Crew pairing at Air France. *European Journal of Operations Research*, 97:245 – 259, 1997. doi: 10.1016/S0377-2217(96)00195-6.

[9] G. Desaulniers, J. Desrosiers, J. Ioachim, I. M. Solomon, F. Soumis, and D. Villeneuve. A unified framework for deterministic time constrained vehicle routing and crew scheduling problems. In T. G. Crainic and G. Laporte, editors, *Fleet Management and Logistics*, pages 57–93. Kluwer, 1998.

[10] J. Doucette, D. He, W. D. Grover, and O. Yang. Algorithmic approaches for efficient enumeration of candidate p-cycles and capacitated p-cycle network design. In *Proceedings. Fourth International Workshop on Design of Reliable Communication Networks, 2003. (DRCN 2003)*, pages 212–220. IEEE, 2003. ISBN 0780381181. doi: 10.1109/DRCN.2003.1275359.

[11] M. Dror. Note on the complexity of the shortest path models for column generation in VRPTW. *Operations Research*, 42(5):977–979, 1994. doi: 10.1287/opre.42.5.977.

[12] I. Dumitrescu and N. Boland. Improved preprocessing, labeling and scaling algorithms for the weight-constrained shortest path problem. *Networks*, 42(3):135–153, 2003. doi: 10.1002/net.10090.

[13] Dominique Feillet, Pierre Dejaxa, Michel Gendreaua, and Cyrille Gueguen. An exact algorithm for the elementary shortest path problem with resource constraints: Application to some vehicle routing problems. *Networks*, 44(3):216–229, 2004. ISSN 0028-3045. doi: 10.1002/net.v44:3.

[14] M. R. Garey and D. S. Johnson. *Computers and Intractability. A guide to the theory of NP-completeness.* Freeman, 1979.

[15] W. D. Grover. *Mesh-Based Survivable Networks.* Prentice Hall PTR, 2004.

[16] D. Haskin and R Krishnan. A method for setting and alternative label switch path to handle fast reroute. Technical report, Internet Engineering Task Force, 2002.

[17] Jian Qiang Hu. Diverse routing in optical mesh networks. *IEEE Transactions on Communications*, 51(3):489–494, 2003. ISSN 00906778. doi: 10.1109/TCOMM.2003.809779.

[18] S. Irnich. Resource extension functions: Properties, inversion, and generalization to segments. *OR Spectrum*, 30(1):113–148, 2008. doi: 10.1007/s00291-007-0083-6.

[19] S. Irnich and G. Desaulniers. Shortest path problems with resource constraints. In G. Desaulniers, Jacques Desrosiers, and M.M. Solomon, editors, *Column Generation*, chapter 2, pages 33–65. Springer, 2005. doi: 10.1007/0-387-25486-2_2.

[20] S. Irnich and D. Villeneuve. The shortest path problem with resource constraints and k-cycle elimination for $k \geq 3$. *INFORMS Journal on Computing*, 18:391–406, 2006. doi: 10.1287/ijoc.1040.0117.

[21] M. 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. doi: 10.1287/opre.1070.0449.

[22] P. Laborczi and T. Cinkler. Efficient algorithms for physically-disjoint routing in survivable GMPLS/ASTN networks. In *11th International Telecommunications Network*

Strategy and Planning Symposium. (NETWORKS 2004), pages 185–192. IEEE, 2004. ISBN 3800728400. doi: 10.1109/NETWKS.2004.1341839.

[23] P. Laborczi, J. Tapolcai, P.-H. Ho, T. Cinkler, A. Recski, and H. T. Mouftah. Algorithms for asymmetrically weighted pair of disjoint paths in survivable networks. In T. Cinkler, editor, *Third International Workshop on Design of Reliable Communication Networks, 2001. (DRCN 2001)*, pages 220–227. BME - Budapest Univ. Technol. & Econ, 2001.

[24] Jean François Maurras and Sonia Vanier. Network synthesis under survivability constraints. *4OR*, 2(1):53–67, 2004. doi: 10.1007/s10288-003-0025-3.

[25] S. Orlowski. Local and global restoration of node and link failures in telecommunication networks. Master's thesis, Technische Universität Berlin, February 2003. URL `http://www.zib.de/orlowski/Orlowski2003.pdf.gz`.

[26] M. Pioro and D. Medhi. *Routing, Flow, and Capacity Design in Communication and Computer Networks.* Elsevier, 2004.

[27] G. Righini and M. Salani. Symmetry helps: bounded bi-directional dynamic programming for the elementary shortest path problem with resource constraints. *Discrete Optimization*, 3(3):255–273, 2006. doi: 10.1016/j.disopt.2006.05.007.

[28] Giovanni Righini and Matteo Salani. New dynamic programming algorithms for the resource constrained elementary shortest path problem. *Networks*, 51(3):155–170, 2008. ISSN 0028-3045. doi: 10.1002/net.v51:3.

[29] SNDlib. SNDlib 1.0–Survivable network design data library, 2005. URL `http://sndlib.zib.de`.

[30] T. Stidsen and P. Kjærulff. Complete rerouting protection. *Journal of Optical Networking*, 5(6):481–492, 2006. doi: 10.1364/JON.5.000481.

[31] T. Stidsen and T. Thomadsen. Joint routing and protection using p-cycles. Technical report, Technical University of Denmark, 2005. URL `http://www2.imm.dtu.dk/pubdb/p.php?3939`.

[32] T. Stidsen, M. Kiese, B. Petersen, S. Spoorendonk, and M. Zachariasen. Network capacity planning with shared protection. Work in progress, 2008.

[33] J.W. Suurballe. Disjoint paths in a network. *Networks*, 4(2):125–145, 1974. doi: 10.1002/net.3230040204.

[34] G. Swallow and L. Andersson. MPLS working group. URL `http://www.ietf.org/html.charters/mpls-charter.html`.

Chapter 11

Liner Shipping Revenue Management with Repositioning of Empty Containers

Berit Løfstedt
DIKU Department of Computer Science, University of Copenhagen

David Pisinger
DIKU Department of Computer Science, University of Copenhagen

Simon Spoorendonk
DIKU Department of Computer Science, University of Copenhagen

Abstract

Repositioning of empty containers pose a significant cost in shipping due to the large difference in export and import between some parts of the world, e.g., North America and Asia. Dejax and Crainic [9] estimate, that movement of empty containers comprise up to 40% of all container movements. This paper presents a revenue management model for a liner shipping company where the repositioning of empty containers is taken into account. The booking model aims at maximizing the profit of transported cargo in a network, subject to the cost and availability of empty containers. The model is an augmented multi-commodity flow problem with additional inter-balancing constraints to control repositioning of empty containers. An arc flow formulation is Dantzig-Wolfe decomposed to a path flow formulation, where the LP relaxation is solved with a delayed column generation algorithm. A feasible IP solution is hereafter found by rounding down the LP solution and adjusting flow balance constraints with leased containers. Computational results are reported for eight instances based on real-life shipping networks. Solving the path flow model with a simple column generation algorithm outperforms solving the arc flow model with the *CPLEX* barrier solver even for very small instances. The proposed algorithm is able to solve instances with 234 ports, and 293 vessels for 9 time periods in 34 minutes. The integer solutions found by rounding are computed in less than 5 seconds and the gap is within 0.01% of the LP upper bound, which is well below the uncertainty of the input data. The solved instances are quite large compared to computational results in the reviewed literature on models for empty container repositioning.

1 Introduction

This paper presents a revenue management model for strategic planning within a *liner shipping* company. A revenue management model is a strategic tool that given a schedule and a fleet over time decides which orders are profitable to transport with the planed capacity. A mathematical model is presented for maximizing the profit of cargo transportation while considering the possible cost of repositioning empty containers. The model is denoted revenue management with repositioning of empty containers (RMREC). Empty containers tend to accumulate at import intensive regions due to a significant imbalance in world trade. Therefore, repositioning empty containers to export intensive regions impose a large cost on liner shippers. RMREC incorporates the potential repositioning cost such that the profit of an order takes into account the derived demand for empty containers. As opposed to most models RMREC permits load rejection, since we believe, that an unprofitable order may be rejected due to capacity constraints in the liner shipping network.

A *liner shipping* company is a shipping operator with a public itinerary and schedule visiting certain ports at a given service frequency. The objective is to maximize profit for freighting optional cargo between ports. A liner shipping company differs from *industrial shipping*, where the objective is to minimize the transportation cost of delivering all cargo, and from *tramp shipping*, where the objective is to maximize the profit of optional cargo while delivering obligated cargo. Current practice with regards to empty containers is to have conservative stock policies and empty deadweight on vessels to ensure sufficient availability of empty containers. With RMREC, we hope to reduce deadweight on board vessels and to minimize stock of empty containers. Furthermore, sensitivity analysis may be applied to identify bottlenecks in the shipping network and help price commodities according to demand and empty repositioning cost. Lastly, the model may be used to investigate alternative leasing policies for liner shippers.

The strategic booking decision of a liner shipper considering empty container repositioning can be described as a specialized multi-commodity flow problem with inter-balancing constraints to control the flow of empty containers. A commodity in logistic terms is a pair (O, D), where O is the origin and D is the destination of a container demand. The set of commodities is denoted K. The network is represented by a graph $G = (N, A)$, where the node set N represents the ports and the arc set A represent the scheduled itineraries. The capacity associated with each edge is determined by the assignment of vessels to the schedule. The objective is to find a set of feasible paths in the network such that the profit of routing cargo between the (O, D) port pairs is maximized.

The classical formulation of the standard multi-commodity flow problem is the arc flow formulation with $|K||A|$ variables and $|A| + |K||N|$ constraints due to flow conservation at every node. Although the number of variables is polynomially bounded, it will be huge for a global shipping network. In addition, a large constraint set results in poor performance for the simplex method. Dantzig-Wolfe decomposition can be applied to generate a path flow formulation with only $|A|+|K|$ constraints. However, the number of variables in the path flow formulation may exponential. To circumvent this problem we use delayed column generation, as it can be proven that at most $|K| + |A|$ paths carry positive flow [1]. The pricing problem is a shortest path problem, where the cost of a path represents the reduced cost of a path variable.

RMREC is an augmented multi-commodity flow problem where the extra constraints stem from the inter-balancing constraints which ensure repositioning or leasing of empty containers at nodes with a positive net flow. Due to the structure of the dual problem, the arc costs of the pricing problem for RMREC are positive resulting in a polynomially solvable pricing problem. As containers cannot be split, RMREC is an integer multi-commodity flow problem which is $\mathcal{NP}$-hard. A nice property of the path flow formulation is, that flow conservations constraints are implicitly satisfied on a path. Hence, a feasible integer solution can be obtained by rounding down all fractional variables and supplying empty containers through a leasing variable at nodes with violated inter-balancing constraints.

Solving both standard and augmented integer multi-commodity flow problems is a well-studied area within airline management, e.g., crew scheduling [10] and fleet assignment [11, 3] where a branch-and-price algorithm on the path flow formulation is used to find an integer solution. The integer variables of these known problems are mostly binary. However, the integer variables of RMREC are large integer numbers because demands are expressed in containers as the total demand between any two (O, D) port pair. Because the number of containers transported in a global shipping network is huge, rounding down the fractional part of demands may be considered insignificant. This is confirmed by experimental results in this paper. Therefore, we consider the LP relaxation of RMREC and obtain a heuristic integer solution by rounding down the LP solution of the path flow model.

The contribution of this paper is to present an augmented multi-commodity flow formulation of the container transportation problem considering repositioning of empty containers (RMREC). A basic arc flow model of RMREC is decomposed into a path flow model. The LP relaxation of RMREC is solved with a delayed column generation algorithm. Computational results are reported for eight instances based on real life shipping networks. The results show that the delayed column generation algorithm for the path flow model clearly outperforms solving the arc flow model with the *CPLEX* barrier solver. Instances with up to 234 ports and 293 vessels for 9 time periods were solved in less than 34 minutes with the column generation algorithm. The largest instance solved for 12 time periods contains 151 ports and 222 vessels and was solved in less than 75 minutes. It is shown, that high quality integer solutions within 0.01% from the LP upper bound of the path flow formulation can be found by a simple rounding heuristic.

The following section describes related work. Section 3 describes the network representation used throughout this paper. Section 4 presents the arc flow model, and Section 5 presents the decomposed path flow model and the pricing problem used in the delayed column generation algorithm. Section 6 reports our computational results on eight generated test instances. Section 7 provides some concluding remarks and future work of RMREC.

2 Literature Overview

According to Ronen et al. [16] papers on optimization based decision support systems within shipping are scarce. There is an increasing interest in operations research within the area of shipping, but most papers concern scheduling and routing of vessels. Furthermore, most papers are concerned with industrial and tramp shipping. Within the area of liner shipping only a few references are found and they concern deployment of vessels [16]. Christiansen et al. [5] describe models for designing shipping networks for a traditional liner operation as well as for a hub-and-spoke liner network. According to the paper, a booking is accepted if there is space available on a vessel. This may lead to non-optimal decisions since the space may be used more profitably by demands in subsequent ports on the route. However, the issue of empty container availability is not regarded as a component of cargo profitability, although the connection to profit is evident. The paper encourages research into the area of revenue management and booking models, but very little work has been published on this subject as mentioned by [16, 5].

The airline industry holds several similarities with the booking models and flow constraints encountered in the maritime industry. Especially with regard to the relation of the underlying network structure. RMREC was originally inspired by Bartodziej and Derigs [2] who present a revenue management model for Cargo Airlines. The model is a special multi-commodity flow problem which is solved by column generation.

Hane et al. [11] solve an airline fleet assignment problem as a multi-commodity problem, where air-crafts need repositioning and where aggregation of the graph is considered. Desaulniers et al. [10] solve a crew scheduling problem using multi-commodity flows, where the problem of repositioning crew is mentioned, but not thoroughly treated. Bélanger et al. [3] solve a fleet assignment problem with time-windows as a multi-commodity flow problem.

Empty container repositioning can be regarded as empty flow in a network. Empty flows have been studied within all areas of the transportation sector because they represent a significant cost. In a survey on empty flows, Dejax and Crainic [9] estimate that up to 40% of all movements for rail cars and containers are empty. The need for models considering empty and loaded movements simultaneously is emphasized. Crainic et al. [7] present a multi mode multi-commodity location-distribution problem with inter-depot balancing requirements. The model is primarily a location problem deciding the number and locations of inland depots for empty vehicles. However, it also determines the empty flows between depots according to the inter-balancing constraints. The model has been tested on data from a European company, which operated 23 major European ports at the time. A large reduction in the number of inland depots is reported, which along with management of the empty flows represent a 47% annual saving for the company. An international liner shipping company of today will span the globe and service hundreds of ports. Furthermore, container vessels have increased from 5,9% to 9% of the world fleets total deadweight capacity [16] from 1995 to 2001.

Crainic et al. [8] present a dynamic and stochastic model for the empty container allocation problem. The context is an international shipping company with focus on the land operations, i.e., movements between customers and depots. The paper gives a very thorough description of the container trade with regards to the space and time of events along with the complex issue of asymmetric substitution between container-types. A single- and a multi-commodity model with containers as commodities are presented using a time-space network in a rolling horizon manner.

Shen and Khoong [17] present a decision support system for empty container distribution planning for a shipping company at port level. The paper describes a network optimization model and a heuristic to solve the problem but actual implementation and results are not reported.

Cheung and Chen [4] have developed a two stage stochastic network model for the dynamic empty container allocation problem. The model minimizes the total cost of repositioning or leasing containers at deficit ports and is highly related to Crainic et al. [8].

Li et al. [13] present the empty container problem as a non standard inventory model at a port to reduce holding of redundant empty containers. The paper also explores finite and infinite horizon methods.

The above papers all assume no load rejection as competition in the trade is fierce. Hence simultaneous optimization of loaded movements is irrelevant. Shintani et al. [18] are the first to discuss the possibility of rejecting unprofitable cargo, but within the context of designing container ship networks with regard to empty container repositioning. The model is a knapsack formulation choosing the ports to call, with an underlying model choosing the optimal port calling sequence under the assumption that all demand is satisfied in ports called. Repositioning of empty containers is only allowed in case of excess capacity making the cost of the repositioning negligible. The incurred cost is the penalty cost of storing or leasing empty containers. Using a genetic algorithm, results are reported for a model with 20 ports in Asia.

None of the above papers solve problems of a size corresponding to present shipping networks. Cheung and Chen [4] perform experiments for 3 randomly generated networks, where the largest instance has 10 ports, 6 voyages/vessels and 42 time periods. Shintani et al. [18] solve test instances for 5-8 ports out of 20 potential ports. The number of voyages/vessels is not declared. The number of time periods is 52.

3 Network Representation

RMREC may be modeled as a multi-commodity flow problem with inter-balancing constraints having as objective to maximize the profit of the demanded flow in a capacitated network.

The network consists of a set of unique ports P connected by the services offered by the liner shipping company. All services are cyclic. Since a service may take months to rotate, the network must be modeled over time. Let T be the set of time periods.

A *time-space* network is created as a graph $G = (N, A)$, where $N = \{p^t \mid p \in P, t \in T\}$ is the set of nodes. Let $A = A_G \bigcup A_R$ and let $A_G = \{(p^t, p^{t+1}) \mid p^t, p^{t+1} \in N\}$ be the set of ground arcs representing the stock at a port between two subsequent time periods. Let $A_R = \{(p^t, q^{t'}) \mid p^t, q^{t'} \in N,\ t \leq t',\ p \neq q\}$ be the set of travel arcs representing a voyage on a vessel between two ports $p, q \in P$ departing at time $t \in T$ and arriving at time $t' \in T$. The capacity of an arc $a \in A_R$ is given by the capacity of the vessels on the service at that specific time. An illustration of the *time-space* network may be seen in Figure 1 where time spans the x-axis and space, i.e., the geographical location of ports spans the y-axis.

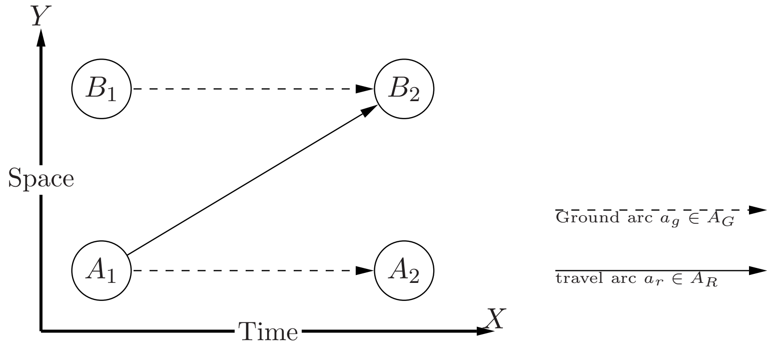


Figure 1: Example of time-space network - dotted arcs are ground arcs.

Each origin O_k and destination D_K of commodity k is given by a port $p \in P$ and time $t \in T$. For instance the tuple $(O_k, D_k, d_k, s_k) = (\texttt{BUA}_{08}, \texttt{BRV}_{09}, 2000, 220)$, means that the origin O_k is Buenos Aires (BUA) in month 08, the destination D_k is Bremerhaven (BRV) in month 09, the demand is 2000 TEU (twenty foot container type) and the sales price s_k is 220. The origin and destination time may be thought of as a *time window* for the delivery of the commodity.

In RMREC no consideration is taken for substitution of container types and it is assumed that all demands are accounted for with the smallest container type, enabling us to scale all larger container types to the smallest container type.

The granularity of time may be defined in two ways resulting in networks with different properties. T may represent the schedule directly if all events for a set of voyages is defined by a point in time, i.e., the voyage of a vessel gives rise to a totally ordered set of time units. This is illustrated in Figure 2. The resulting graph is directed, acyclic and can be topologically sorted. Hence, finding a shortest path can be done in $O(N + A)$ time.

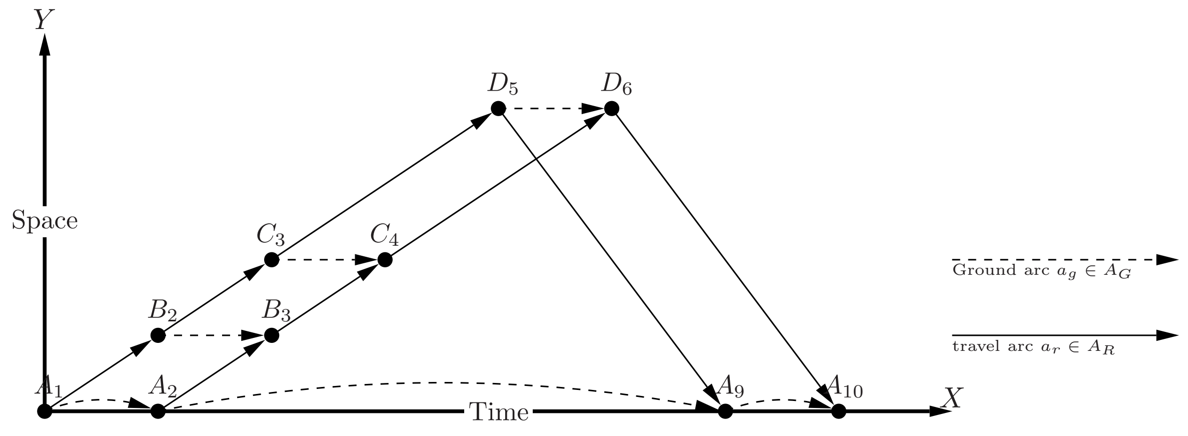


Figure 2: Time-space network with the schedule as time units. Service 1: $A \rightarrow B \rightarrow C \rightarrow D \rightarrow A$, vessel $v_{1,1}$ departing A at time $t = 1$ vessel $v_{1,2}$ departing A at time $t = 2$.

The network will be very sparse, but may increase the number of path variables for a commodity, if the time windows of the commodities are not tight. Each vessel represents an itinerary and has its own nodes and arcs in the network. However, several vessels offer the same service with regard to visited ports on a voyage. This means, that there may be several daily departures from a port heading to the same destination. Figure 2 illustrates the issue: a commodity from port $A1$ to port $C4$ can be serviced by four possible paths with an identical geographical port sequence ($A \rightarrow B \rightarrow C$). This can be avoided by tight time windows, but finding a feasible itinerary for a commodity may then become a problem. RMREC is intended for long-term planning preferably spanning 6 months or more. If a very time detailed network is applied on a problem with hundreds of ports with several daily departures, the network will be extremely large, making it impractical to solve. Hence, it seems reasonable to search for a more compact representation.

The shipping network may also be described by representing each port in a given time interval and aggregating all arrival and departure events to/from this port within the specified period. This definition of time gives a less detailed, but more compact graph, where the paths of the individual vessels are aggregated into a path per rotation for the given time period. This will lead to fewer variables as well as fewer constraints in the path flow formulation.

Compared to the graph representing the detailed time-space network, the compact graph is not acyclic. In the compact graph a service can rotate within a single time period (see

Figure 3), and the services may contain cycles within them (see Figure 4). If a service takes more than one time period to rotate, then an itinerary is a directed path from port h_t to port h_{t+i}, where i represents the number of time periods it takes the vessel to rotate (see Figure 5). However, if the time unit is set to one month, we believe most services will rotate in a single time period. RMREC is intended for long term planning and a detailed schedule is not needed at this point. Hence, the compact aggregated time definition is preferred.

4 Arc Flow Formulation

Although the integer version of RMREC is $\mathcal{NP}$-hard, the LP relaxation may be solved in polynomial time. The challenge lies in the expected size of a liner shipping network and the number of periods in the planning horizon, which result in very large LPs.

In the arc flow formulation we have a set of commodities K, defined on a graph with nodes N and arcs A. The unit cost of arc (i,j) for commodity k is denoted c^k_{ij}. The non-negative integer variable x^k_{ij} is the flow on arc (i,j) of commodity k. The capacity of arc (i,j) is u_{ij} and d_k is the demand for commodity k. Finally, O_k is the origin of commodity k and D_k is the destination. A commodity in the network is defined as the tuple (O_k, D_k, d_k, s_k) which represents a demand of d_k from node $O_k = p^t$ to node $D_k = q^{t'}$ with a sales price per unit of s_k.

The standard multi-commodity flow model does not consider the supply of empty containers. Inter-balancing constraints are applied to every node to account for availability of empty containers. The constraints require, that the amount of containers arriving at a port must be at least the amount of containers leaving the port for all commodities. Therefore, we get a demand for empty containers depending on the actual allocation of loaded commodities. The inter-balancing constraints also introduce a new set of variables representing leased containers at a node. The cost of leasing is modeled in the objective. Let c^i_l be the cost of leasing a container at port i, while l_i is the leasing variable at port i. If the demand for empty containers are seen as commodities, a set of empty commodities with no revenue and a derived demand is needed. In the arc flow formulation a set of empty commodities must be defined consisting of every possible (O,D) pair with no upper bound on the flow and with no sales price. The set would be huge and the constraints redundant, as they do not impose bounds on the flow. The only purpose of the constraints would be to define origin and destination of empty flows. Since flow conservation is redundant and defining origin and destination of empty flows is already done by the inter-balancing constraints, an empty super commodity without flow conservation constraints may be defined in the arc flow model. The empty super commodity is defined for all arcs in the network, allowing empty flows to start and end anywhere needed in the network. The empty super commodity has no flow conservation constraints and appear in the objective with a cost and in the bundled capacity and inter-balancing constraints. For

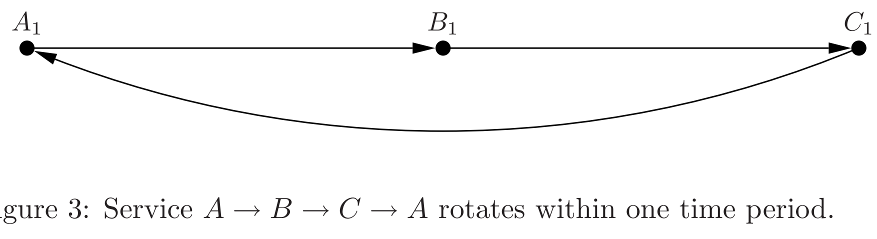


Figure 3: Service $A \to B \to C \to A$ rotates within one time period.

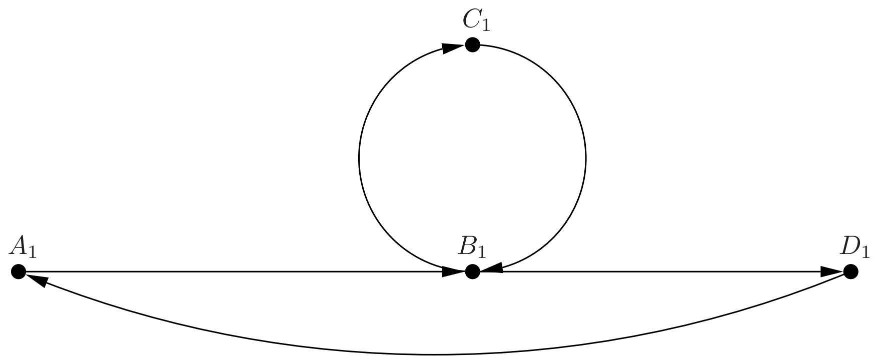


Figure 4: Service $A \rightarrow B \rightarrow C \rightarrow B \rightarrow D \rightarrow A$ with an internal cycle.

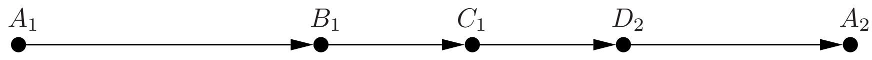


Figure 5: Service $A \rightarrow B \rightarrow C \rightarrow D \rightarrow A$ rotates in two time periods.

convenience the commodity set is split into the loaded commodities and the empty super commodity: Let K_F be the set of commodities with a cargo, a sales price and a demand. Let k_e for all arcs in A be the empty super commodity with no cargo, no sales price and a demand implicit derived from K_F. Finally, let $K = K_F \cup K_e$. Load rejection and capacity constraints mean that demand may not be met for all demand pairs. The net flow of commodity k at the origin node reveals the quantity of demand transported in the network and hence the revenue of commodity k. The cost of transport, leasing and the empty super commodity must be subtracted. RMREC with a profit maximizing objective, an empty super commodity, leasing variables, load rejection and inter-balancing constraints is stated as:

$$\max \sum_{k \in K_F} \sum_{j \in N} s^k (x^k_{O_k j} - x^k_{j O_k}) - \sum_{k \in K} \sum_{(ij) \in A} c^k_{ij} x^k_{ij} - \sum_{i \in N} c^i_l l^i \tag{1}$$

$$\begin{aligned}
\text{s.t.} \sum_{j \in N} x^k_{ij} - \sum_{j \in N} x^k_{ji} &\leq d_k & i = O_K \quad & k \in K_F & (2)\\
\sum_{j \in N} x^k_{ji} - \sum_{j \in N} x^k_{ij} &\leq d_k & i = D_k \quad & k \in K_F & (3)\\
\sum_{j \in N} x^k_{ij} - \sum_{j \in N} x^k_{ji} &= 0 & i \in N \setminus \{O_k, D_K\} \quad & k \in K_F & (4)\\
\sum_{k \in K} x^k_{ij} &\leq u_{ij} & & (i,j) \in A & (5)\\
\sum_{k \in K} \sum_{j \in N} x^k_{ij} - \sum_{k \in K} \sum_{j \in N} x^k_{ji} - l^i &\leq 0 & & i \in N & (6)\\
x^k_{ij} \in \mathbb{Z}_+ & & & k \in K, (i,j) \in A & (7)\\
l_i \in \mathbb{Z}_+ & & & i \in N & (8)
\end{aligned}$$

The objective (1) is to maximize the profit of the demanded flow of all commodities in K on the arcs. Constraints (2)-(4) are the flow conservation constraints which ensure flow of a demand from origin to destination. Furthermore, the flow is bounded by the demanded quantity d_k. Constraints (5) are the bundle constraints ensuring that the flow of all commodities do not exceed the capacity of the arcs. Constraints (6) are the inter-balancing constraints which gives a derived demand for the empty super commodity and leased containers. Constraints (7)-(8) ensure non-negative and integral flow and leasing variables. The formulation is polynomial in the input size as the number of variables is $O(|K||A| + |N|)$ and the number of constraints is $O(|N||K| + |A| + |N|)$. Although the problem size is polynomially bounded, models of large networks have a vast number of variables and a substantial number of constraints, which deteriorates the performance of the simplex algorithm [19].

It should be noted that when a container is leased, it remains in the network for the remainder of the period. This corresponds to long-term leasing for the first period and short term leasing in the last periods. The cost of a leasing variable should depend on the amount of time periods remaining in T at the node where it is leased. Off-leasing, that is terminating the lease of a container, can be modeled by defining an off-leasing variable and making cost dependent on the net leasing between in- and off-leasing variables at the nodes. This changes the objective function (1) to:

$$\max \sum_{k \in K_F} \sum_{j \in N} s^k (x^k_{O_k j} - x^k_{j O_k}) - \sum_{k \in K} \sum_{(i,j) \in A} c^k_{ij} x^k_{ij} - \sum_{i \in N} c^i_l (l^i_{in} - l^i_{off})$$

and the inter-balancing constraints (6):

$$\sum_{k \in K} \sum_{j \in N} x^k_{ij} - \sum_{k \in K} \sum_{j \in N} x^k_{ji} - l^i_{in} + l^i_{off} \leq 0 \qquad i \in N$$

The above model assumes off-leasing can occur at any port. Off-leasing can be restricted to certain ports by defining off-leasing variables accordingly. When the container is leased the remainder of the optimization period is paid for. When it is off leased the remainder of the optimization period at the off-leasing point is refunded.

Various services are offered by leasing companies, that own half the maritime container fleet worldwide, see [12]. Leasing services vary from one-trip and round-trip leases to short-, medium- and long-term leasing ranging from one month to 42 months, see [20]. For the current RMREC, it is chosen to model leasing on a monthly basis, i.e., one time period, which may range from a single month to the entire time period optimized upon. However, the leasing mode may be altered, if different leasing services is explored.

5 Path Flow Formulation

Solving the LP relaxed RMREC model has several advantagds:

- Although there is a polynomial bound on the number of variables in the arc flow formulation, it is a large polynomial factor. Even though the number of paths in the network *may* be exponential, it depends on how dense the network is. If the network is sparse, like in most liner shipping networks, there will be few path variables for each commodity $k \in K$.

- The network size of an international liner shipping company means that column generation is our only hope to solve the problem in reasonable time. This is true, even for the LP relaxation.

- An LP-solution to the path flow formulation can be transformed to a feasible IP solution to RMREC by rounding, as flow conservation is respected implicitly in the path variables (see figures 6 and 7). This makes it possible to translate the solution directly into itineraries for the demand pairs.

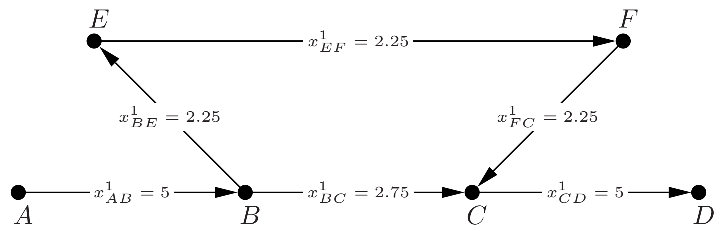


Figure 6: A fractional solution to the arc flow model: Containers may be split at every node.

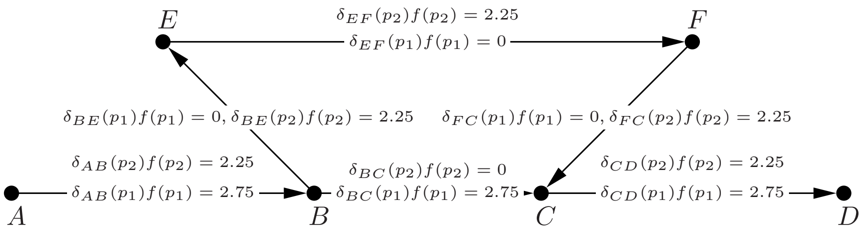

E
$\delta_{EF}(p_2)f(p_2) = 2.25$
$\delta_{EF}(p_1)f(p_1) = 0$
F
$\delta_{BE}(p_1)f(p_1) = 0, \delta_{BE}(p_2)f(p_2) = 2.25$
$\delta_{FC}(p_1)f(p_1) = 0, \delta_{FC}(p_2)f(p_2) = 2.25$
$\delta_{AB}(p_2)f(p_2) = 2.25$
$\delta_{AB}(p_1)f(p_1) = 2.75$
$\delta_{BC}(p_2)f(p_2) = 0$
$\delta_{BC}(p_1)f(p_1) = 2.75$
$\delta_{CD}(p_2)f(p_2) = 2.25$
$\delta_{CD}(p_1)f(p_1) = 2.75$
A
B
C
D

Figure 7: Fractional solution to the path flow model: Containers may only be split at the origin.

The RMREC given as the arc flow model (1)-(8) has block-angular structure with $|K_F|$ subproblems given by the flow conservation constraints for each full commodity. The commodity subproblems are tied together by the bundle constraints, i.e., the arc capacity constraints, and the inter-balancing constraints regarding the supply of empty containers. Using Dantzig-Wolfe decomposition we get a master problem considering paths for all commodities, and a subproblem defining the possible paths for each commodity $k \in K$. Due to the *the flow decomposition theorem* (Chapter 3.5 in [1]), which states that every nonnegative arc flow can be represented as a nonnegative path and cycle flow, this can be formulated such that master variables define the flow of a commodity on a path, and the subproblem find valid paths for that commodity.

Let $\overline{p}$ be a path connecting O_k and D_k and P_k be the set of all paths belonging to commodity k. The flow on path $\overline{p}$ is denoted by the variable $f(\overline{p})$. The binary indicator $\delta_{ij}(\overline{p})$ is one if and only if arc (i,j) is on the path $\overline{p}$. Finally, $c^k_{\overline{p}} = \sum_{(i,j)\in A} \delta_{ij}(\overline{p})c^k_{ij}$ is the cost of path $\overline{p}$ for commodity k. The master problem is:

$$\max \quad \sum_{k\in K_F} \sum_{\overline{p}\in P_k} (s^k - c^k_{\overline{p}}) f(\overline{p}) - \sum_{(i,j)\in A} c^{K_E}_{ij} x^{K_E}_{ij} - \sum_{i\in N} c^i_l l^i \tag{9}$$

$$\text{subject to} \quad \sum_{k \in K_F} \sum_{\overline{p} \in P_k} \delta_{ij}(\overline{p}) f(\overline{p}) + x_{ij}^{K_E} \leq u_{ij} \qquad (i,j) \in A \qquad (10)$$

$$\sum_{\overline{p} \in P_k} f(\overline{p}) \leq d_k \qquad k \in K_F \qquad (11)$$

$$\sum_{k \in K_F} \sum_{\overline{p} \in P_k} \sum_{j \in N} (\delta_{ij}(\overline{p}) - \delta_{ji}(\overline{p})) f(\overline{p}) + x_{ij}^{K_E} - x_{ji}^{K_E} - l^i \leq 0 \qquad i \in N \qquad (12)$$

$$f(\overline{p}) \in \mathbb{Z}_+ \qquad \overline{p} \in P_k k \in K_F \qquad (13)$$

$$x_{ij}^{K_E} \in \mathbb{Z}_+ \qquad (i,j) \in A \qquad (14)$$

$$l^i \in \mathbb{Z}_+ \qquad i \in N \qquad (15)$$

Where the x_{ij}^k variables are replaced by $x_{ij}^k = \sum_{\overline{p} \in P_k} \delta_{ij}(\overline{p}) f(\overline{p})$ according to the flow decomposition theorem for all $k \in K_F$. The subproblems for commodities $k \in K_F$ are given by the polytopes:

$$\mathcal{P}_k = \left\{ \begin{array}{lcr} \sum_{j \in N} x_{ij} - \sum_{j \in N} x_{ji} & = 1 & i = O_K \\ \sum_{j \in N} x_{ji} - \sum_{j \in N} x_{ij} & = 1 & i = D_k \\ \sum_{j \in N} x_{ij} - \sum_{j \in N} x_{ji} & = 0 & i \in N \setminus \{O_k, D_K\} \\ x_{ij} \in \{0,1\} & & (i,j) \in A \end{array} \right\}$$

The convexity constraints for the individual subproblems (11) bound the flow between the (O_k, D_k) pair from above (a maximal flow of d_k is possible). The extreme points of $\mathcal{P}_k$ model a path between the (O_k, D_k) pair.

The exponential number of variables is handled by considering only a small subset in a restricted master problem. Paths/columns are then generated on the fly using delayed column generation. The dual variables corresponding to the three constraint sets are:

- w_{ij} for each $(i,j) \in A$ corresponding to the bundle constraints (10)
- σ^k corresponding to the convexity constraints (11) for each commodity $k \in K$
- α^i corresponding to the inter-balancing constraints (12) for each port $i \in N$.

The reduced cost $\hat{c}$ of a path $\overline{p} \in P_k$ is then given by the variable $f(\overline{p})$ in subproblem $k \in K_F$ is given as:

$$\hat{c}_{\overline{p}} = s^k - \sum_{(i,j) \in A} \delta_{ij}(\overline{p}) \left(c_{ij}^k - w_{ij} \right) - \sigma^k - \alpha^{O_k} + \alpha^{D_k}$$

Note that the dual values α^i cancel each other out for intermediate ports on a path in constraints (12), i.e., only the supply port O_k and the demand port D_k of a path are affected by these constraints.

As the subproblems are only dependent on the arc variables x_{ij} the constant terms given by commodity k may be treated isolated implying:

$$\sum_{(i,j) \in A} \delta_{ij}(\overline{p})(c_{ij}^k + w_{ij}) < s^k - \sigma^k - \alpha^{O_k} + \alpha^{D_k}$$

Minimizing $\sum_{(i,j)\in A}(c_{ij}+w_{ij})x_{ij}$ when finding an extreme point of $\mathcal{P}_k$ will return the path variable with the best reduced cost for the subproblem belonging to k. The subproblem corresponds to an ordinary shortest path problem with positive arc costs as $c_{ij}, w_{ij} \geq 0$:

$$\min \sum_{(ij)\in A} (c_{ij}+w_{ij})x_{ij} \tag{16}$$

$$\text{s.t.} \sum_{j\in N} x_{ij} - \sum_{j\in N} x_{ji} = 1 \qquad i = O_K \tag{17}$$

$$\sum_{j\in N} x_{ji} - \sum_{j\in N} x_{ij} = 1 \qquad i = D_k \tag{18}$$

$$\sum_{j\in N} x_{ij} - \sum_{j\in N} x_{ji} = 0 \qquad i \in N \setminus \{O_k, D_K\} \tag{19}$$

$$x_{ij} \in \{0,1\} \qquad (i,j) \in A \tag{20}$$

6 Computational Results

The experimental results are performed on data based on real life shipping networks. The test instances are created from a snapshot of the Containership Databank [6] from 2005. The set P of ports (and hence the set N of nodes) and the set of arcs A with capacities are created from [6] services. Cost and demand functions are generated randomly but such that both profitable and unprofitable products are present, demands are asymmetric in the sense that an area such as Asia should have more export than import (and vice versa for, e.g., Europe and North America), the total demand must exceed the capacity of the network for some areas, and the profit of some products must be able to support the price of leasing containers. Liner shipping operators are chosen so that the instances vary in size from 34 ships to 316 ships. Test instances are named according to the number of ships in the fleet. Please note that instance 293 is larger than instance 316 in terms of the number of ports and unique rotation legs, see Table 1.

Test instance	Ports	Unique rotation legs	Average out degree	Fleet capacity in TEU
34	44	101	2.295	21035
62	60	104	1.733	111004
98	58	122	2.103	348356
136	96	198	2.063	383179
159	117	253	2.162	422796
222	151	326	2.156	633719
293	234	565	2.415	846447
316	185	455	2.459	992479

Table 1: Test instances - instance name denotes the fleet size, e.g., 34 has a fleet size of 34 ships

All tests were performed on a *Intel(R) Xeon(R) CPU 2.66 GHz processor* with 8 GB RAM. As LP solver we have tested both *CPLEX 10.2* and the open source *CLP* solver from *COIN-OR*. All tests were performed with the *CPLEX* Barrier solver, *CPLEX* dual simplex solver and the *CLP* dual simplex solver. The best results for the arc flow model were obtained with *CPLEX* barrier solver and the best results for the path flow model were obtained with *CLP* dual simplex. Computational results are stated for the best results for each model for 1,3,6,9 and 12 time periods. The models are solved to LP-optimality. For larger test instances the arc flow model cannot be generated with the available memory. In all tests where it has been possible to generate the arc flow model, the objective value for the two models are identical. This confirms the correctness of the path flow model and the implementation of the column generation algorithm.

In the following we compare performance of the arc flow and the path flow model for test instances with 1 and 3 time periods, where the arc flow model can be generated for most instances. Next we present results for the solution times for large instances using the path flow model in conjunction with delayed column generation. For the path flow model all test instances up to 9 time periods can be solved within an hour. For 12 time periods all tests that may be generated with the available memory are solved in less than 75 minutes. Lastly, we present the integer solutions for a simple rounding heuristic applied to the LP solutions of the path flow model. The IP solutions presented are within a very reasonable distance of the LP upper bound and the gap is sufficiently small to discard the need for a branch-and-price algorithm as well as more sophisticated heuristics.

6.1 Arc flow and path flow compared

The result tables and graphs abbreviate the arc flow model to `A` and the path flow model to `P`. The size of the respective models is stated as $m \times n$. `MEM` indicates that memory was not sufficient for the process to complete. The size of the LP for the arc flow model is calculated for comparison with the size of the master problem of the path flow model. Column `time` denotes the CPU time in seconds to solve the respective model, while `iter` denotes the number of iteration for the column generation algorithm. The arc flow model is solved in one LP iteration.

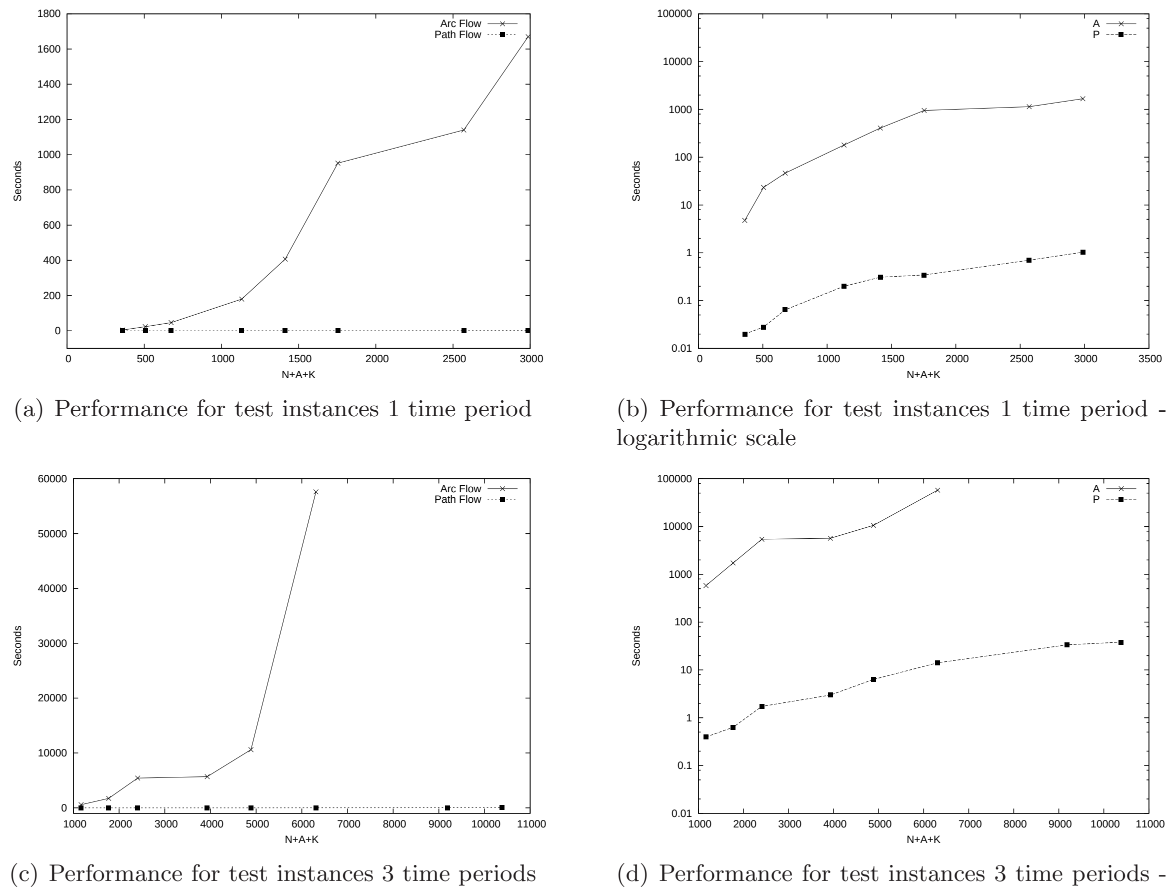
Table 2 shows the relative performance of the arc flow model and the path flow model for 1 and 3 time periods respectively. The path flow formulation in conjunction with delayed column generation outperforms the arc flow model by a wide margin even for small instances. The size of the LPs for the column generation algorithm is surprisingly small and the column generation algorithm is at least two orders of magnitude faster than the arc flow model for one time period and three orders of magnitude faster for three time periods.

	arc flow model			path flow model			
Test no.	$m \times n$	objective	time	$m \times n$	objective	time	iter
34-01	21860 × 9605	$2.59 \cdot 10^{07}$	4.76	360 × 378	$2.59 \cdot 10^{07}$	0.02	4
62-01	35732 × 20684	$1.21 \cdot 10^{09}$	23.30	506 × 542	$1.21 \cdot 10^{09}$	0.03	4
98-01	60448 × 28832	$5.37 \cdot 10^{09}$	46.40	674 × 889	$5.37 \cdot 10^{09}$	0.06	4
136-01	166020 × 80646	$4.32 \cdot 10^{09}$	180.00	1131 × 1589	$4.32 \cdot 10^{09}$	0.20	6
159-01	264249 × 122401	$4.27 \cdot 10^{09}$	407.00	1413 × 2155	$4.27 \cdot 10^{09}$	0.31	8
222-01	416779 × 193304	$7.92 \cdot 10^{09}$	952.00	1754 × 2285	$7.92 \cdot 10^{09}$	0.34	5
293-01	1237583 × 510835	$1.1 \cdot 10^{10}$	1670.00	2987 × 4008	$1.1 \cdot 10^{10}$	1.03	8
316-01	878790 × 357690	$1.22 \cdot 10^{10}$	1140.00	2570 × 3352	$1.22 \cdot 10^{10}$	0.70	6
34-03	252718 × 85663	$7.9 \cdot 10^{07}$	580	1168 × 2962	$7.9 \cdot 10^{07}$	0.40	5
62-03	501300 × 209232	$4.3 \cdot 10^{09}$	1730	1771 × 2159	$4.3 \cdot 10^{09}$	0.63	10
98-03	844638 × 305330	$1.98 \cdot 10^{10}$	5420	2407 × 3638	$1.98 \cdot 10^{10}$	1.73	15
136-03	2245890 × 823602	$1.55 \cdot 10^{10}$	5690	3930 × 5863	$1.55 \cdot 10^{10}$	2.98	11
159-03	3518550 × 1244586	$2.04 \cdot 10^{10}$	10600	4886 × 8894	$2.04 \cdot 10^{10}$	6.34	18
222-03	5860293 × 2075114	$2.62 \cdot 10^{10}$	57600	6310 × 10039	$2.62 \cdot 10^{10}$	14.10	16
293-03	16257902 × 5260738	-	MEM	10382 × 16620	$3.65 \cdot 10^{10}$	38.20	14
316-03	11965115 × 3829015	-	MEM	9185 × 14097	$5.06 \cdot 10^{10}$	33.50	24

Table 2: Test instances for 1 and 3 time periods

Figures 8(a)–8(b) plot the solution time of the algorithms as a function of the size of the

LP model. The unit $|N|+|A|+|K|$ is chosen at the $x-$axis because it is the decisive factors in the size of the LP constraint set for both models. Figure 8(a) shows a fast growth in solution time for the arc flow model (full lines). Using a logarithmic scale in Figure 8(b) it is seen that the growth is exponential. The solution times of the path flow model (dotted lines) grow more moderately. Figures 8(c)–8(d) correspond to figures 8(a)–8(b) when considering three time periods. Again, Figure 8(d) shows an exponential growth of the arc flow model (the two largest instances could not be generated due to memory limitations).



(a) Performance for test instances 1 time period

(b) Performance for test instances 1 time period - logarithmic scale

(c) Performance for test instances 3 time periods

(d) Performance for test instances 3 time periods - logarithmic scale

Figure 8: Relative performance of the arc flow and path flow model, 1 and 3 time periods

6.2 Solution of large instances

We now consider the solution times for 6, 9 and 12 time periods. The arc flow model cannot be generated for the largest instances and hence is not discussed further in this section.

Table 3 shows that test instances with 6 time periods solved with the path flow model complete within 10 minutes. The master problems are small compared to the arc flow model and the number of iterations is reasonable. The sparsity of the networks probably results in few path variables for a commodity, which leads to relatively fast convergence of the delayed column generation algorithm. Notice that test 316 is slower than test 293 in spite of a smaller LP. This might be specific for test 316 but may also be due to degeneracy, ϵ rounding or many

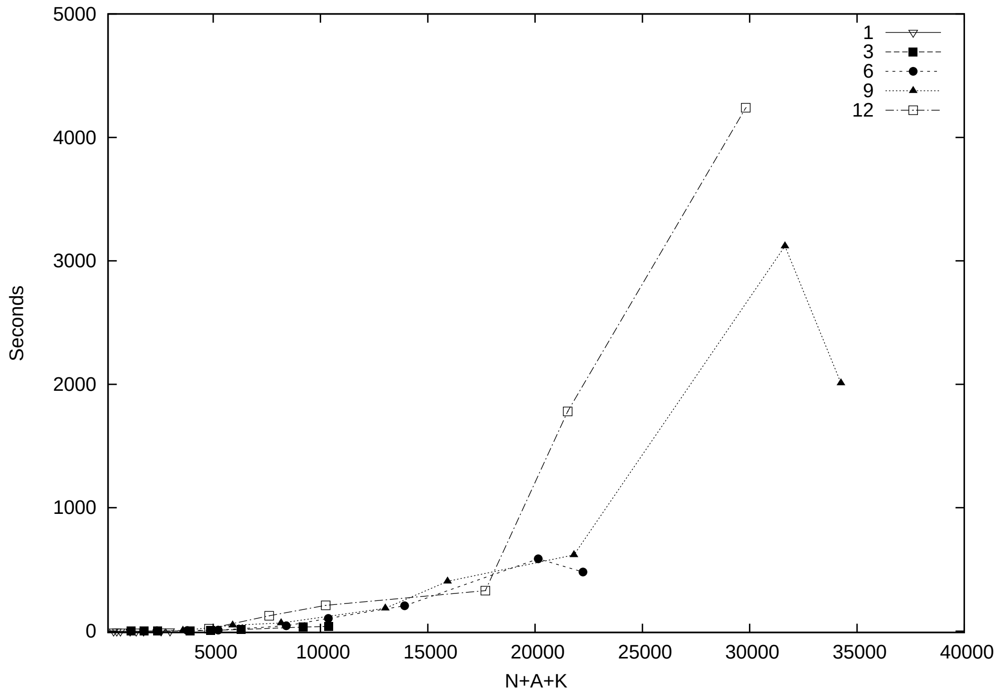
	arc flow model			path flow model			
Test no.	$m \times n$	objective	time	$m \times n$	objective	time	iter
34-06	1066630 × 341650	$2.49 \cdot 10^{08}$	3410	2380 × 11070	$2.49 \cdot 10^{08}$	4.56	7
62-06	2325144 × 906684	$9.18 \cdot 10^{09}$	7300	3799 × 5305	$9.18 \cdot 10^{09}$	5.70	19
98-06	3951400 × 1346390	$5.03 \cdot 10^{10}$	35100	5235 × 7989	$5.03 \cdot 10^{10}$	8.61	12
136-06	3552132 × 10282128	-	MEM	8407 × 15251	$3.9 \cdot 10^{10}$	43.70	17
159-06	5310627 × 15903588	-	MEM	10366 × 24020	$4.09 \cdot 10^{10}$	104.00	24
222-06	9339332 × 27929628	-	MEM	13918 × 25635	$7.29 \cdot 10^{10}$	206.00	23
293-06	22762597 × 74147688	-	MEM	22231 × 41419	$7.71 \cdot 10^{10}$	480.00	23
316-06	17078785 × 56225975	-	MEM	20147 × 36229	$1.18 \cdot 10^{11}$	586.00	29
34-09	767917 × 2441692	$4.98 \cdot 10^{08}$	12500	3592 × 11205	$4.98 \cdot 10^{08}$	5.7	9
62-09	2134956 × 5595156	-	MEM	5906 × 12790	$1.78 \cdot 10^{10}$	48.7	89
98-09	3176366 × 9500606	-	MEM	8165 × 14201	$7.11 \cdot 10^{10}$	66.1	21
136-09	8302998 × 24496164	-	MEM	13020 × 24212	$7.17 \cdot 10^{10}$	186.0	41
159-09	12274875 × 37445355	-	MEM	15919 × 51822	$7.36 \cdot 10^{10}$	404.0	28
222-09	22172150 × 67565663	-	MEM	21812 × 42444	$1.26 \cdot 10^{11}$	618.0	17
293-09	52866028 × 175158501	-	MEM	34252 × 69529	$1.37 \cdot 10^{11}$	2010.0	27
316-09	17078785 × 56222321	-	MEM	31643 × 63309	$2.18 \cdot 10^{11}$	3120.0	45
34-12	1364464 × 4377904	-	MEM	4804 × 20557	$3.69 \cdot 10^{08}$	20	5
62-12	3587508 × 9502560	-	MEM	7607 × 36750	$1.4 \cdot 10^{10}$	125	27
98-12	5183822 × 15650086	-	MEM	10242 × 39919	$8.62 \cdot 10^{10}$	210	18
136-12	15092328 × 44953488	-	MEM	17681 × 36455	$1.24 \cdot 10^{11}$	329	30
159-12	22176291 × 68270220	-	MEM	21518 × 87218	$9.42 \cdot 10^{10}$	1780	48
222-12	40655981 × 125020921	-	MEM	29818 × 67841	$1.72 \cdot 10^{11}$	4240	32
293-12	95531887 × 319187701	-	MEM	46302 × 114088*	-	MEM	25*
316-12	74721600 × 252239000	-	MEM	43369 × 90759*	-	MEM	40*

Table 3: Test instances for 6, 9 and 12 time periods. * indicates the size of the LP and the last iteration of the process when aborting due to insufficient memory

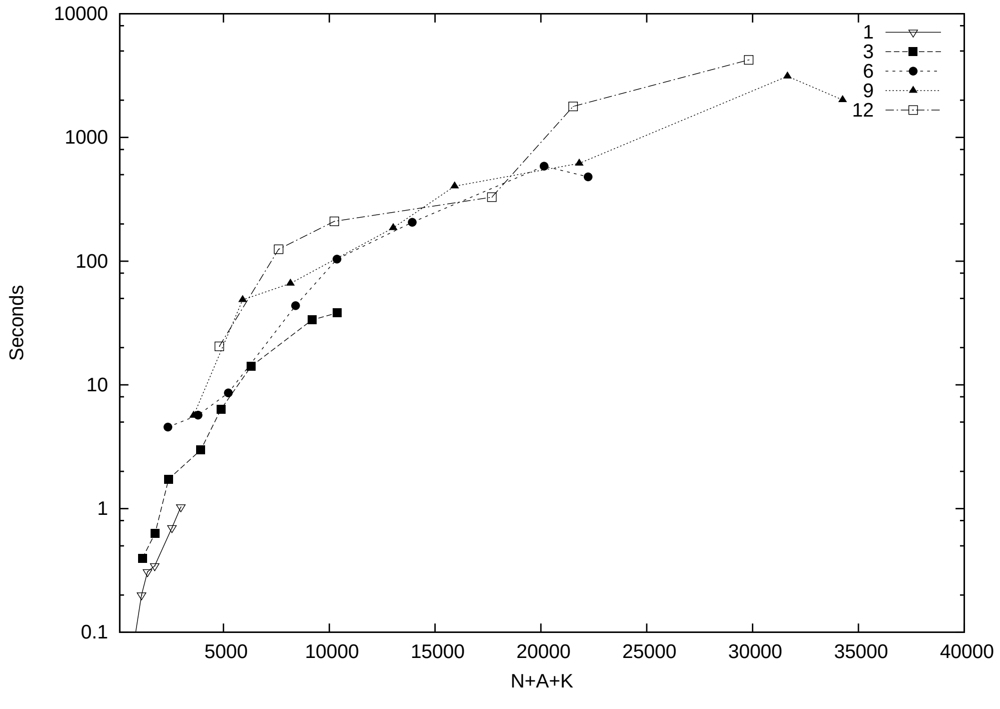
cache misses. Test instances with 9 time periods may all be completed in less than one hour. All but the two largest tests, 293 and 316 complete within 12 minutes. The number of columns in the two largest test instances is significant. Again we see test 316 needing more time and iterations to complete than test 293 although the LP is smaller. The increased solution time seems to be problem specific and it is interesting to note that the network of test 316 is denser than that of test 293. This supports the theory that the sparsity of shipping networks is a key to success for the path flow model and the column generation algorithm. For 9 time periods we see that the number of iterations varies from 9 to 89. However, execution times still grow steadily and the size of the LPs is reasonable considering the size of the networks. For 12 time periods the LPs have reached a critical size and tests 293 and 316 do not have sufficient memory to complete — the * indicates the size of the LP and the iteration just before the process was aborted. The number of iterations for the remaining test instances is reasonable and the solution times are still within 75 minutes which is good for such large models representing a shipping network for a whole year.

Figure 9(a)-9(b) show the solution time of the path flow model for 1,3,6,9 and 12 time periods as a function of LP size. It is seen that the growth in solution times is relatively steady for 3 time periods. Figure 9(a) shows an exponential tendency for the graphs of 9 and 12 time periods, where the LPs have reached a critical size. The trend is even more explicit in figure 9(b) where the graphs are plotted on a logarithmic scale. The trend is particular for the larger test instances, which have denser networks and hence, more path variables per commodity. The exponential tendency is very clear in the graph of 12 time periods although the two largest tests did not complete.

The delayed column generation method shows good convergence for the generated test instances, and methods to reduce the master problem constraint set has not been required.



(a) Performance of model P for 3,6,9 and 12 time periods



(b) Performance of model P for 3,6,9 and 12 time periods - logarithmic scale

Figure 9: Relative performance of the path flow model - 1,3,6,9 and 12 time periods

We are able to solve instances of large shipping networks spanning 9 months in less than one hour. For 12 months the two largest instances cannot be generated with the available memory, but the remaining tests which have a significant size are solved within 75 minutes. The convergence is suspected to be correlated to the sparsity of the networks. The results of the tests show that the column generation technique is very effective for solving RMREC for sparse networks. The path flow model and column generation algorithm outperforms solving the arc flow model by a wide margin. It appears that the number of path variables for real life liners is very modest and therefore the number of variables in the restricted master problem is relatively small. Services, capacities and ports are based on the real world, the cost structure is randomly generated. This indicates that RMREC will perform well on real life problems, but the real partition of surplus/deficit zones for empty containers and the commodity set of a real life instance might be harder to solve than the generated instances presented in this paper.

6.3 Quality of integer solutions and speed of rounding heuristic

In this section we present the integer solutions obtained by a simple rounding heuristic of the LP solution provided by the path flow model. We show that the integer solutions have a very small gap to the LP upper bound. An integer solution to RMREC is obtained by rounding down all fractional variables. At nodes with violated inter-balancing constraints we supply empty containers through the leasing variable to maintain feasibility. Table 4 shows the integer solutions obtained by rounding. For each test instance we report the fraction of fractional basis variables (`Frac/basis`), the percentage of fractional basis variables (`Frac %`), the fractional moves as a percentage of total moves (`Rounded %`), the gap (`obj gap`) and gap percentage (`gap %`) between the LP and IP solution and the CPU time in seconds (`time`).

Table 4 shows that 10 out of 38 ($\approx 26\%$) of the LP solutions are already integer. Test instance 34 is integer throughout. The remaining integer LP solutions are found in time periods one and three. 20 out of 38 ($\approx 53\%$) LP solutions have less than 10% fractional basis variables. The highest percentage among the remaining 18 LP solutions is 21.8%. The most fractional solutions seems to be test instances with 9 time periods. Despite having more than 20% fractional basis variables the amount of flow rounded is never more than 0.3% of the total flow and the gap percentage in terms of the objective value is never higher than 0.01%. This confirms that the rounded integer solution is a good solution in terms of the gap to the LP upper bound. This is probably due to generally large flows on path variables making the rounding insignificant. Execution times are mostly less than one second but on larger instances execution times rise to at most 5 seconds. The optimal integer solution might be slightly better, but given a gap percentage less than 10^{-4} the computation time needed for a branch-and-price algorithm does not seem justified since the gap is smaller than the data uncertainty.

7 Concluding Remarks

We have presented a mathematical model for the container revenue management problem considering empty repositioning and solved it to near-optimality using delayed column generation and rounding. To the best of our knowledge a revenue management model considering empty repositioning has not been presented in the literature before. The mathematical model is surprisingly simple. The inter-balancing constraints, which ensure repositioning of empty

Test	Frac/basis	Frac %	Rounded %	obj gap	gap %	time
34-01	0/183	0.0	0.0	0	0.0	0.01
62-01	0/214	0.0	0.0	0	0.0	0.01
98-01	0/331	0.0	0.0	0	0.0	0.01
136-01	10/538	1.9	$2.7 \cdot 10^{-4}$	13800	$3.2 \cdot 10^{-6}$	0.01
159-01	0/464	0.0	0.0	0	0.0	0.02
222-01	0/836	0.0	0.0	0	0.0	0.02
293-01	23/1430	1.6	$2.2 \cdot 10^{-4}$	2940	$2.7 \cdot 10^{-6}$	0.06
316-01	28/1210	2.3	$2.5 \cdot 10^{-4}$	34100	$2.8 \cdot 10^{-6}$	0.05
34-03	0/588	0.0	0.0	0	0.0	0.02
62-03	46/723	6.4	$1.4 \cdot 10^{-3}$	81900	$1.9 \cdot 10^{-6}$	0.03
98-03	116/1020	11.4	$1.06 \cdot 10^{-3}$	271000	$1.37 \cdot 10^{-5}$	0.03
136-03	0/1680	0.0	0.0	0	0.0	0.08
159-03	16/1500	1.1	$1.32 \cdot 10^{-4}$	62000	$3.05 \cdot 10^{-6}$	0.15
222-03	324/2600	12.5	$1.81 \cdot 10^{-3}$	767000	$2.93 \cdot 10^{-5}$	0.20
293-03	375/4450	8.4	$1.41 \cdot 10^{-3}$	902000	$2.47 \cdot 10^{-5}$	0.48
316-03	161/3620	4.4	$5.03 \cdot 10^{-4}$	518000	$1.02 \cdot 10^{-5}$	0.36
34-06	0/1060	0.0	0.0	0	0.0	0.04
62-06	102/1330	7.7	$1.7 \cdot 10^{-3}$	457000	$4.98 \cdot 10^{-5}$	0.10
98-06	290/1960	14.8	$1.37 \cdot 10^{-3}$	1340000	$2.66 \cdot 10^{-5}$	0.14
136-06	453/3290	13.8	$2.17 \cdot 10^{-3}$	1610000	$4.13 \cdot 10^{-5}$	0.36
159-06	401/3130	12.8	$1.88 \cdot 10^{-3}$	1620000	$3.96 \cdot 10^{-5}$	0.56
222-06	969/5060	19.1	$2.40 \cdot 10^{-3}$	3850000	$5.28 \cdot 10^{-5}$	0.86
293-06	1114/9460	11.8	$2.14 \cdot 10^{-3}$	3650000	$4.74 \cdot 10^{-5}$	2.01
316-06	1281/7660	16.7	$2.04 \cdot 10^{-3}$	5220000	$4.44 \cdot 10^{-5}$	1.55
34-09	0/1560	0.0	0.0	0	0.0	0.10
62-09	259/1990	13.0	$3.22 \cdot 10^{-3}$	1810000	$1.02 \cdot 10^{-4}$	0.21
98-09	673/3080	21.8	$2.18 \cdot 10^{-3}$	3770000	$5.30 \cdot 10^{-5}$	0.29
136-09	744/4800	15.5	$2.44 \cdot 10^{-3}$	4260000	$5.94 \cdot 10^{-5}$	0.82
159-09	637/4910	13.0	$1.89 \cdot 10^{-3}$	2960000	$4.03 \cdot 10^{-5}$	1.27
222-09	1568/7610	20.6	$3.05 \cdot 10^{-3}$	8070000	$6.41 \cdot 10^{-5}$	1.99
293-09	1838/13900	13.2	$2.39 \cdot 10^{-3}$	7540000	$5.49 \cdot 10^{-5}$	4.93
316-09	2357/11500	20.5	$2.61 \cdot 10^{-3}$	12900000	$5.92 \cdot 10^{-5}$	3.70
34-12	0/2030	0.0	0.0	0	0.0	0.15
62-12	50/2470	2.0	$4.53 \cdot 10^{-4}$	366000	$2.61 \cdot 10^{-5}$	0.38
98-12	532/3300	16.1	$1.25 \cdot 10^{-3}$	2870000	$3.32 \cdot 10^{-5}$	0.49
136-12	630/6280	10.0	$1.63 \cdot 10^{-3}$	5470000	$4.41 \cdot 10^{-5}$	1.46
159-12	984/6410	15.4	$2.18 \cdot 10^{-3}$	4870000	$5.17 \cdot 10^{-5}$	2.30
222-12	2163/10600	20.3	$3.05 \cdot 10^{-3}$	13500000	$7.89 \cdot 10^{-5}$	3.59
293-12	-	-	-	-	-	-
316-12	-	-	-	-	-	-

Table 4: Rounded integer solutions - 1-12 time periods

containers, results in an augmented multi-commodity flow problem. It appears that these constraints do not complicate the model to an extent where the solution time is affected. However, it should be noted that in real life instances the mapping of surplus and deficit zones can make it harder to fulfill the inter-balancing constraints. Furthermore, test results show that the inter-balancing constraints ensure transportation of low profitable products before repositioning empty containers to deficit ports [14]. This demonstrates the importance of considering empty container repositioning in a booking model for *Liner shipping*. Another strength of RMREC is its ability to route products along the cheapest path. The size of the instances created and solved in this paper are significantly larger than previously reported in the reviewed literature.

Solving the LP-relaxed model with delayed column generation turned out to be very successful compared to solving the arc flow model with *CPLEX* barrier solver. The column generation algorithm is at least two orders of magnitude faster for one time period and three orders of magnitude faster for three time periods. The path flow formulation of RMREC solved by a simple delayed column generation algorithm is able to solve all instances for 6 time periods in 586 seconds. For 9 time periods test instance 316 containing 1665 nodes (185 ports in 9 periods), 5575 arcs and 24403 commodities is solved in 3120 seconds. For 12 periods the two largest test instances cannot be solved within the space limit. The largest instance completed contains 1812 nodes (151 ports in 12 periods), 5573 arcs and 22433 commodities and is solved in 4240 seconds. A rounding heuristic is applied to the LP solutions of the path flow model with great success. The heuristic finds a solution in less than 5 seconds. All integer solutions have a gap to the LP upper bound of at most 0.01% which is well below the data uncertainty.

A suggestion for future work is to study the start and end conditions for RMREC, using a rolling horizon schedule. The aspect of continuous networks is also theoretically interesting. Substitution of containers could also be investigated further. It is believed that substitution can be implemented with success for RMREC. The model may be generalized in many ways, e.g., by defining a minimum bound in ship capacity utilization. Some of the modifications may result in hard pricing problems, but we believe that even a complex pricing problem may be solved in reasonable time as the graph may be split into subgraphs according to time periods, and since paths are generally very short. Furthermore, pricing problems may be parallelized to decrease solution times. Reinhardt [15] solve a multi-objective shortest path problem for liner shipping with non-additive costs. These techniques could be relevant for RMREC because it is likely that several criteria needs to be taken into account when defining the attractiveness of a path and various strategic goals may have a non-additive cost structure. To fully incorporate the transportation problem of an international liner shipping company, a model could be developed that considers the transport of containers from customer to customer. An obvious area of future work is to incorporate booking and empty repositioning into routing/scheduling decisions of the vessel fleet as the overall cost of running a liner shipping company is the fixed cost of committing to a schedule.

References

[1] R.K. Ahuja, T.L. Magnanti, and J.B. Orlin. *Network Flows - Theory, Algorithms and Applications.* Prentice Hall, 1993. ISBN 0-13-617549-X.

[2] P. Bartodziej and U. Derigs. *Experimental and Efficient Algorithms.* Springer Berlin/Heidelberg, 2004. ISBN 978-3-540-22067-1.

[3] N. Bélanger, G. Desaulniers, F. Soumis, and J. Desrosiers. Periodic airline fleet assignment with time windows, spacing constraints, and time dependent revenues. *European Journal of Operational Research*, 175(3):1754–1766, 2006. doi: 10.1016/j.ejor.2004.04.051.

[4] R.K. Cheung and C.-Y. Chen. A two-stage stochastic network model and solution methods for the dynamic empty container allocation problem. *Transportation Science*, 32(2): 142–162, 1998. doi: 10.1287/trsc.32.2.142.

[5] M. Christiansen, D. Ronen, B. Nygreen, and K. Fagerholt. Maritime transportation. *Handbooks in Operations Research and Management Sciences*, 14:189–284, 2007.

[6] Containership Databank. Containership databank. URL `http://www.mdst.co.uk`.

[7] T.G. Crainic, P.J. Dejax, and L. Delorme. Models for multimode multicommodity location problems with interdepot balancing requirements. *Annals of Operational Research*, 18(1):277–302, 1989. doi: 10.1007/BF02097809.

[8] T.G. Crainic, M. Gendreau, and P.J. Dejax. Dynamic and stochastic models for the allocation of empty containers. *Operations Research*, 41(1):102–126, 1993. doi: 10.1287/opre.41.1.102.

[9] P.J. Dejax and T.G. Crainic. Survey paper - a review of empty flows and fleet management models in freight transportation. *Transportation Science*, 21(4):227–247, 1987. doi: 10.1287/trsc.21.4.227.

[10] G. Desaulniers, J. Desroisers, Y. Dumas, S. Marc, B. Rioux, M. M. Solomon, and F. Soumis. Crew pairing at Air France. *European Journal of Operations Research*, 97:245 – 259, 1997. doi: 10.1016/S0377-2217(96)00195-6.

[11] C.A. Hane, C. Barnhart, E.L. Johnson, R.E. Marsten, G.L. Nemhauser, and G. Sigismondi. The fleet assignment problem solving a large scale integer program. *Mathematical Programming*, 70:211–232, 1995. doi: 10.1007/BF01585938.

[12] IICL. Institute of international container lessors. URL `http://www.iicl.org`.

[13] J. Li, K. Liu, S.C.H. Leung, and K.K Lai. Empty container management in a port with long-run average criterion. *Mathematical and Computer Modelling*, 40:85–100, 2004. doi: 10.1016/j.mcm.2003.12.005.

[14] B. Løfstedt. Crmer - container revenue management with empty repositioning. Master's thesis, Department of Computer Science, Copenhagen University, 2007.

[15] L.B. Reinhardt. Multi objective shortest path for cargo transportation. Master's thesis, Department of Computer Science, Copenhagen University, 2005.

[16] D. Ronen, K. Fagerholt, and M. Christiansen. Ship routing and scheduling: Status and perspectives. *Transportation Science*, 38(1):1–18, 2004. doi: 10.1287/trsc.1030.0036.

[17] W.S. Shen and C.M. Khoong. A DSS for empty container distribution planning. *Decision Support Systems*, 15(1):75–82, 1995. doi: 10.1016/0167-9236(94)00037-S.

[18] K. Shintani, A. Imai, E. Nishimura, and S. Papadimitriou. The container shipping network design problem with empty container repositioning. *Transportation Research Part E: Logistics and Transportation Review*, 43(1):39–59, 2007. doi: 10.1016/j.tre.2005.05.003.

[19] H.A. Taha. *Operations Research*. Prentice Hall, 7 edition, 2003. ISBN 0-13-048808-9.

[20] TAL. Tal international container corporation. URL `http://www.talinternational.com`.