Container Terminal Management: Integrated Models and Large-Scale Optimization Algorithms

THÈSE N^{O} 4926 (2011)

PRÉSENTÉE LE 4 FÉVRIER 2011

À LA FACULTÉ SCIENCES DE BASE
LABORATOIRE TRANSPORT ET MOBILITÉ
PROGRAMME DOCTORAL EN MATHÉMATIQUES

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

POUR L'OBTENTION DU GRADE DE DOCTEUR ÈS SCIENCES

PAR

Ilaria VACCA

acceptée sur proposition du jury:

Prof. F. Eisenbrand, président du jury
Prof. M. Bierlaire, Dr M. Salani, directeurs de thèse
Prof. M. Christiansen, rapporteur
Prof. N. Geroliminis, rapporteur
Prof. M.G. Speranza, rapporteur

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Suisse
2011

Acknowledgments

This thesis wouldn't have been possible without the precious help and support that I received from many people during my stay at EPFL and in Switzerland.

My supervisor and mentor, Michel Bierlaire. He gave me the opportunity to pursue my research in a challenging and stimulating working environment. It has been a privilege to be guided by such a brilliant and demanding scientist, who has always pushed me to do my best and to discover my potential. His generosity and positive approach to life have enriched this experience professionally and personally, and I'll be forever thankful for this.

My co-supervisor and friend, Matteo Salani. This thesis is the result of our day-to-day work, made of continuous discussions and fruitful exchanges. He was always encouraging, patient and supportive. He shared with me all the efforts, disappointments, surprises and success that research is made of, and greatly contributed to make this experience successful. I've learnt from him how to be a good researcher and I express to him and his family my deepest gratitude.

I thank Luigi Moccia and Giovanni Giallombardo for sharing with me their expertise on container terminal management when I was at the early stage of my research. They co-authored the work on the Tactical Berth Allocation Problem presented in Chapter 3.

I'm thankful to all the members of my thesis committee, for the interesting discussion and their constructive commentaries to the manuscript.

I thank all the members of the Transport and Mobility Laboratory for welcoming me in a very nice and friendly environment. It has been a pleasure to be part of this group and I've enjoyed a lot the great working atmosphere in our lab.

I'm particularly thankful to the new friends I've had the chance to meet during my stay in Lausanne: you made me feel home! And to all the farther friends that, beyond the distance, have supported me in this Swiss adventure.

A special thanks to my parents and to my sister. They have always encouraged me to pursue my freedom and to realize my wishes, with love and enthusiasm.

Finally, words are not enough to thank Stefano, my soon-to-be husband, for his constant love and continuous support. I dedicate this thesis to him with all my love.

The last year of my research was funded by the Swiss National Science Foundation, grant 200021_125317.

Abstract

This thesis deals with models and methods for large scale optimization problems; in particular, we focus on decision problems arising in the context of seaport container terminals for the efficient management of terminal operations.

Large-scale optimization problems are both difficult to handle and important in many concrete contexts. They usually originate from real world applications, such as telecommunication, transportation and logistics, and their combinatorial complexity often represents a major issue; therefore, optimization models are crucial to support the decision making process. In particular, column generation and branch-and-price schemes currently represent one of the most advanced and efficient exact optimization approaches to solve large scale combinatorial problems. However, the increasing size and complexity of practical problems arising in real-world applications motivates the design of new solution approaches able to tackle current optimization challenges.

In this thesis, we address two complementary research streams where both methods and applications play an important role. On the one hand, we focus on the specific application of container terminals: we propose a new model for the integrated planning of operations and we provide a heuristic and an exact solution algorithm; the broader objective is to devise solution methods that can be generalized and extended to other applications and domains. On the other hand, we aim to develop new methods and algorithms for general large scale problems and, in this context, we investigate a new column generation framework that exploits the relationship between compact and extensive formulation. In particular, we focus on a class of split delivery vehicle routing problems that generalizes a large number of applications arising in the real world, such as transportation and logistics, including container terminal management.

In the context of container terminals, we propose a model for the integrated planning of berth allocation and quay crane assignment: the two decision problems are usually solved hierarchically by terminal planners, whereas in the Tactical Berth Allocation Problem we optimize the two problems simultaneously. We firstly present a mixed integer programming formulation that is embedded into a two-level heuristic algorithm based on tabu search and mathematical programming techniques: our heuristic proves to be very efficient, providing good-quality solutions in a reasonable time. The problem is reformulated via Dantzig-Wolfe decomposition and solved via column generation: we propose an exact branch-and-price algorithm and our implementation, that includes state-of-the-art techniques for the master and the pricing

problem, outperforms commercial solvers. Furthermore, the exact approach allows us to provide an interesting experimental comparison between hierarchical and integrated planning: computational tests confirm the added value of integration in terms of cost reduction and efficient use of resources.

From a methodological point of view, this dissertation investigates a new column generation concept for difficult large scale optimization problems. In particular, we study a class of split delivery vehicle routing problems that generalizes some interesting features of Tactical Berth Allocation Problem, which are relevant also to other applications such as transportation, logistics and telecommunication.

The problem, called Discrete Split Delivery Vehicle Routing Problem with Time Windows, presents two main modeling features: demand is discrete and delivered in discrete orders, opposite to the usual assumption of continuously splittable demand; the service time is dependent on the delivered quantity, opposite to the usual assumption of constant service time, regardless of the quantity. The problem is used to validate and test the new column generation approach studied in this thesis.

The proposed framework, called Two-stage column generation, represents a novel contribution to recent advances in column generation: the basic idea is to simultaneously generate columns both for the compact and the extensive formulation. We propose to start solving the problem on a subset of compact formulation variables, we apply Dantzig-Wolfe decomposition and we solve the resulting master problem via column generation. At this point, profitable compact formulation variables are dynamically generated and added to the formulation according to reduced cost arguments, in the same spirit of standard column generation. The key point of our approach is that we evaluate the contribution of compact formulation variables with respect to the extensive formulation: indeed, we aim at adding compact formulation variables that are profitable for the master problem, regardless of the optimal solution of the linear relaxation of the compact formulation.

We apply two-stage column generation to the Discrete Split Delivery Vehicle Routing Problem with Time Windows. Computational results show that our approach significantly reduces the number of generated columns to prove optimality of the root node. Furthermore, suboptimal compact formulation variables are detected correctly and a large number of variables is not taken into account during the solution process, thus reducing the size of the problem. However, the additional effort required by such a sophisticated approach makes the method competitive in terms of computational time only for instances of a certain difficulty.

To conclude, two-stage column generation is a promising new approach and we believe that further research in this direction may contribute to solve more and more complex large scale optimization problems.

Keywords container terminal, berth allocation, quay crane assignment, integrated planning, large scale optimization, Dantzig-Wolfe decomposition, branch-and-price, split delivery vehicle routing, two-stage column generation.

Sommario

Questa tesi si occupa di metodi e modelli per problemi di ottimizzazione su larga scala; in particolare, trattiamo problemi decisionali che si presentano nei terminal portuali per una gestione efficiente delle operazioni di movimentazione dei container.

I problemi di ottimizzazione su larga scala sono importanti in molti contesti concreti, ma difficili da trattare analiticamente. In genere derivano da applicazioni reali (telecomunicazioni, trasporti, logistica) e la loro complessità pone ostacoli alla loro risoluzione; per questa ragione i metodi di ottimizzazione risultano fondamentali nel supporto alle decisioni. In particolare, i metodi di generazione di colonne e branch-and-price sono attualmente tra le tecniche più avanzate ed efficienti per l'ottimizzazione esatta di problemi su larga scala. Tuttavia, la crescente dimensione e complessità di tali problemi derivanti da applicazioni reali motivano l'elaborazione di nuovi approcci risolutivi capaci di affrontare le attuali sfide in ottimizzazione.

In questa tesi ci occupiamo di due filoni di ricerca complementari tra loro, in cui sia i metodi che le applicazioni hanno un ruolo fondamentale. Da un lato ci concentriamo su applicazioni specifiche dei terminal container: presentiamo un nuovo modello per la pianificazione integrata delle operazioni e proponiamo due algoritmi risolutivi; l'obiettivo ultimo è l'elaborazione di metodi che possano essere facilmente estesi ad altri domini di applicazione. Dall'altro, miriamo ad elaborare nuovi metodi generici per problemi di ottimizzazione su larga scala ed in tale contesto studiamo un nuovo schema di generazione di colonne che si basa sulla relazione tra formulazione compatta e formulazione estesa. In particolare, ci concentriamo su una nuova classe di problemi di instradamento di veicoli con frazionamento della domanda, che generalizza diverse applicazioni reali, tra cui alcuni problemi di trasporti e logistica.

Nell'ambito dei terminal container proponiamo un modello per la gestione integrata degli attracchi e delle gru di banchina: in genere questi due problemi vengono risolti in maniera sequenziale, mentre nell'approccio integrato sono risolti contemporaneamente. Presentiamo una formulazione mista intera che viene utilizzata da un algoritmo euristico basato su tabu search e tecniche di programmazione matematica. L'euristica proposta è efficiente e produce soluzioni di buona qualità in poco tempo. Il problema è riformulato tramite decomposizione Dantzig-Wolfe e risolto con generazione di colonne: l'algoritmo di branch-and-price che ne risulta include lo stato dell'arte per le tecniche di accelerazione dei problemi di master e di pricing, ed è superiore a solutori commerciali. Inoltre, l'approccio esatto ci permette di fornire un

interessante confronto sperimentale tra la pianificazione sequenziale e la pianificazione integrata: i risultati computazionali confermano il valore aggiunto dell'integrazione in termini di riduzione dei costi ed uso efficiente delle risorse.

Da un punto di vista metodologico, questa tesi propone un nuovo concetto di generazione di colonne per problemi difficili di ottimizzazione su larga scala. In particolare, ci concentriamo su una classe di problemi di instradamento di veicoli che generalizza alcune interessanti caratteristiche del nostro approccio integrato per la gestione degli attracchi e delle gru di banchina; tali caratteristiche sono rilevanti anche per altre applicazioni nei settori di trasporti, logistica e telecomunicazioni.

Il problema presenta due peculiarità: la domanda è discreta e recapitata in ordini anch'essi discretizzati, contrariamente alla classica ipotesi di domanda frazionabile in maniera continua; il tempo di servizio dipende dalla quantità che viene recapitata al cliente, contrariamente alla classica ipotesi di tempo di servizio costante, indipendentemente dalla quantità. Il problema è utilizzato per validare e testare il nuovo approccio di generazione di colonne studiato in questa tesi.

Lo schema proposto, detto generazione di colonne a due fasi, rappresenta un contributo originale nel campo della generazione di colonne: l'idea di base consiste nel generare contemporaneamente colonne per la formulazione compatta e per la formulazione estesa. Proponiamo di risolvere inizialmente il problema su un sottoinsieme di variabili compatte, riformuliamo tramite decomposizione Dantzig-Wolfe e risolviamo il problema di master tramite generazione di colonne. A questo punto, le variabili della formulazione compatta sono generate dinamicamente ed incluse nella formulazione in base al valore dei costi ridotti, come per la generazione di colonne standard. Il punto chiave del nostro approccio consiste nel valutare il contributo di una variabile compatta rispetto alla formulazione estesa: infatti, il nostro scopo è generare variabili compatte che sono vantaggiose per il problema di master, indipendentemente dalla soluzione ottima del rilassamento lineare della formulazione compatta.

Applicando lo schema di generazione di colonne a due fasi al nostro problema di instradamento di veicoli otteniamo una riduzione significativa del numero di colonne, come mostrato dai risultati computazionali. Inoltre, le variabili della formulazione compatta che sono sub-ottimali per la formulazione estesa sono identificate correttamente e un numero importante di variabili non viene considerato durante il processo risolutivo, riducendo quindi la dimensione del problema. Tuttavia, il carico computazionale aggiuntivo richiesto da un approccio così sofisticato, fa sì che il nostro metodo risulti competitivo a livello di tempi solo su istanze di una certa difficoltà.

In conclusione, la generazione di colonne a due fasi è un approccio nuovo e promettente, e crediamo fermamente che ulteriori ricerche in questa direzione possano contribuire a risolvere problemi di ottimizzazione su larga scala sempre più complessi.

Parole-chiave: terminal container, gestione degli attracchi, assegnazione gru di banchina, pianificazione integrata, decomposizione Dantzig-Wolfe, branch-and-price, problemi di instradamento di veicoli, generazione di colonne a due fasi.

Contents

List of Tables

List of Figures

Chapter 1

Outline

Large-scale optimization problems are both difficult to handle and important in many concrete contexts. They usually arise in real world applications, such as telecommunication, transportation and logistics. The complexity of these problems typically originates from very complex networks, non-linearities in objectives and constraints, highly inter-connected decisions and complex feasibility rules. Therefore, optimization models are crucial to support the decision making process.

Column generation and branch-and-price schemes represent nowadays the most successful tool to solve integer large-scale optimization problems that commercial solvers could never cope with. However, practical problems of growing size and complexity represent a challenge for the research community and the need of further advances in column generation, both theoretically and algorithmically, is well recognized.

In this thesis, both methods and applications play an important role. In particular, we focus on decision problems arising in the context of seaport container terminals, where the efficient management of logistic activities represents a major issue. Not surprisingly, the optimization of container terminal operations has received increasing interest in the scientific literature over the last years and large-scale optimization methods are the most appropriate tool to tackle the complexity of decision problems involved in maritime transport and logistics.

The research streams addressed by this dissertation are complementary:

- in Part I we move from the specific application of container terminals to solution methods that aim to be generalized and extended to other large-scale optimization problems; in particular, we plan to identify and model relevant decision problems in container terminal management and to provide effective solution algorithms able to impact on terminal's productivity and efficiency;

- in Part II we move from general methods for large-scale optimization towards their application to transportation and logistic problems arising in a real-world context; in particular, we plan to develop advanced solution techniques for

large-scale optimization problems, able to overcome the current issues of state-of-the-art column generation and branch-and-price schemes.

1.1 Motivation

The resolution of large scale problems has improved over the last decades thanks to the advances in combinatorial optimization theory (Nemhauser and Wolsey, 1988). In particular, column generation (Lübbecke and Desrosiers, 2005; Lübbecke, 2010) has been intensively used to compute good quality lower bounds for combinatorial problems reformulated through Dantzig-Wolfe decomposition (Dantzig and Wolfe, 1960). The original problem is decomposed into a number of pricing subproblems, that are solved independently and coordinated at a higher level by a master problem.

Column generation has been traditionally embedded in a branch-and-bound scheme, called branch-and-price, to solve large-scale integer programs (Barnhart et al., 1998; Desaulniers et al., 2005). Branch-and-price currently represents one of the most advanced and efficient exact optimization approaches to solve large-scale combinatorial problems. However, their successful implementation requires a very good knowledge of the problem structure and the development of sophisticated tailor-made accelerating techniques. Furthermore, the increasing size and complexity of practical problems arising in real-world applications motivates the design of solution approaches able to overcome current issues that affect column generation, such as instability, lack of dual information as well as master and pricing problems of unmanageable size. Recent attempts in this direction include stabilization methods (du Merle et al., 1999; Ben Amor, 2002; Briant et al., 2008), dynamic aggregation of constraints (Elhallaoui et al., 2005; Elhallaoui et al., 2008; Elhallaoui et al., 2010) and variable elimination techniques (Irnich et al., 2010).

Transportation and logistics are a major source of complex optimization problems. Air transport and the airline industry have greatly benefited from operations research (OR) methods since the 1950s (Klabjan, 2005), and the models have become more and more complex, mainly because of the increasing problem size and accuracy. Maritime transport and seaport logistics represent a more recent OR research field, that has been mainly pushed forward by the dramatic and rapid growth of containerization over the last decade and, more in general, of sea-freight transportation (UNCTAD, 2009).

The need for an efficient management of logistic activities at modern container terminals is well recognized and there exists a rich literature of optimization models and algorithms conceived for specific operational problems (Steenken et al., 2004; Stahlbock and Voss, 2008). In particular, current research directions in container terminal management are pointing towards integrated planning of operations, as already occurred in the airline sector (Lohatepanont and Barnhart, 2004; Sandhu and Klabjan, 2007). This yields to significant improvements in terms of efficiency and productivity for the terminal; however, from a mathematical point of view, the resulting integrated problems are very complex and therefore, advanced techniques for

solving such difficult large-scale optimization problems need to be designed in order to cope with this complexity.

Along with specialized algorithms targeted to specific applications, it is equally important to pursue fundamental research and to make progress in general methods and algorithms for large scale optimization problems. In particular, the transferability of algorithms across applications and the need of general results represent a major challenge that motivates this research stream.

1.2 Research objectives

The objectives of this dissertation are twofold and our research work is organized in two parallel and complementary streams.

From Applications to Methods We contribute in bridging the gap between the different research challenges arising in container terminal management, with a focus on integrated solution approaches. In particular, we study new models for the integrated planning of berth allocation and quay crane assignment, addressing the problem at the tactical decision level. The overall objective is to improve terminal efficiency and provide decisional tools with a significant added value for terminal managers. Furthermore, we plan to take into account yard congestion issues related to the transshipment flow of containers.

From a modeling point of view, this involves overcoming the limiting assumptions of existing approaches and providing a more realistic problem definition that includes operational constraints, such as unwritten rules and policies and/or best practices, in an aggregated fashion.

From an algorithmic point of view, this requires to design efficient solution methods for the proposed models. We consider both heuristic and exact methods: the heuristic approach is meant to provide good solutions with the minimum computational effort; the exact approach, based on column generation and branch-and-price, aims to provide good dual bounds and optimal solutions to the problem.

A broader objective of this research stream is to provide solution methods that are specifically conceived for our application in container terminals, but that can be further generalized and applied to other domains.

From Methods to Applications We develop new methods and algorithms for generic large-scale optimization problems: in this context, a new column generation framework is investigated.

To this purpose, we identify a new class of split delivery vehicle routing problems with specific features that allow us to generalize several applications in transportation and other domains. In particular, vehicle routing problems are widely

studied in the literature and can be considered a well accepted benchmark to validate and test new methodologies.

Our main objective is to propose a new concept in column generation theory that is transferable across applications. We consider the global column generation process with a particular focus on the relationship between compact and extensive formulation as defined by Dantzig-Wolfe decomposition.

The new framework is designed to tackle complex large scale optimization problems and aims to reduce the overall computational effort by providing a more efficient way of handling master and pricing problems. In particular, we focus on problems that present a large number of compact formulation variables and that are unmanageable with standard column generation methods.

Concerning algorithmic issues, we are aware that in many applications, and especially in vehicle routing problems, the pricing represents a major bottleneck to the overall efficiency of the algorithm; therefore, particular attention is given to this aspect.

1.3 Contributions

In this section we summarize the main contributions of this dissertation.

Part I: from applications to methods.

- We propose a new model for the integrated optimization of berth allocation and quay crane assignment, called the Tactical Berth Allocation Problem (TBAP). We provide a mixed integer quadratic programming (MIQP) formulation and a mixed integer linear programming (MILP) formulation for the integrated problem.

  The key feature of our model is the inclusion of a quay crane profile, a decision variable that represents the number of quay cranes available to a berthed vessel at each time step. We define the concept of QC profile in order to capture real-world issues; furthermore, the new concept works well to represent the control that the terminal has on several aspects of QC assignment during the optimization process; in particular, it enables us to overcome the limits of existing models.

  We also take into account traffic and congestion issues originated in the yard by the transshipment flows. In particular, our model aims to minimize relocations of containers within the yard: we are able to reduce costs with respect to transfer equipment and traveled distance, but also to reduce traffic and thus congestion within the yard.

  We address the tactical problem in order to support the terminal in its negotiation with shipping companies. During this process, terminal managers must

be able to evaluate the impact that a certain assignment of resources, such as berths and cranes, has on the terminal performance.

Computational tests confirm the added value of integration in terms of cost reduction and efficient use of resources. We provide an experimental comparison between the traditional hierarchical approach solving sequentially berth allocation and quay crane assignment, and our proposed integrated TBAP model. The main outcome of the analysis is that the strong assumptions made by the sequential approach may prevent to find any feasible solution, whereas the integrated approach always finds the optimal one. This occurs especially for congested instances. Furthermore, the additional effort required to solve the integrated problem is moderate.

- We present a new heuristic and a new exact method for solving the TBAP.

  Firstly, we propose a specialized heuristic algorithm organized in two stages: for a given QC profile assignment, we solve the resulting berth allocation plan using tabu search; the QC profile assignment is then updated using reduced cost information, and the procedure is iterated. Our heuristic is able to solve very large instances in a reasonable time: the proposed method clearly outperforms commercial solvers and provides good-quality solutions.

  Secondly, we propose an exact branch-and-price algorithm with the purpose of proving good-quality bounds and optimal solutions to the problem. We propose a specific branching scheme and several accelerating techniques both for the pricing and the master problem. In addition to state-of-the-art techniques, such as bidirectional dynamic programming and dual stabilization, we present advanced techniques specifically conceived for our problem. Some of these techniques proved to be very useful and can be easily generalized for other branch-and-price schemes.

  Computational tests prove that our exact algorithm outperforms commercial solvers: especially on small instances, branch-and-price always provides optimal solutions relatively fast. Results on larger instances are promising, although the increased complexity is not completely overcome. However, the improved linear relaxation bound provided by the Dantzig-Wolfe reformulation of the problem confirms that our heuristic algorithm produces very good-quality solutions.

- An additional key point of our research is represented by the fact that all the validation phase is conducted on real data provided by MCT container terminal in the port of Gioia Tauro, Italy. Thanks to this collaboration, we are able to test our methodologies and algorithms on real data and in a real-world context. Therefore, the findings of this work aim to have a relevant impact on real-world applications.

Part II: from methods to applications.

- We introduce a new class of vehicle routing problems with split deliveries that generalizes some interesting modeling features of TBAP rarely treated in the literature: (i) discrete demand delivered in discrete orders (while the usual approach is to model continuous demand that can be delivered in any fraction) and (ii) service time dependent on the delivered quantity (while the usual approach is to assume a constant service time, regardless of the quantity delivered to the customer).

- We propose a new model for the Discrete Split Delivery Vehicle Routing Problem with Time Windows (DSDVRPTW). The problem definition originates from a generalization of TBAP into a more general VRP class of problems with discrete demands that includes applications in logistics and telecommunication.

  From a modeling point of view, the introduction of quantity-dependent service time overcomes the usual assumption of constant service time, that is not always realistic.

  We study the properties of the new problem and we remark that the introduction of quantity-dependent service times modifies some known properties of split delivery vehicle routing problems that are usually exploited in specialized solution algorithms.

  We present a flow-based mixed integer program for the DSDVRPTW, we reformulate it via Dantzig-Wolfe and we apply column generation. We propose a branch-and-price algorithm that presents extensions of state-of-the-art techniques in order to cope with the additional complexity of quantity-dependent service times. Computational results on instances based on Solomon's dataset confirm that our implementation outperforms commercial solvers by several order of magnitude and by number of solved instances.

  Furthermore, we experimentally compare constant service time vs quantity-dependent service time with respect to complexity, in order to support the importance of the new modeling feature; in particular we show that additional complexity comes with potential savings.

- We present a novel framework called Two-stage column generation, specifically conceived to tackle complex large-scale optimization problems that cannot be efficiently solved by standard column generation.

  In the context of Dantzig-Wolfe (DW) decomposition, we basically propose to simultaneously generate "columns" both for the compact and the extensive formulation. The approach is particularly suited for those problems where the large number of variables in the compact formulation directly affects the pricing problem and its efficiency.

We focus our attention on the relationship between compact and extensive formulation. The key point of our method is that we evaluate the contribution of compact formulation variables with respect to the extensive formulation, in order to take advantage of the constraints that have been "convexified" in the reformulation: indeed, we only add compact formulation variables that are profitable for the master problem, regardless of the optimal solution of the linear relaxation of the compact formulation.

We provide a formal description of the new framework and an example based on the Resource Constrained Shortest Path Problem to illustrate how two-stage column generation basically works.

We apply the proposed methodology to the Discrete Split Delivery Vehicle Routing Problem with Time Windows and we present an extensive computational campaign that validates our new framework. Furthermore, we give an outline of how Two-stage column generation could be applied to the Tactical Berth Allocation Problem and we discuss some major issues.

Computational results show that two-stage column generation significantly reduces the number of generated columns to prove optimality of the root node with respect to standard column generation. Suboptimal compact formulation variables are detected correctly and a large percentage of variables do not need to be taken into account during the solution process. However, the additional effort required by our sophisticated approach makes the method competitive in terms of computational time only for instances of a certain difficulty.

1.4 Structure

This dissertation is organized in two main parts.

Part I is devoted to models and algorithms for container terminal management, with a specific focus on the application.

Chapter 2 provides an introduction to maritime logistics and container terminals. The main operations and processes are described and relevant operations research literature is reviewed. A discussion on recent trends in container terminal optimization is also provided.

Part of this chapter is based on:

Vacca, Bierlaire and Salani (2007). Optimization at Container Terminals: Status, Trends and Perspectives. Proceedings of the Swiss Transport Research Conference (STRC) September 12-14, 2007.

Chapter 3 introduces the Tactical Berth Allocation Problem. Two formulations are presented: a mixed integer quadratic program and a linearization that reduces to

a mixed integer program. A heuristic algorithm that combines tabu search and mathematical programming techniques has been designed to solve the problem. The proposed method is shown to significantly outperform commercial solvers. Computational tests are carried out on realistic instances based on the MCT, port of Gioia Tauro, Italy.

This chapter has been published as:

Giallombardo, Moccia, Salani and Vacca (2010). Modeling and solving the tactical berth allocation problem, Transportation Research Part B: Methodological 44(2): 232-245.

Ranked 16th in the TOP 25 hottest articles of Transportation Research Part B for January-March 2010.

Chapter 4 proposes an exact algorithm for the TBAP based on column generation. After a brief introduction to branch-and-price methods, we reformulate the TBAP via Dantzig-Wolfe decomposition: the extensive formulation, as well as the master and the pricing problem are derived. The pricing subproblem is a Resource Constrained Elementary Shortest Path Problem and it is solved by dynamic programming. The implementation of the branch-and-price algorithm is discussed and computational results are presented. Our exact approach significantly outperforms commercial solvers and is very efficient especially on small size instances. A comparative analysis between the traditional hierarchical solution approach and the integrated TBAP concludes the chapter.

Preliminary results have been presented and published as:

Vacca, Salani and Bierlaire (2010b). Recursive column generation for the Tactical Berth Allocation Problem, Proceedings of the 7th Triennial Symposium on Transportation Analysis (TRISTAN VII), Tromso, Norway.

Vacca, Salani and Bierlaire (2010a). Optimization of operations in container terminals: hierarchical vs integrated approaches. Proceedings of the 10th Swiss Transport Research Conference (STRC), Monte Verità, Ascona, Switzerland.

Part II is devoted to general models and algorithms for large-scale optimization problems, with a specific focus on methods.

We propose a new concept in column generation for handling complex large scale optimization problems, as those identified in Part I. The main objective is to design a framework that is transferable across applications and to provide general results. The proposed methodology is validated on a class of vehicle routing problems that generalizes specific features of the Tactical Berth Allocation Problem and other applications in transportation, telecommunication and logistics.

Chapter 5 introduces the Vehicle Routing Problem with Discrete Split Deliveries and Time Windows as a generalization of TBAP. A mixed integer program

based on arc-flow formulation is presented; the problem is reformulated using Dantzig-Wolfe decomposition and a branch-and-price algorithm is implemented to solve the problem. Computational results on instances based on Solomons data set are presented and discussed. Furthermore, we analyze the impact of quantity-dependent service time on the resulting solutions.

This chapter is mainly based on:

Salani and Vacca (2009). Branch and Price for the Vehicle Routing Problem with Discrete Split Deliveries and Time Windows. Technical report TRANSP-OR 091224. Transport and Mobility Laboratory, École Polytechnique Fédérale de Lausanne, Switzerland.

submitted to the European Journal of Operational Research, currently under 3rd revision for possible publication.

Chapter 6 introduces a new framework called Two-stage column generation. A formal description is provided and major theoretical issues are discussed. An example based on the Resource Constrained Shortest Path Problem illustrates the overall methodology. The two-stage scheme is applied to the Discrete Split Delivery Vehicle Routing Problem and extensive computational results are provided. Furthermore, the application of the method to the Tactical Berth Allocation Problem is outlined and further research directions are discussed.

The chapter is mainly based on:

Salani, Vacca and Bierlaire (2010). Two-stage column generation. Technical report TRANSP-OR 101130. Transport and Mobility Laboratory, École Polytechnique Fédérale de Lausanne, Switzerland.

Finally, **Chapter 7** provides conclusions and future research perspectives.

Detailed computational results are provided in **Appendix A** for the Discrete Split Delivery Vehicle Routing Problem with Time Windows and in **Appendix B** for Two-stage column generation.

Part I

From Applications to Methods

Part I of this dissertation is devoted to models and algorithms for container terminal management, with a specific focus on the application.

We introduce the context of maritime transportation and logistics and we provide an overview of container terminal operations. A discussion on current research challenges in container terminal management identifies promising research directions, that include integrated planning of operations, analysis of congestion and tactical aspects of the decision making process.

We contribute in bridging the gap between the different research challenges by proposing the integrated planning of berth allocation and quay crane assignment, addressing the problem at the tactical decision level.

We present models and algorithms that are specifically conceived for this application, with the broader objective of providing methods that can be further generalized to solve large scale optimization problems in other domains.

Chapter 2

Container terminals

2.1 Introduction

Containerized sea-freight transportation has grown dramatically over the last two decades, much faster than other sea transportation modes. Container traffic increased about 9.5% per year between 2000 and 2008, while the average annual rate for cargo traffic was 5.3% (ISL, 2009). The share of containerized trade in the world's total dry cargo increased from 5.1% in 1980 to 25.4% in 2008 (UNCTAD, 2009).

This rapid growth is explained by several factors, such as reduced transit time, reduced shipping costs, increased reliability and security, multi-modality. Containers are nowadays the main type of equipment used in intermodal transport: any container has a standardized load unit that is suitable for ships, trucks and trains and can be transferred very quickly from one transport mode to another. In this context, container terminals are crucial connections between different transportation modes and cargo handling represents a critical point in the transportation chain. Moreover, they also represent the site where several market players involved in maritime transportation (such as the terminal itself, the port authority, the shipping companies) trade for their business.

Due to the dramatic increase of container traffic, terminals are reaching their capacity limits, leading to traffic and port congestion. In particular, the costs associated with port congestion affect not only the terminal (extra manpower, yard congestion, re-handling) but also the shipping lines (ship delays, missed connections, extra-costs) and the other market players.

One solution to congestion is to increase capacity with bigger yards and new equipments, although this way is often precluded due to lack of physical space and budget constraints. Furthermore, whenever the port can afford the additional investment, it may take years to make the new infrastructure operational. Therefore, improvements in port efficiency and productivity are nowadays more and more needed and effective operational systems can significantly help to make the best use of port infrastructure and resources.

Terminal managers should take into primary consideration the interests of actors that are most critically involved in container transportation (Vanelslander, 2005). Competition and competitiveness are strong in the current shipping market (Notteboom, 2004; Tongzon and Heng, 2005): besides competing with terminals in other ports, terminal managers are faced against competition issues even among terminals of the same port. The biggest ports in the world often consist of several terminals: just to mention a few examples, the port of Hong Kong consists of 9 container terminals operated by 5 companies and the port of Hamburg has 4 dedicated container terminals and 8 multi-purpose terminals able to handle containers. Competitiveness is therefore a crucial issue to survive in the market, as a shipping company that decides to serve a certain port with regular services has the possibility to choose among several terminals the most appropriate for its business; typical performance indicators in container terminals are the vessel turn-around time, the crane productivity and the total throughput.

The need for an efficient management of logistic activities at modern container terminals is well recognized and the management of container terminal operations can greatly benefit from operations research methods.

In the remainder of this chapter we provide a brief description of operations and decision problems in container terminals (section 2.2) and we review the most significant literature in the field (section 2.3). In section 2.4 we discuss the status of research, with a particular focus on recent promising trends.

2.2 Operations and decision problems

A container terminal is the zone of the port where vessels dock on a berth and containers are loaded, unloaded and stored in a buffer area called yard. The terminal can be ideally divided into three areas: the quayside, the yard and the gate (Figure 2.1). A real container terminal (CTA Hamburg) is illustrated in Figure 2.2, where we can clearly identify the quayside in the upper part of the picture, the yard in the middle part and the gate in the bottom part.

The *quayside* is made up of berthing positions along the quay and quay cranes (QCs) that load/unload containers from vessels calling at the berth (Figure 2.3). Containers are commonly transferred to the yard by automatic guided vehicles (AGVs, Figure 2.4), straddle carriers (SCs, Figure 2.5) or internal trucks. The transportation equipment is also used to move containers from yard to gate and, when needed, to relocate containers within the storage area.

The *yard* serves as a buffer for loading, unloading and transshipping containers and it is typically divided into blocks: each container block is served by one or more yard cranes, such as rubber-tired or rail-mounted gantry cranes (RTG/RMG), illustrated in Figure 2.6. Another possible yard configuration is composed of lanes served by straddle carriers (Figure 2.7). The equipment used to operate the yard makes the difference between an intensive and extensive yard utilization: intensive

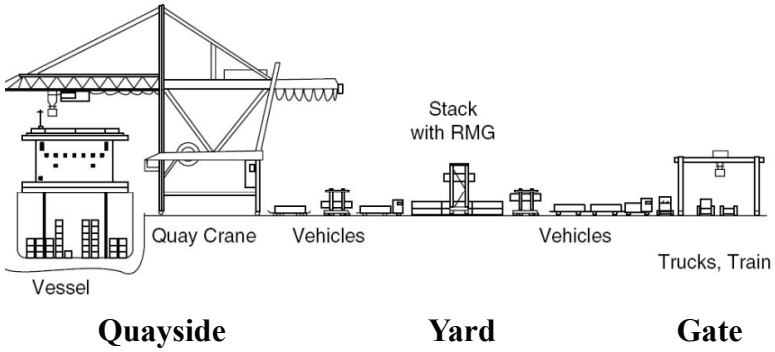

Figure 2.1: *Schematic representation of a container terminal (Steenken et al., 2004).*

yard terminals require a high storage capacity and are mainly operated by RMGs or RTGs, that can store approximately 1000-1100 TEU per hectare when container stacks are 4-5 levels high (they can stack up to 8); extensive yard terminals require a lower storage capacity and are typically operated by straddle carriers, that can store approximately 500-750 TEU per hectare when container stacks are 2-3 levels high (KALMAR, 2010).

Figure 2.2: *The Container Terminal Altenwerder (CTA) in Hamburg, Germany.*

Figure 2.3: *Containerships and quay cranes (QCs).*

Figure 2.4: *AGVs.*

Figure 2.5: *Straddle carriers (SCs).*

Figure 2.6: *Yard operated by RTGs.*

Figure 2.7: *Yard operated by SCs.*

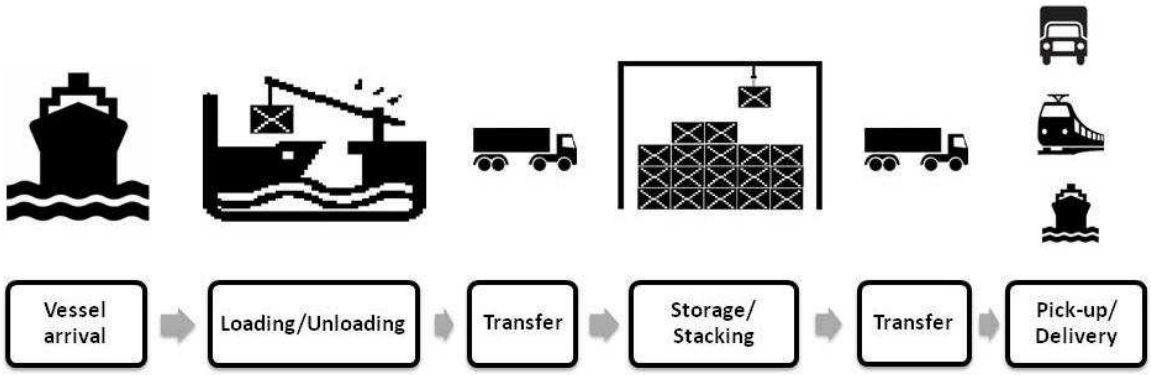

Figure 2.8: *The main processes at a container terminal.*

In import/export terminals, the flow of containers continues inland and containers are picked-up and delivered by trucks and trains in a area called *gate*. Gate congestion represents a critical issue and it limits the efficiency of the whole intermodal logistics system; the recognized direct reason is often the unmanaged container truck arrivals.

Recently, container transport has evolved towards a particular case of single-mode transportation, called *transshipment*, mainly motivated by the reduction of transportation costs: shippers seek to increase economies of scale, building ever larger container ships for long-haul routes, and demanding terminals with facilities and technologies able to handle them (mega-terminals). The resulting system is known as *hub and spoke* and is analogous to the way airlines route their traffic: deep sea containerships (*mother vessels*) operate among a limited number of transhipment terminals (*hubs*), and smaller vessels (*feeders*) link the hubs with the other ports (*spokes*). In this context, many of the multi-modality issues typical of import/export terminals are concentrated within the terminal along the quayside. In particular, congestion issues raise when mother vessels and feeders are performing simultaneously loading and unloading operations.

The typical processes of a container terminal are illustrated in Figure 2.8 and refer to the flow of import containers; all the processes can be executed in the reverse order, when export containers are loaded onto a ship.

Container terminal operations can be grouped in four main classes, that are associated with specific processes and stages in the container flow (Vis and de Koster, 2003). We mainly focus on the transshipment flow of containers and the associated decision problems that usually originate between the quayside and the yard.

Berth allocation and scheduling These decisions are associated with the *vessel arrival*. The berth allocation problem (BAP) consists of assigning and scheduling ships to berths (discrete case) or to quay locations (continuous case) over a given time horizon. A schematic representation of a berth plan for 5 vessels is provided in Figure 2.9(a): vessels are scheduled over time (x axis) according to their expected

handling time and assigned to different berthing positions on a quay of 600 meters (y axis) according to their length. Additional constraints usually include vessel's draft, time windows on the arrival/departure time of vessels, priority ranking, favorite berthing areas.

Quay crane allocation and scheduling These decisions are associated with the *loading and unloading operations*. An efficient use of quay cranes is crucial, since quay cranes are highly expensive and represent one of the most scarce resources in the terminal. The quay crane allocation problem (QCAP) aims to efficiently assign quay cranes to vessels that must be operated over a given time horizon. The allocated cranes must be sufficient to complete the workload within the given time window, although many configurations are possible. The loss of productivity due to crane interference is also taken into account. The quay crane scheduling problem (QCSP) is more operational: planners must assign specific quay cranes to specific tasks (set of containers) and produce a detailed schedule of the loading and unloading moves for each quay crane. Issues related to interference among cranes, precedence and operational constraints, such as no overlapping, are also taken into account.

Figure 2.9(b) represents a schematic representation of a quay crane assignment and scheduling plan, built on the berth allocation plan sketched in (a). For every time step, a certain number of cranes is allocated to vessels; the number of cranes may vary (as for vessels 2 and 3) or remain constant (as for vessels 1, 4 and 5) during vessel's stay-at-the-port. Furthermore, not only the amount but specific cranes (identified by an index) are assigned to vessels; finally, the assignment must not exceed the total qc capacity, that is of 4 quay cranes in the example.

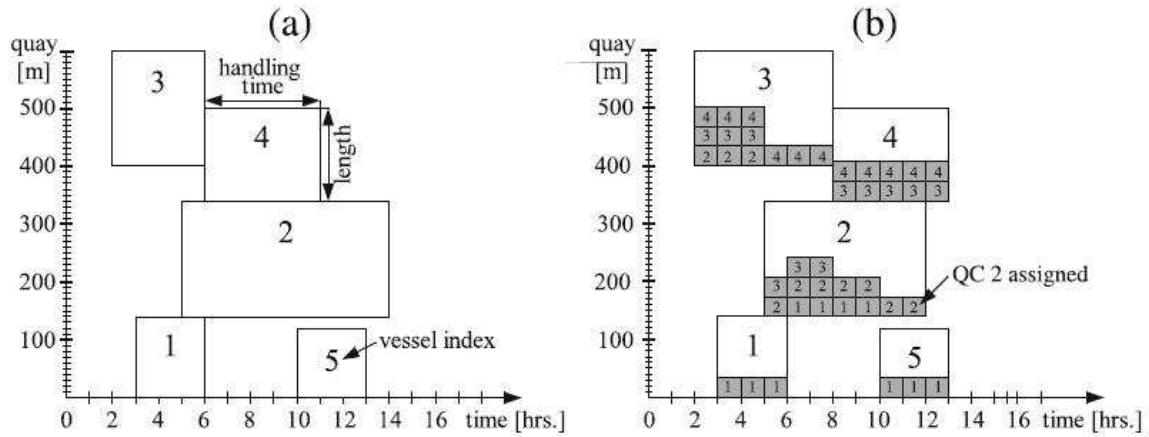

Figure 2.9: *Space-time representation of a berth plan (a) and quay crane assignment (b) (Bierwirth and Meisel, 2010).*

Transfer Operations Containers are usually transferred inside the terminal by internal trucks, straddle carriers and automated guided vehicles. The transfer originates decision problems such as vehicle routing and dispatching strategies. Typical objectives aim to minimize the vehicle fleet size, the total distance traveled to complete the tasks, the fleet operating costs or the total operations delay. Some optimization strategies may also include deadlock prevention and real-time conflict avoidance for automated guided vehicles. We distinguish between quayside transfer operations (from quay to yard, from yard to quay) and landside transfer operations (from yard to gate, from gate to yard).

Yard operations These decisions are associated with container *storage and stacking*. The management of yard operations involves several decision problems. The yard allocation problem refers to the design of storage policies at the block level according to the specific features of the container (size, weight, destination, import/export etc.). At the operational level, the plan becomes more detailed and a specific position in the block (identified by the row, bay and tier) is assigned to each container. Another interesting problem is the yard crane deployment, that concerns the allocation of cranes to blocks, their routing and the scheduling of the tasks based on the container loading sequence. Finally, advanced methods for container stacking are investigated, such as the design of re-marshalling policies for export containers. In fact, the container retrieval process of export containers can be optimized via relocation and rehandling strategies performed in advance, in order to speed up the loading operations; this policy is also referred to as *housekeeping*.

So far, we have mostly mentioned operational decision problems arising in container terminal management. More generally, decisions mainly concern *planning* and *controlling*, and depending on the time frame we can distinguish three decision levels, illustrated in Figure 2.10. The *strategic level* involves long-term decisions regarding terminal layout and infrastructure; the time horizon usually covers several years. Typical decisions are terminal location, terminal equipment, terminal infrastructure, multi-modal interfaces as well as strategic alliances with shipping companies. The *tactical level* involves mid-term and short-term decisions regarding the allocation of resources, such as berth and yard templates, storage policies, human resources management, etc. At this level, the experience of planners plays an important role in decision making, that is currently based in large part on "rules of thumb". The *operational level* involves daily and real-time decisions concerning both the quayside (e.g. quay cranes scheduling) and the landside (e.g. AGV routing, yard crane deployment). Additional decision problems, not mentioned so far, concern the ship stowage planning and the repositioning of empty containers.

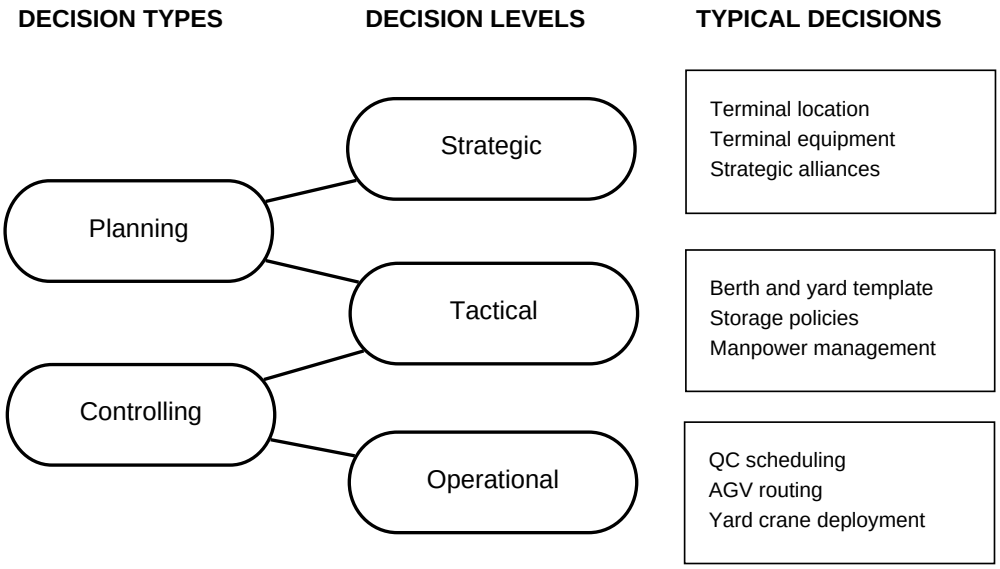

Figure 2.10: *Decision levels and operations in container terminal management.*

2.3 Literature review

Container terminal operations and maritime logistics have received increasing interest in the operations research (OR) community over the last years. In this section we review relevant contributions to the main container terminal operations. The list is not exhaustive, but we rather cite the papers that had a major impact in the research community. Specific literature will be further discussed in every chapter, according to the studied problem and proposed methodology.

The first survey on container terminal management is by Vis and de Koster (2003), where authors illustrate the main logistic processes in a container terminal reporting about 50 references up to 2001. Steenken et al. (2004) present an exhaustive overview of optimization methods in container terminal management, reviewing more than 200 references up to 2004, and also provide a detailed description of terminal structure and handling equipment. This survey has been recently updated and extended with the state-of-the-art (Stahlbock and Voss, 2008). Christiansen et al. (2004) review operations research methods in the more general field of maritime logistics, focusing on ship routing and scheduling, while Crainic and Kim (2007) provide a general discussion on intermodal freight transportation and container-based systems.

In the past years, research focused on very specific problems and there exists many contributions dedicated to sophisticated models for single operational problems at container terminals.

The *berth allocation problem* was firstly modeled as a discrete problem by Imai et al. (1997): the authors represent the quay as a finite set of berths and propose a multi-objective approach to solve the problem. Lim (1998) introduces the continuous BAP and proves that it is NP-Hard; the problem is further investigated by Imai

et al. (2005). Both papers propose a heuristic method to solve the problem. The dynamic arrival of vessels is studied by Imai et al. (2001), that propose a heuristic solution approach based on Lagrangian relaxation; the model is further extended (Imai et al., 2003) to consider service priorities and solved by a genetic algorithm. Cordeau et al. (2005) model the discrete BAP as a multi-depot vehicle routing problem with time windows and solve the problem with a tabu search metaheuristic; their model is further solved by Mauri et al. (2008) using a column-generation-based heuristic algorithm. Recently, Buhrkal et al. (2009) have presented a generalized set-partitioning model for the discrete BAP where all columns are enumerated a-priori: the formulation outperforms existing models and guarantees optimality. However, authors recognize that a branch-and-price algorithm should be implemented in order to solve larger instances.

The *quay crane scheduling problem* was introduced by Daganzo (1989) as a mixed integer programming model and solved by Peterkofsky and Daganzo (1990) through a branch-and-bound algorithm. Kim and Park (2004) introduce in the model precedence relationships between tasks and non-interference constraints between cranes; solution algorithms proposed for this advanced model include branch-and-cut (Moccia et al., 2006), tabu search (Sammarra et al., 2007) and genetic algorithms (Lee et al., 2008). Recently, Bierwirth and Meisel (2009) have relaxed some limiting assumptions of previous models by introducing a new set of constraints for crane interference; the problem is solved by a heuristic branch-and-bound algorithm. Finally, crane-double cycling and its effects on loading and unloading operations have been investigated by Goodchild and Daganzo (2006; 2007).

The literature devoted to *transfer operations* can be distinguished based on the means of transport, mainly AGVs and straddle carriers. Kim and Bae (2004) propose a model for dispatching AGVs solved by a heuristic algorithm: their main objective is to reduce delay in ship operations. Liu et al. (2004) analyze the effect of different yard configurations on AGVs deployment: the authors propose a simulation model to compare the two terminals and results show that yard layout has an effect on the number of AGVs used and on their performance. The issue of deadlocks in AGV systems is discussed by Moorthy et al. (2003) and Kim et al. (2007), who propose efficient strategies for predicting and avoiding deadlocks in large-scale systems; the proposed methods are validated through simulation. Cheng et al. (2005) present a network flow based model for deadlock prediction and prevention, while Möhring et al. (2005) study the real-time problem: a method based on shortest paths with time windows and re-adjustment proves to be efficient and to outperform the static routing approach. The routing of straddle carriers is discussed by Steenken et al. (1993), with a particular focus on internal moves: the objective is to reduce the no-load traveled distance and simulation results are provided. Kim and Kim (1999) study the routing of a single straddle carrier and propose a heuristic beam-search solution approach. Recently, Ndiaye et al. (2008) have proposed a reformulation based on DC (Difference of Convex functions) programming, solved by cutting plane techniques.

Research on *yard operations* concerns space allocation, stacking and retrieval poli-

cies and yard crane deployment. The seminal papers by de Castilho and Daganzo (1993) and Taleb-Ibrahimi et al. (1993) analyze container handling strategies in terminals operated by yard cranes. The storage space allocation problem is further studied by Zhang et al. (2003), that propose a two-level solution approach based on mathematical programming. Dekker et al. (2006) compare different policies for stacking containers in the yard using simulation, in order to provide a decision support system to automated terminals. Lee et al. (2006) study the specific case of yard management in transshipment hubs: a mixed integer programming model is provided and solved by heuristic procedures. Kim et al. (2000) investigate location rules for export containers by using dynamic programming; a decision tree on the set of optimal solutions is also developed to support real-time decisions. Kang et al. (2006) propose an algorithm based on simulated annealing to derive good stacking strategies when information on container's weight is uncertain. Container reshuffling strategies are investigated by Kim and Bae (1998), Lee and Hsu (2007) and Yang and Kim (2006): all papers suggest a solution approach based on heuristic algorithms. Several optimization methods have been proposed for yard crane deployment: Zhang et al. (2002) solve the problem by Lagrangian relaxation, while Linn and Zhang (2003) develop a least-cost heuristic. Simulation studies are provided by Kim et al. (2003; 2006). Ng and Mak (2005) model the yard crane scheduling problem as a mixed integer program; a branch-and-bound algorithm solves the problem exactly. Crane interference is further considered by Ng (2005): the resulting model is solved by a a dynamic programming based heuristic.

Finally, *simulation* models and methods have been used to study the terminal as a global system and to analyze the entire flow of containers. Gambardella et al. (1998) present a decision support system for resource allocation in intermodal container terminals: solutions provided by the optimization module are validated via discrete event simulation. The scheduling of loading and unloading operations is further included by Gambardella et al. (2001). Legato and Mazza (2001) present a queuing network model for the arrival, berthing and departure of vessels; authors develop a simulation tool that provides a practical support for decision making. The same approach is adopted by Canonaco et al. (2008) for the management of berth crane operations. Recently, Legato et al. (2010) have proposed an optimization-based simulation approach for loading and unloading operations, that relies on simulated annealing and discrete event simulation. Finally, multi-agent based simulation for evaluating the management of container terminal operations has been proposed by Henesey (2006).

From the reviewed contributions we can conclude that decision problems in container terminals are very complex. Most solution approaches rely on heuristic methods to provide fast solutions; furthermore, assumptions and relaxations are made to maintain the problem tractable. However, exact approaches for these large scale optimization problems should be further investigated.

2.4 Research trends

In this section we discuss what are, in our opinion, the current trends and challenges in container terminal management.

2.4.1 Integration of operations

A promising research track is represented by the integrated optimization of decision problems that are highly interdependent, yet usually solved hierarchically by terminal's planners. With respect to quayside operations, a recent survey by Bierwirth and Meisel (2010) reviews contributions on integrated solution approaches for the berth allocation problem and quay crane scheduling problem; with respect to landside operations, transfer and storage planning are two important problems affecting the efficiency of the operations and a few contributions investigate the integration of yard truck scheduling and storage allocation (Bish et al., 2001; Bish, 2003; Lee et al., 2009).

In particular, the simultaneous optimization of berth allocation and quay crane assignment is a critical issue in terminal management, since the two problems are strictly correlated. An integrated solution approach is therefore more appropriate and should be preferred to the traditional sequential decision-making process illustrated in Figure 2.11. The integrated planning of BAP and QCAP, introduced by Park and Kim (2003), has been recently investigated by Imai et al. (2008) and Meisel and Bierwirth (2009). The resulting models are a good starting point for tackling such a complex problem; however, they still present some unrealistic assumptions and limits. In fact, the relationship between the number of quay cranes and the handling time is ignored (Imai et al., 2008) or the crane productivity is assumed to be proportional to the number of QCs (Park and Kim, 2003), although it is known that

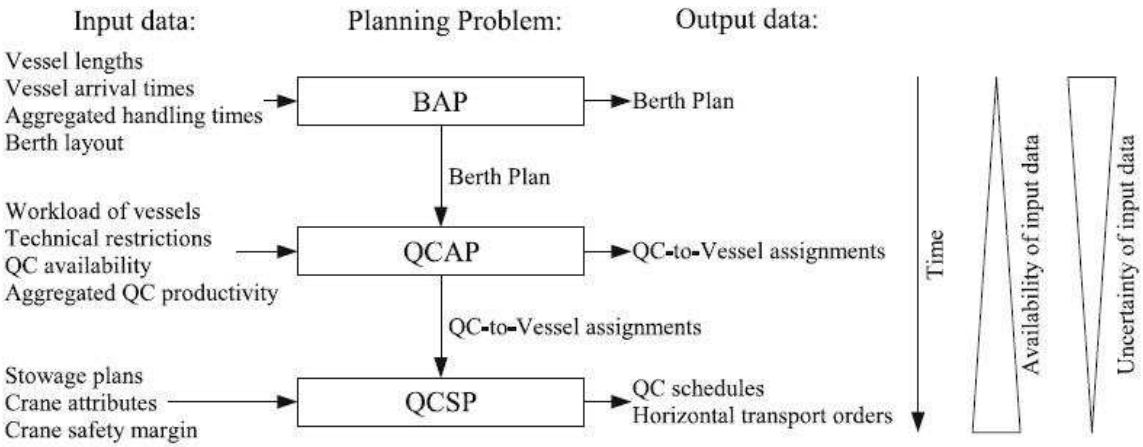

Figure 2.11: *Sequential planning of quayside operations (Bierwirth and Meisel, 2010).*

quay cranes interference reduces the marginal productivity. Also, no hard constraints are imposed on berthing times. Furthermore, in all the approaches the quay crane assignment is made hour by hour, without any control on the final outcome, and this may result in plans not acceptable or applicable in practice. In particular, existing models cannot incorporate specific requirements and "unwritten rules" that terminal planners usually control: for instance, cranes should be moved only at the end of the shift because of manpower-related issues; also, for a given vessel, the number of assigned quay cranes should not vary too much during its stay-at-the-port; a break in the service during operations, i.e., zero quay cranes assigned to a vessel, may be a solution suggested by the existing models that perform hourly assignment of cranes, although it would be unacceptable in practice. Therefore, we think that a good integrated model should overcome these unrealistic assumptions.

2.4.2 Tactical decision level

In the current practice, many tactical problems are still solved by rules of thumb by terminal planners: it is therefore likely that improvements in terms of efficiency and productivity can be reached by using more quantitative solution approaches at the tactical decision level. Furthermore, taking into account operational constraints (in terms of rules, common policies and best practices) in the definition of the tactical problems would allow to introduce the concept of robustness in the tactical planning.

Most papers in the literature study operational problems, whereas only a few contributions on tactical planning are available. Moorthy and Teo (2006) address the design of a berth template, a tactical planning problem that arises in transshipment hubs and that concerns the allocation of favorite berthing locations (home berths) to services periodically calling at the terminal. The authors propose two procedures able to build good and robust templates and evaluate their performance via simulation. Cordeau et al. (2007) introduce the service allocation problem, a yard-related decision problem that occurs at the tactical planning level. The authors provide a berth and yard template able to minimize the container rehandling operations in the context of a transshipment container terminal.

Although both papers deal with tactical problems, the interaction and the negotiation aspects with market players, especially with shipping companies, that are typical of this decision level are not taken into account. Furthermore, no contributions address the tactical level for integrated decision problems.

2.4.3 Congestion and traffic

Congestion issues in container terminals are becoming more and more relevant, especially because of the volume increase in container traffic. Although congestion is often disregarded in the planning, operations are usually slowed down because of overloaded areas, such as the yard, and congestion rapidly spreads to the whole system.

A few papers have investigated this aspect in the very recent past. Lau and Lee (2007) analyze the quayside traffic condition of a very busy container port in Hong Kong. The authors use simulation to study the effects of deployment strategies for trucks on traffic congestion and quay crane utilization and it results that a traffic reduction also reduces the performance of quay cranes. Han et al. (2008) focus on transshipment hubs, where loading and unloading operations are highly concentrated and yard activity is heavy: in order to minimize traffic congestion, the authors propose a storage strategy using a high-low workload balancing protocol.

However, an analytical description of congestion is lacking; this represents, in our opinion, another promising research direction.

2.5 Conclusions

In this chapter we have provided an introduction to maritime logistics and to the main operations and processes that arise in container terminal management.

Relevant operations research literature has been reviewed and current research trends and challenges have been discussed.

We believe that future research directions should focus on bridging the gap between the different research challenges arising in container terminal management, with a focus on integrated solution approaches.

Chapter 3

The Tactical Berth Allocation Problem

In this chapter we propose a new model for the integrated planning of container terminal operations, called the Tactical Berth Allocation Problem, that combines berth allocation and quay crane assignment into a unique optimization model.

The studied problem contributes to bridge the gap between the different research challenges arising in container terminal management, with a specific focus on integrated solution approaches. In particular, we address tactical planning aspects and we take into account yard congestion issues related to the transshipment flow of containers.

3.1 Introduction

Among the several problems addressed in the literature, one of the most relevant is the well known Operational Berth Allocation Problem (OBAP), which consists of assigning and scheduling ships to berthing positions along the quay, with the aim of minimizing ships' turnaround time. The OBAP typically covers a planning horizon of at most one week, due to the uncertainties of maritime traveling times.

This chapter deals with a new model for the integration, at the tactical level, of the berth allocation problem with quadratic yard costs and the quay crane assignment problem. Our specific motivation in building a Tactical Berth Allocation Problem (TBAP), is not simply the obvious one of considering a longer planning horizon, but mainly that of supporting decisions made by terminal managers in the negotiation process with shipping lines. During this process, terminal managers need to evaluate the impact on the performance of the terminal of assigning certain operating resources, i.e., berths and quay cranes, to the shipping lines. Drawing inspiration from the actual negotiation process, the key feature of our model is the inclusion of a Quay Crane (QC) profile. This profile, which represents the number of quay cranes available to a berthed vessel at each time step, is explicitly modeled as a decision variable. While

this will be clarified later in the chapter, for the remainder of this section we highlight the main features of TBAP assuming the reader to be familiar with existing OBAP formulations.

Basically, both the tactical and the operational problems deal with assigning and scheduling ships to berthing positions, i.e. deciding where and when the ships should moor. Both the TBAP and the OBAP aim to balance terminal costs and service quality. However, as already noted, the different decision levels and time frames induce different problems. In the TBAP service quality depends upon the negotiation between the terminal and the shipping lines regarding the terminal resources. A higher service quality occurs when the terminal can accommodate shipping lines requests in terms of expected berthing times, assigned quay cranes, and expected handling times. In the TBAP, for a given amount of requested QC hours, it could be possible to create different QC profiles. For example, assume that we have a request for a vessel that requires six QCs work shifts, and the customer is potentially willing to accept either an intensive profile (for example three QCs on two work shifts) or a longer one (two QCs on three work shifts). The terminal managers want to know the trade-off between the two profiles. The faster one will be likely more satisfying for the customer because of the smaller handling time; while the slower one will put less pressure on the quay cranes availability, which could be a bottleneck in some periods. However, the problem is more complicated because if the QC availability is not a limiting factor, then a faster handling time is advantageous for the terminal, because it augments berth availability. This shows why the Quay Crane Assignment Problem (QCAP), i.e. deciding how many QCs to assign and for how long, has an impact on the berth allocation.

The TBAP, thanks to the longer planning horizon, can optimize the terminal's total costs in a more comprehensive way. In a transshipment terminal, containers arrive and depart on vessels while being temporarily stored in the yard. When unloading a vessel, the discharged containers must be allocated to yard positions close enough to the vessel berthing point in order to speed up the vessel handling. However, when the departure position of a container is far from its yard position, the container must be reallocated before the arrival of the outbound vessel. In the OBAP, since the planning horizon is shorter than the average container dwell time inside the yard, one can assume that the majority of the outbound containers are already in the yard, and disregard the effects of transshipment flows inside the yard. In the TBAP, the yard costs cannot be simplified by this assumption, and a quadratic term must be considered to account for the simultaneous assignment of vessels to berths.

The remainder of this chapter is organized as follows. Specific literature is reviewed in Section 3.2. The problem description as well as two formulations for the TBAP are presented in Section 3.3. A heuristic solution algorithm based on tabu search and mathematical programming is proposed in Section 3.4. While the mathematical model is not addressable by a state-of-the-art solver, the proposed heuristic proves its efficacy as documented by the computational experiments reported in Section 3.5.

3.2 Literature review

The operational berth allocation problem has received so far a larger attention than the tactical one in the scientific literature. In this section we discuss in more detail only the articles relevant to the TBAP.

Cordeau et al. (2007) provides an initial introduction for the basis of the TBAP. The paper deals with the Service Allocation Problem (SAP), a tactical problem arising in the yard management of a container transhipment terminal. A *service,* also called *port route*, is the sequence of ports visited by a vessel. Shipping companies plan their port routes in order to match the demand for freight transportation. A shipping company usually asks the terminal management to dedicate specific areas of the yard and the quay (home berths) to their services. The SAP objective is the minimization of container rehandling operations inside the yard through choosing the home berth for each service. The SAP is formulated as a Generalized Quadratic Assignment Problem (GQAP, see e.g. Cordeau et al. (2006), and Hahn et al. (2008)) with side constraints, and solved by an evolutionary heuristic. The SAP can be seen as a relaxed TBAP when collapsing the temporal dimension, and disregarding the choice of QC profiles. The SAP output consists of reference home berths that planners consider when drawing the berth template.

Park and Kim (2003) are the first to integrate the BAP in the continuous case with the QCAP, also considering the scheduling of quay cranes. The integrated problem is formulated as an integer program and a two-phase solution procedure is presented to solve the model. In the first phase, the berthing time and position of vessels and the number of quay-cranes assigned to each vessel at each time step are determined using Lagrangian relaxation and a subgradient optimization technique; the objective is to minimize the sum of penalty costs over all ships. In the second phase, cranes are scheduled along the quay via dynamic programming, with the objective of minimizing the number of setups. With respect to the problem formulation, the authors take into account some practical aspects such as favourite berthing positions of vessels, maximum and minimum number of cranes to be assigned to each vessel, penalty costs due to earlier or later berthing time, and later departure time (with respect to previously committed time).

Meisel and Bierwirth (2006) investigate the simultaneous allocation of berths and quay cranes, focusing on the reduction of QCs idle times, which significantly impact on terminal's labor costs. A heuristic scheduling algorithm based on priority-rules methods for resource-constrained project scheduling is proposed and tested on instances based on real data. Preliminary results, compared to the manually generated schedules which have been used in practice, are encouraging. In this approach, each vessel represents an activity which can be performed in 8 different modes, each mode representing a given QC-to-Vessel assignment over time. The concept of "mode" seems analogous to the concept of profile we have introduced so far; however, no detailed description of these modes is available in the paper.

Imai et al. (2008) address the simultaneous berth-crane allocation and scheduling problem, taking into account physical constraints of quay cranes, which cannot move freely among berths as they are all mounted on the same track and cannot bypass each other. A MIP formulation which minimizes the total service time is proposed and a genetic algorithm-based heuristic is developed to find an approximate solution. As authors recognize, the relationship between the number of cranes and the handling time is not investigated in the paper; indeed, a reference number of cranes needed by each ship is assumed to be given as input of the problem.

Meisel and Bierwirth (2009) study the integration of BAP and QCAP with a focus on quay crane productivity. An integer linear model is presented and a construction heuristic, local refinement procedures and two meta-heuristics are developed to solve the problem. Authors compare their approach to the one proposed by Park and Kim (2003) over the same set of instances and they always provide better solutions. An analysis of quay crane's productivity losses, mainly due to interference among QCs and to the distance of the vessel berthing position from the yard areas assigned to this vessel, is also presented and their impact on the terminal's service cost is evaluated.

3.3 Mathematical Models

In this section we provide a compact description of the problem and motivate our modeling choices. In particular, we illustrate the concept of QC assignment profiles in Section 3.3.1 and we provide additional details regarding yard costs related to transshipment flows among ships in Section 3.3.2. The described cost figures and operational parameters are provided by the Medcenter Container Terminal (MCT), port of Gioia Tauro, Italy. We present a mixed integer quadratic programming formulation (MIQP) for the TBAP with integrated QCs assignment in Section 3.3.3, as well as a linearization of the MIQP model which reduces to a mixed integer linear program (MILP) in Section 3.3.4.

With respect to the BAP, we consider the discrete case. As described in Section 3.1, the fundamental modeling tool of our formulation is the quay crane profile, representing the number of quay cranes assigned to the ship at each time step. Given $\mathfrak{n}$ ships, $\mathfrak{m}$ berths, and a particular time horizon, we aim to assign a home berth and a QC profile to each ship, as well as schedule incoming ships according to time windows on their arrival time and on berths' availabilities. These decisions are made to maximize the total value of chosen QC assignment profiles and minimize the housekeeping costs generated by transshipment flows between ships.

The integrated problem presents increased complexity because the ship handling time is not constant but depends on the number of quay cranes assigned to the ship. With respect to the classical OBAP, this implies additional decision variables and constraints.

3.3.1 QC assignment profiles

The use of QC profiles to handle the assignment of quay cranes to ships is firstly motivated by the needs of terminal managers during negotiations with shipping companies. In particular, managers need to be aware of the trade-off among the different QC profiles they may propose to the shippers.

Concerning the mathematical model, the concept of QC profiles can capture real-world issues, and works well to represent the control that the terminal has on several aspects of QC assignment during the optimization process. These are the main reasons why we have explicitly introduced this feature in the formulation.

We assume to have a set of feasible QC profiles P_{i} for every ship $\mathsf{i} \in \mathsf{N}$, which are defined by the terminal according to the specific amount of QC hours requested by the ship, internal rules and good practices related to the efficiency of terminal operations, and legal contracts.

Our approach differs from the traditional modeling choice present in the literature, e.g. Park and Kim (2003), Imai et al. (2008), Meisel and Bierwirth (2009), which usually assigns quay cranes hour by hour, without any control on the final outcome in terms of QC profiles. As mentioned, the concept of "mode" in Meisel and Bierwirth (2006) is somehow similar to our concept of QC profile, but the authors do not provide enough details to allow comparisons.

For a given vessel, feasible QC profiles usually vary in length (number of shifts) as well as in the distribution of QC cranes over the active shifts, in order to ensure the requested amount of QC hours.

Some operational constraints, which are usually not taken into account by other models, can be directly integrated in the definition of the set of feasible profiles. A common rule, for instance, is that quay cranes are assigned to vessels and placed on the corresponding quay segment shift by shift. This means that a quay crane cannot be moved from one vessel to another at any arbitrary moment, but only between two shifts. This constraint can be easily handled by forcing profiles to maintain a constant number of quay cranes during a shift. Another good practice is to keep the distribution of quay cranes as regular as possible among active shifts; a variance of one or at most two QCs can be considered acceptable, although high variability should be avoided as much as possible. Also this feature can be included in our profile set definition easily.

In addition to these general rules, the terminal can manage more directly some priority-related issues. Since the set of feasible QC profiles is defined for every ship, managers can assign different minimum and maximum handling times not only depending on the ship's size and the traffic volume but also depending on the ship's relative importance for the terminal. This also applies for the minimum and maximum number of quay cranes allowed to be assigned to a given ship. We would like to remark that this is an important advantage provided by our approach, compared to other models in the literature where handling time is either considered an input of the problem or barely controlled by time windows on the vessel's arrival and departure,

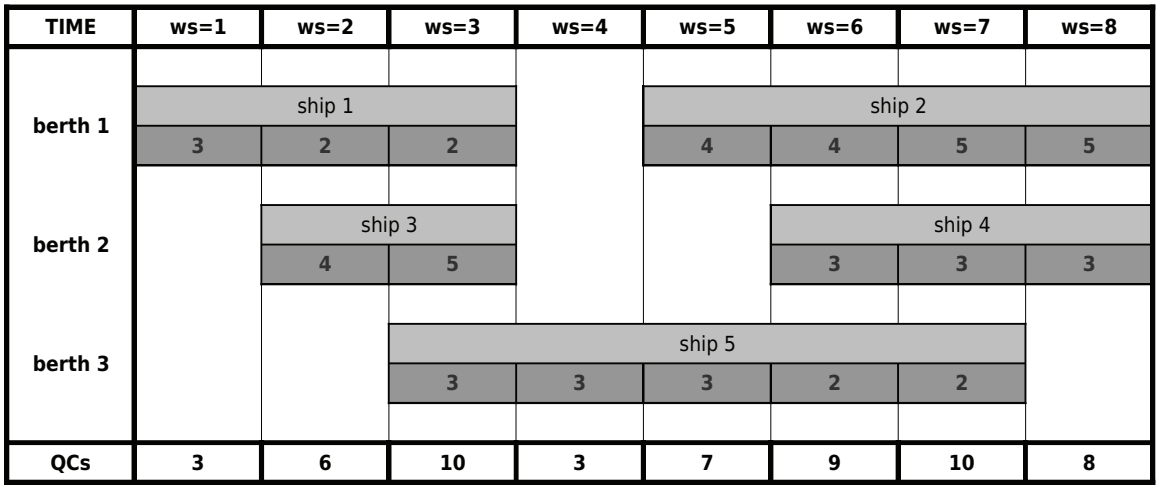

Figure 3.1: *Example of a Berth & Quay Cranes Allocation Plan.*

in addition to some priority-related weights in the objective function, which usually aim to serve faster vessels with high priority. Furthermore, each QC profile has an associated "value" which reflects technical aspects (such as the resources utilized) but which is also computed by taking into account the specific vessel which will use the profile. In other words, the same QC profile can have different values when applied to different ships, according to their priority or importance.

We can also include productivity losses due to quay crane interference, recently studied by Meisel and Bierwirth (2009), in the definition of the feasible set of profiles. Indeed, we can use the approach suggested by the authors to compute, for each profile, the actual quay crane productivity instead of the theoretical one.

In order to improve understanding of the QC profile concept, and its relation with the integration between Berth and QC Allocation planning, we provide in Figure 3.1 an example of such a plan. The example is for the scheduling and assignment of 5 vessels to 3 berths over a time horizon of 8 working shifts. Consider, for instance, the Ship 1: it berths at shift 1, and three QCs are allocated to it for carrying out operations during the same shift; next, at working shifts 2 and 3, Ship 1 remains berthed, but one QC is de-allocated, with only two QCs remaining allocated to the ship. At the end of shift 3 operations terminate and the ship is released.

3.3.2 Transshipment-related yard costs

When loading (or unloading) a vessel, the containers must be at (or allocated to) yard positions close enough to the vessel berthing point to maintain the required speed of quay crane operation. Usually, at the Medcenter Container Terminal (MCT) of the port of Gioia Tauro, a yard position is evaluated as satisfyingly close to a berth

if the distance along the quay axis is less than 600 meters. This maximal close distance value can be lowered for higher priority workloads. Furthermore, when we estimate yard-related transshipment costs induced by berth allocation, we do not consider the real yard position of the loading and unloading containers. In fact, we assume that the expected travelled distance along the quay axis is given by the distance between the incoming and outgoing berths. If this distance is lower than the threshold value of 600 meters, then a container will likely move from the quay to its assigned yard position when unloading and from this yard position to the quay when loading. However, in a large transshipment terminal, such as the one at the Gioia Tauro port, the distance between the unloading berth and the loading one is often larger than 600 meters. Therefore, containers are moved before the arrival of the outgoing vessel from their current yard positions to new ones closer to the outgoing berth. This process is called *housekeeping* and requires a dedicated management in order to accommodate operational constraints like the capacity of the yard positions, the maximum container handling workload for a given work shift, etc. In synthesis, the yard management deals with a dynamic allocation of containers through their duration-of-stay inside the terminal, see Moccia et al. (2009). A rule motivated by cost minimization enforces that whenever the distance along the quay axis is larger than 1100 meters, the yard-to-yard transfer is operated by deploying multi trailer vehicles instead of straddle carriers. Therefore we have a yard cost function that depends upon the distance between the incoming and outgoing berths according to three transport modalities:

- the distance is below 600 meters: no housekeeping is performed, the unitary transport cost, euro/(meter x container), depends upon straddle carriers cost figures only;
- the distance is between 600 and 1100 meters: a housekeeping process is activated by deploying straddle carriers only, however we face a transport cost larger than in the previous distance range;
- the distance is larger than 1100 meters: the housekeeping is performed by using the less expensive multi trailer vehicles (higher capacity than the straddle carriers).

The qualitative pattern of this piecewise linear cost function is given in Figure 3.2, where we indicate by SC the direct transfer with straddle carriers, by HK SC the housekeeping with straddle carriers, and by HK MT the housekeeping with multi trailer vehicles.

3.3.3 MIQP Formulation

In this section we present a mixed integer quadratic programming formulation for the TBAP with QCs assignment. Input data for this problem are:

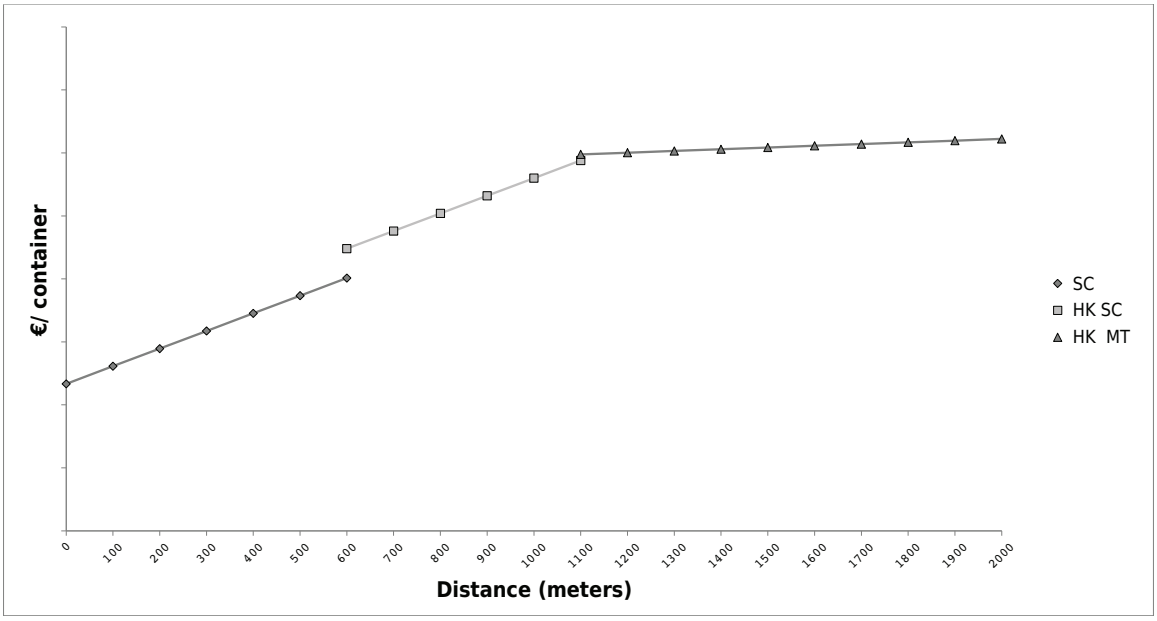

Figure 3.2: *Yard costs according to the distance between the incoming and outgoing berths.*

N	set of vessels, with $\|N\| = n$;
M	set of berths, with $\|M\| = m$;
H	set of time steps (each time step $h \in H$ is submultiple of the work shift length);
S	set of the time step indexes $\{1, ..., \bar{s}\}$ relative to a work shift; $\bar{s}$ represents the number of time steps in a work shift;
H^s	subset of H which contains all the time steps corresponding to the same time step $s \in S$ within a work shift;
P_i^s	set of feasible quay crane assignment profiles for the vessel $i \in N$ when vessel arrives at a time step with index $s \in S$ within a work shift;
P_i	set of quay crane assignment profiles for the vessel $i \in N$, where $P_i = \cup_{s \in S} P_i^s$;
t_i^p	handling time of ship $i \in N$ under the QC profile $p \in P_i$ expressed as multiple of the time step length;
v_i^p	the value of serving the ship $i \in N$ by the quay crane profile $p \in P_i$;
q_i^{pu}	number of quay cranes assigned to the vessel $i \in N$ under the profile $p \in P_i$ at the time step $u \in (1, ..., t_i^p)$, where $u = 1$ corresponds to the ship arrival time;
Q^h	maximum number of quay cranes available at the time step $h \in H$;
f_{ij}	number of containers exchanged between vessels $i, j \in N$;
d_{kw}	unit housekeeping cost between yard slots corresponding to berths $k, w \in M$;
$[a_i, b_i]$	[earliest, latest] feasible arrival time of ship $i \in N$;
$[a^k, b^k]$	[start, end] of availability time of berth $k \in M$;
$[a^h, b^h]$	[start, end] of the time step $h \in H$.

We define a graph $G^k = (V^k, A^k)\ \forall k \in M$, where $V^k = N \cup \{o(k), d(k)\}$, with $o(k)$ and $d(k)$ additional vertices representing berth k, and $A^k \subseteq V^k \times V^k$. The following decision variables are defined:

- $x_{ij}^k \in \{0,1\}\ \forall k \in M, \forall (i,j) \in A^k$, set to 1 if ship j is scheduled after ship i at berth k, and 0 otherwise;

- $y_i^k \in \{0,1\}\ \forall k \in M, \forall i \in N$, set to 1 if ship i is assigned to berth k, and 0 otherwise;

- $\gamma_i^h \in \{0,1\}\ \forall h \in H, \forall i \in N$, set to 1 if ship i arrives at time step h, and 0 otherwise;

- $\lambda_i^p \in \{0,1\}\ \forall p \in P_i, \forall i \in N$, set to 1 if ship i is served by the profile p, and 0 otherwise;

- $\rho_i^{ph} \in \{0,1\}\ \forall p \in P_i, \forall h \in H, \forall i \in N$, set to 1 if ship i is served by profile p and arrives at time step h, and 0 otherwise;

- $T_i^k \geq 0\ \forall k \in M, \forall i \in N$, representing the berthing time of ship i at the berth k, i.e. the time when the ship moors;

- $T_{o(k)}^k \geq 0\ \forall k \in M$, representing the starting operation time of berth k, i.e. the time when the first ship moors at the berth;

- $T_{d(k)}^k \geq 0\ \forall k \in M$, representing the ending operation time of berth k, i.e. the time when the last ship departs from the berth.

The TBAP with QC assignment can therefore be formulated as follows:

$$\max \sum_{i\in N}\sum_{p\in P_i} v_i^p \lambda_i^p - \frac{1}{2}\sum_{i\in N}\sum_{k\in M} y_i^k \sum_{j\in N}\sum_{w\in M} f_{ij} d_{kw} y_j^w \tag{3.1}$$

$$\text{s.t.} \sum_{k\in M} y_i^k = 1 \qquad \forall i \in N, \tag{3.2}$$

$$\sum_{j\in N\cup\{d(k)\}} x_{o(k),j}^k = 1 \qquad \forall k \in M, \tag{3.3}$$

$$\sum_{i\in N\cup\{o(k)\}} x_{i,d(k)}^k = 1 \qquad \forall k \in M, \tag{3.4}$$

$$\sum_{j\in N\cup\{d(k)\}} x_{ij}^k - \sum_{j\in N\cup\{o(k)\}} x_{ji}^k = 0 \qquad \forall k \in M, \forall i \in N, \tag{3.5}$$

$$\sum_{j\in N\cup\{d(k)\}} x_{ij}^k = y_i^k \qquad \forall k \in M, \forall i \in N, \tag{3.6}$$

$$T_i^k + \sum_{p\in P_i} t_i^p \lambda_i^p - T_j^k \le (1 - x_{ij}^k) M1 \qquad \forall k \in M, \forall i \in N, \forall j \in N \cup \{d(k)\}, \tag{3.7}$$

$$T_{o(k)}^k - T_j^k \le (1 - x_{o(k),j}^k) M2 \qquad \forall k \in M, \forall j \in N, \tag{3.8}$$

$$a_i y_i^k \le T_i^k \qquad \forall k \in M, \forall i \in N, \tag{3.9}$$

$$T_i^k \le b_i y_i^k \qquad \forall k \in M, \forall i \in N, \tag{3.10}$$

$$a^k \le T_{o(k)}^k \qquad \forall k \in M, \tag{3.11}$$

$$T_{d(k)}^k \le b^k \qquad \forall k \in M, \tag{3.12}$$

$$\sum_{p\in P_i} \lambda_i^p = 1 \qquad \forall i \in N, \tag{3.13}$$

$$\sum_{h\in H^s} \gamma_i^h = \sum_{p\in P_i^s} \lambda_i^p \qquad \forall i \in N, \forall s \in S, \tag{3.14}$$

$$\sum_{k\in M} T_i^k - b^h \le (1 - \gamma_i^h) M3 \qquad \forall h \in H, \forall i \in N, \tag{3.15}$$

$$a^h - \sum_{k\in M} T_i^k \le (1 - \gamma_i^h) M4 \qquad \forall h \in H, \forall i \in N, \tag{3.16}$$

$$\rho_i^{ph} \ge \lambda_i^p + \gamma_i^h - 1 \qquad \forall h \in H, \forall i \in N, \forall p \in P_i, \tag{3.17}$$

$$\sum_{i\in N}\sum_{p\in P_i}\sum_{u=\max\{h-t_i^p+1;1\}}^{h} q_i^{p(h-u+1)} \rho_i^{pu} \le Q^h \qquad \forall h \in H^{\bar{s}}, \tag{3.18}$$

$$x_{ij}^k \in \{0,1\} \qquad \forall k \in M, \forall (i,j) \in A^k, \tag{3.19}$$

$$y_i^k \in \{0,1\} \qquad \forall k \in M, \forall i \in N, \tag{3.20}$$

$$\gamma_i^h \in \{0,1\} \qquad \forall h \in H, \forall i \in N, \tag{3.21}$$

$$\lambda_i^p \in \{0,1\} \qquad \forall p \in P_i, \forall i \in N, \tag{3.22}$$

$$\rho_i^{ph} \in \{0,1\} \qquad \forall p \in P_i, \forall h \in H, \forall i \in N, \tag{3.23}$$

$$T_i^k \ge 0 \qquad \forall k \in M, \forall i \in N \cup \{o(k), d(k)\}, \tag{3.24}$$

where M1, M2, M3 and M4 represent large positive constants.

The objective function (3.1) maximizes the difference between the sum of the values of the chosen quay crane assignment profiles over all the vessels and the yard-related housekeeping costs generated by the flows of containers exchanged between vessels. Constraints (3.2) state that every ship i must be assigned to one and only one berth k. Constraints (3.3) and (3.4) define the outgoing and incoming flows to the berths, while flow conservation for the remaining vertices is ensured by constraints (3.5). Constraints (3.6) state the link between variables x_{ij}^k and y_i^k, while precedences in every sequence are ensured by constraints (3.7) and (3.8), which coherently set time variables T_i^k. Time windows on the arrival time are stated for every ship by constraints (3.9) and (3.10), while time windows on berths' availabilities are stated by constraints (3.11) and (3.12). Constraints (3.13) ensure that one and only one QC profile is assigned to every ship. Constraints (4.41) define the link between variables γ_i^h and λ_i^p while constraints (3.15) and (3.16) link binary variables γ_i^h to the arrival time T_i^k. Observe that constraints (3.10) imply $T_i^k = 0$ when ship $i \in N$ does not moor at berth $k \in K$. Variables ρ_i^{ph} are linked to variables λ_i^p and γ_i^h by constraints (3.17): in particular, ρ_i^{ph} is equal to 1 if and only if $\lambda_i^p = \gamma_i^h = 1$. Finally, constraints (3.18) ensure that, at every time step, the total number of assigned quay cranes does not exceed the number of quay cranes which are available in the terminal.

To better illustrate capacity constraints (3.18), we come back to the example shown in Figure 3.1, which refers to the scheduling and assignment of $|N| = 5$ vessels to $|M| = 3$ berths over a time horizon of $|H| = 8$ time steps. Here we assume that a time step corresponds to one working shift. From the plan we can infer the following non-zero data:

$i = 1 \quad \rho_1^{p1} = 1 \quad t_1^p = 3 \quad q_1^{p1} = 3,\ q_1^{p2} = 2,\ q_1^{p3} = 2$
$i = 2 \quad \rho_2^{p5} = 1 \quad t_2^p = 4 \quad q_2^{p1} = 4,\ q_2^{p2} = 4,\ q_2^{p3} = 5,\ q_2^{p4} = 5$
$i = 3 \quad \rho_3^{p2} = 1 \quad t_3^p = 2 \quad q_3^{p1} = 4,\ q_3^{p2} = 5$
$i = 4 \quad \rho_4^{p6} = 1 \quad t_4^p = 3 \quad q_4^{p1} = 3,\ q_4^{p2} = 3,\ q_4^{p3} = 3$
$i = 5 \quad \rho_5^{p3} = 1 \quad t_5^p = 5 \quad q_5^{p1} = 3,\ q_5^{p2} = 3,\ q_5^{p3} = 3,\ q_5^{p4} = 2,\ q_5^{p5} = 2$

For each time step $h = 1, \ldots 8$, the corresponding constraint in (3.18) counts the number of active quay cranes. Let us consider the case $h = 3$: the index u changes its range for each vessel, because, starting from $h = 3$, it goes backwards until the beginning of the profile. Therefore we have:

$i = 1 \quad u = 1, 2, 3$
$i = 2 \quad u = 1, 2, 3$
$i = 3 \quad u = 2, 3$
$i = 4 \quad u = 1, 2, 3$
$i = 5 \quad u = 1, 2, 3$

We remark that vessels $i = 2, 4$ do not contribute to the sum, since $\rho_2^{pu} = \rho_4^{pu} = 0\ \forall u = 1, 2, 3$ and this is coherent with the plan. For the remaining vessels, ρ_i^{pu} is not zero only for one value u^*:

$$\begin{aligned} i = 1 \quad u^* = 1 \quad &\Longrightarrow q_1^{p(3-1+1)} = q_1^{p3} = 2 \\ i = 3 \quad u^* = 2 \quad &\Longrightarrow q_3^{p(3-2+1)} = q_3^{p2} = 5 \\ i = 5 \quad u^* = 3 \quad &\Longrightarrow q_5^{p(3-3+1)} = q_5^{p1} = 3 \end{aligned}$$

Therefore the sum in (3.18) reduces to:

$$q_1^{p3} + q_3^{p2} + q_5^{p1} = 2 + 5 + 3 = 10$$

which is indeed the total number of quay cranes that are active at time step $h = 3$.

Finally, we observe that the TBAP formulation (3.1)–(3.24) can be interpreted as a Multi-Depot Vehicle Routing Problem with Time Windows (MDVRPTW), see e.g. Cordeau et al. (2005), with an additional quadratic component in the objective function and side constraints.

3.3.4 MILP Formulation

The quadratic objective function (3.1) can be linearized by defining an additional decision variable $z_{ij}^{kw} \in \{0, 1\}\ \forall i, j \in N,\ \forall k, w \in M$, which is equal to 1 if $y_i^k = y_j^w = 1$ and 0 otherwise. Variables z_{ij}^{kw} are linked to variables y_i^k by the following additional constraints:

$$\sum_{k \in K} \sum_{w \in K} z_{ij}^{kw} = g_{ij} \qquad \forall i, j \in N, \tag{3.25}$$

$$z_{ij}^{kw} \leq y_i^k \qquad \forall i, j \in N, \forall k, w \in M \tag{3.26}$$

$$z_{ij}^{kw} \leq y_j^w \qquad \forall i, j \in N, \forall k, w \in M \tag{3.27}$$

where g_{ij} is a constant which is equal to 1 if $f_{ij} > 0$ and 0 otherwise.

TBAP can therefore be formulated as a mixed integer linear program as follows:

$$\max \quad \sum_{i \in N} \sum_{p \in P_i} v_i^p \lambda_i^p - \frac{1}{2} \sum_{i \in N} \sum_{j \in N} \sum_{k \in M} \sum_{w \in M} f_{ij} d_{kw} z_{ij}^{kw} \tag{3.28}$$

$$\text{s.t.} \qquad (3.2) - (3.24)\,, (3.25) - (3.27).$$

3.4 A two-level heuristic for TBAP

Solving the TBAP model with a general-purpose solver is difficult, using either the MIQP or MILP formulation, as shown by computational results in Section 3.5. A specialized heuristic is therefore needed. We propose a two-level heuristic algorithm for solving the TBAP, which is illustrated in this section.

Our heuristic is organized in two stages: first, we identify a QC profiles' assignment for the ships; second, we solve the resulting berth allocation problem for the given QC assignment. This procedure is repeated for several QC profiles, which are chosen,

Algorithm 1: TBAP Bi-level Heuristics

Initialization : Assign a QC profile to each ship.

repeat

1. solve BAP via tabu search;
2. update the QC profiles' assignment by using reduced costs.

until *stop criterion ;*

Figure 3.3: *Scheme of the heuristic algorithm for TBAP.*

iteration by iteration, using the traditional reduced costs arguments of mathematical programming. A scheme of the heuristic algorithm for TBAP is outlined in Figure 3.3.

The initialization consists of assigning a QC profile to each ship. The maximum value profile is chosen for each ship (ties are broken arbitrarily). This is equivalent to assign binary values to variables λ such that equations (3.13) are satisfied. Once the first QC profiles' assignment has been done, the two-level procedure starts.

Given a QC assignment, the TBAP reduces to the berth allocation problem, with additional constraints due to the QC total capacity. We develop a tabu search algorithm that solves the BAP (step 1 in Figure 3.3), aiming to minimize the yard-related transshipment housekeeping costs:

$$\frac{1}{2}\sum_{i\in N}\sum_{k\in M} y_i^k \sum_{j\in N}\sum_{w\in M} f_{ij} d_{kw} y_j^w \tag{3.29}$$

We remark that we take into account only the quadratic term of the TBAP objective function in (3.1) since, for a given QC profiles' assignment, the total value of profiles is constant. The tabu search algorithm for the BAP is illustrated in Section 3.4.1.

In step 2, the QC profiles' assignment vector is updated. The new set of profiles is determined using the reduced costs of variables λ, whose estimation is illustrated in Section 3.4.2.

3.4.1 Tabu search for the berth allocation

Our tabu search heuristic is an adaptation of the one of Cordeau et al. (2005) for the OBAP. However, while in Cordeau et al. (2005) the function to be minimized is the weighted sum for every ship of the service time in the port, our heuristic minimizes the yard-related housekeeping costs generated by the flows of containers exchanged between vessels. Another difference is the handling of the side constraints concerning

the QC availability–for a given assignment of QC profiles (vector λ), our tabu search must take into account the QC capacity constraints (3.18).

Denote by S the set of solutions that satisfy constraints (3.2) – (3.9) and (3.11). The heuristic explores the solution space by moving at each iteration from the current solution s to the best solution in its neighborhood $N(s)$. Each solution $s \in S$ is represented by a set of m berth sequences such that every ship belongs to exactly one sequence. This solution may, however, violate the time window constraints associated with the ships and the berths, and the QC availability. The time window constraint on ship i on a berth k is violated if the arrival time T_i^k of the ship is larger than the time window's upper bound b_i. Berthing before a_i is not allowed; in other words, $T_i^k \geq a_i$. Similarly, the time window of berth k is violated when the completion time of a ship i assigned to berth k is larger than the berth time window's upper bound b^k.

Let $c(s)$ denote the cost of solution defined in (3.29), and let $w_1(s)$ denote the total violation of ships' time window constraints, equal to the sum of the violations on the n ships. We indicate as $w_2(s)$ the total violation of berths' time window constraints, equal to the sum of the violations on the m berths. Finally, let $w_3(s)$ be the total violation of QC availability for each time step of the planning horizon. Solutions are then evaluated by means of a penalized cost function $f(s) = c(s)+\alpha_1 w_1(s)+\alpha_2 w_2(s)+\alpha_3 w_3(s)$, where the α values are positive parameters. By dynamically adjusting the value of these parameters, the relaxation mechanism facilitates the exploration of the search space and is particularly useful for tightly constrained instances.

The tabu search method is based on the definition of attributes used to characterize the solutions of S. They are also used to control tabu tenures and to implement a diversification strategy. An attribute set $B(s) = \{(i,k)$: ship i is assigned to berth $k\}$ is associated with each solution $s \in S$. The neighborhood $N(s)$ of a solution s is defined by applying a simple operator that removes an attribute (i,k) from $B(s)$ and replaces it with another attribute (i,k'), where $k \neq k'$. When ship i is removed from berth k, the sequence is simply reconnected by linking the predecessor and successor of the ship. Insertion in sequence k' is then performed between two consecutive ships so as to minimize the value of $f(s)$. When a ship i is removed from berth k, its reinsertion in that berth is forbidden for the next θ iterations by assigning a tabu status to the attribute (i,k).

An aspiration criterion allows the revocation of the tabu status of an attribute if that would allow the search process to reach a solution of smaller cost than that of the best solution identified having that attribute. To diversify the search, any solution $\overline{s} \in N(s)$ such that $f(\overline{s}) \geq f(s)$ is penalized by a factor proportional to the addition frequency of its attributes, and by a scaling factor. More precisely, let ξ_{ik} be the number of times attribute (i,k) has been added to the solution during the process and let ζ be the number of the current iteration. A penalty $p(\overline{s}) = \beta c(\overline{s})\xi_{ik}/\zeta$ is added to $f(\overline{s})$. The scaling factor $c(\overline{s})$ introduces a correction to adjust the penalties with respect to the total solution cost. Finally, the parameter β is used to control the intensity of the diversification. These penalties have the effect of driving the search

process toward less explored regions of the search space. For notational convenience, assume that $p(\overline{s}) = 0$ if $f(\overline{s}) < f(s)$.

In order to generate a starting solution, the algorithm assigns the ships to the berths at random. This initial solution is constructed by relaxing the time window and QC availability constraints, and therefore it is usually infeasible. However, this is not an issue for the tabu search heuristic.

The search starts from this initial solution and selects, at each iteration, the best non-tabu solution $\overline{s} \in N(s)$. After each iteration, the value of parameters α_1, α_2, and α_3 are modified by a factor $1+\delta$, where $\delta > 0$. For example, if the current solution is feasible with respect to ships' time window constraints, the value of α_1 is divided by $1+\delta$; otherwise, it is multiplied by $1+\delta$. Analogously for the berths' time window and QC availability constraints, i.e. parameters α_2 and α_3, respectively. This process is repeated for η iterations and the best feasible solution s^* is updated throughout the search.

3.4.2 Profile update via mathematical programming

The profiles' updating procedure represents step 2 in the algorithm's scheme illustrated in Figure 3.3. It relies on the MILP formulation for TBAP illustrated in Section 3.3.4. The basic idea of this step is to use the information of reduced costs in order to be able to update vector λ of QC profiles' assignment in a smart way.

Let $s^* = [\bar{x}, \bar{y}, \bar{T}]$ be the BAP solution provided by tabu search for a given QC profile assignment $\bar{\lambda}$. In particular, we are interested in reduced costs of variables λ, which we denote $\tilde{c}(\lambda)$. We remark that a BAP solution plus a QC assignment represent a feasible solution for TBAP. At each iteration, we solve the linear relaxation of the MILP formulation, with the additional constraints:

$$\bar{x} - \epsilon \leq x \leq \bar{x} + \epsilon \tag{3.30}$$

$$\bar{y} - \epsilon \leq y \leq \bar{y} + \epsilon \tag{3.31}$$

$$\bar{T} - \epsilon \leq T \leq \bar{T} + \epsilon \tag{3.32}$$

$$\bar{\lambda} - \epsilon \leq \lambda \leq \bar{\lambda} + \epsilon \tag{3.33}$$

As remarked, e.g., by Desrosiers and Lübbecke (2005), the shadow prices of constraints (3.30)–(3.33) are the reduced costs of original variables x, y, T and λ. At each iteration, we identify the $\lambda^{p^*}_{i^*}$ variable with the maximum reduced cost:

$$(i^*, p^*) = \arg\max_{i \in N, p \in P_i} \{\tilde{c}(\lambda^p_i)\} \tag{3.34}$$

If $\tilde{c}(\lambda^{p^*}_{i^*}) > 0$, profile p^* is assigned to vessel i^*, i.e. $\lambda^{p^*}_{i^*} = 1$ and $\lambda^p_{i^*} = 0 \ \forall p \neq p^*$. We remark that this update, concerning a single vessel, results in a new vector λ of QC profiles' assignment, which differs from the previous one only for two components. Step 2 of the algorithm is now completed. The updated QC profiles' assignment

vector is passed back to the BAP tabu search (step 1) and a new BAP solution is computed.

The whole procedure terminates when all reduced costs are non-positive, or other additional stopping criteria are reached, such as the maximum number of iterations or the time limit.

In order to prevent cycles, a tabu mechanism has been implemented, keeping track of the last ψ updates in the form (i, p). The tabu list (TL) is updated at each iteration and its length has been fixed to $\psi = 0.5n \times \bar{p}$, where $\bar{p} = \max_{i \in N} |P_i|$. According to this mechanism, the pair (i^*, p^*) is therefore chosen as $(i^*, p^*) = \underset{i \in N, p \in P_i : (i,p) \notin TL}{\arg\max} \{\tilde{c}(\lambda_i^p)\}$.

It may happen that the tabu search returns a BAP solution which is infeasible for TBAP with respect to time windows and/or the QC availability. In this case the profiles' update via mathematical programming cannot be performed. We therefore update the set of profiles by randomly assigning a new QC profile to each ship.

3.5 Computational results

In this section we first illustrate how realistic test instances have been generated, then we present results obtained through a general-purpose solver and we compare them with our heuristic algorithm.

3.5.1 Generation of test instances

Our tests are based on real data provided by MCT. We have access to historical berth allocation plans and quay cranes assignment plans concerning about 60 vessels per week over a time horizon of one month; specific information on vessels such as the arrival time and the total number of containers to be handled is also provided. Furthermore, data referring to the flows of containers exchanged between ships as well as a study on the yard-related transshipment costs is available.

Instances generated to validate our models rely on these real data. The quay, which is 3395 m long, is partitioned in 13 berthing points, which are equipped with 25 quay cranes (22 gantry cranes and 3 mobile cranes). The matrix of distances $[d_{kw}]$ is a 13x13 matrix that takes into account the costs estimated by the terminal to move containers between two berthing positions. Several matrices of flows $[f_{ij}]$ are generated according to the distributions of containers reported in the historical data. As usual, we distinguish between feeders and mother vessels: the traffic volume is mostly influenced by the proportion between these two classes, since mother vessels present a number of loading/unloading containers on average higher than feeders. Time windows for the ships' arrival are generated according to the historical data. Berths are assumed to be available for the whole time horizon, which we set to one week. A working day is divided in 4 shifts of 6 hours each, for a total of 56 time steps of 3 hours.

The sets of feasible profiles are synthetically generated in accordance with operational rules and good practices in use at the MCT terminal. As illustrated in Table 3.1, we fix a set of parameters for each ship class to which a profile must comply with in order to be feasible: namely, the minimum and the maximum number of QCs to be assigned to each vessel per shift as well as the minimum and the maximum handling time (HT) allowed for each class. We use a crane productivity of 24 containers per hour and we therefore obtain, per each class, a minimum and a maximum number of containers (column "volume" in the table): vessels' traffic volumes must comply with these ranges, according to the class they belong to. Furthermore, for all classes, a variation of at most 1 QC is allowed between a shift and the subsequent; profiles can start either at the beginning of the shift or in the middle of the shift.

Once the whole feasible set is generated for each class, profiles are assigned to vessels according to the QC hours they need to be operated. At this point, a monetary value is associated with the couple (vessel,profile) with respect to the number of containers to be handled. This value is then adjusted by taking into account the profile's length and the utilized resources with respect to the average case.

To validate our model, we consider 6 classes of instances:

- 10 ships and 3 berths, 1 week, 8 quay cranes;
- 20 ships and 5 berths, 1 week, 13 quay cranes;
- 30 ships and 5 berths, 1 week, 13 quay cranes;
- 40 ships and 5 berths, 2 weeks, 13 quay cranes;
- 50 ships and 8 berths, 2 weeks, 13 quay cranes;
- 60 ships and 13 berths, 2 weeks, 13 quay cranes.

For each class, we generate 12 instances, with high (H) and low (L) traffic volumes. Each scenario is tested with a set of $\bar{p} = 10, 20, 30$ feasible profiles for each ship. We remark that, by construction, instances of size $\bar{p} = 10$ are included in instances of size $\bar{p} = 20$, which are included in instances of size $\bar{p} = 30$. Thus, any feasible solution for $\bar{p} = 10$ is also feasible for $\bar{p} = 20, 30$ and so on.

3.5.2 CPLEX computational results

The MIQP and MILP formulations are tested with CPLEX 10.2, with emphasis on the feasibility of the solution.

Class	min QC	max QC	min HT	max HT	volume (min,max)
Mother	3	5	3	6	(1296, 4320)
Feeder	1	3	2	4	(288, 1728)

Table 3.1: *Parameters for the profile set's generation.*

Time limit for instances 10x3 is 1 hour; instances 20x5 and 30x5 have a time limit of 2 hours; instances 40x5, 50x8, 60x13 have a time limit of 3 hours.

Results are illustrated in Table 3.2. We report only instances for which CPLEX has found a feasible solution, at least. Surprisingly, no feasible solution is found for classes 30x5, 50x8 and 60x13; however, an upper bound is always provided.

10x3			**10x3**		
Instance	MILP	MIQP	Instance	MILP	MIQP
H1_10	99.17	98.90	L1_10	97.68	100.00
H1_20	97.91	97.96	L1_20	100.00	99.76
H1_30	97.98	98.76	L1_30	98.64	99.99
H2_10	98.87	99.26	L2_10	98.82	99.63
H2_20	96.97	96.91	L2_20	99.42	99.06
H2_30	96.79	-	L2_30	99.08	100.00

20x5			**40x5**		
Instance	MILP	MIQP	Instance	MILP	MIQP
H1_10	94.33	-	L1_10	94.92	-
H1_20	93.74	-	L1_20	94.47	-
H2_10	93.52	96.66	L2_20	94.93	-
L2_10	93.87	96.74	L2_30	94.61	-

Table 3.2: *Scaled objective function of the best feasible solutions found by CPLEX in the allowed time limit.*

30x5			**60x13**		
Instance	MILP UB	MIQP UB	Instance	MILP UB	MIQP UB
H1_10	1 754 291	2 288 451	H1_10	3 227 542	5 939 357
H1_20	1 754 633	2 288 793	H1_20	3 228 422	6 038 925
H1_30	1 754 669	2 288 829	H1_30	3 228 709	5 941 943
H2_10	1 708 485	2 256 299	H2_10	3 130 833	5 965 539
H2_20	1 709 020	2 256 834	H2_20	3 131 431	5 966 137
H2_30	1 709 230	2 257 044	H2_30	3 131 677	5 966 383
L1_10	1 420 485	1 787 983	L1_10	3 014 276	5 668 646
L1_20	1 420 713	1 817 824	L1_20	3 014 877	5 669 247
L1_30	1 420 819	1 842 700	L1_30	3 015 054	5 669 424
L2_10	1 613 252	1 948 130	L2_10	3 084 415	5 749 854
L2_20	1 613 769	1 973 914	L2_20	3 085 121	5 750 560
L2_30	1 613 805	2 008 053	L2_30	3 085 364	5 750 803

Table 3.3: *Upper bounds provided by CPLEX using MILP and MIQP formulations.*

The objective function value is scaled to 100 with respect to the upper bound via the formula:

$$\text{scaled obj} = \frac{\text{obj} * 100}{\text{UB}} \tag{3.35}$$

A value of 100 means that the solution is certified to be optimal.

Within class 10x3, 3 out of 12 instances are solved at optimum; both MILP and MIQP formulations provide near-optimal solutions, with an average of 98.44 and 99.11 respectively.

Within class 20x5, a feasible solution is found for 4 instances out of 12 with the MILP formulation, while, using the MIQP formulation, we get a feasible solution only for 2 instances. The quality of the solution is lower, with an average of 93.87 for MILP and of 96.70 for MIQP.

Class 40x5 is only solved using the MILP formulation; a feasible solution is found for 4 instances out of 12, with an average quality of the solution of 94.73.

With respect to the upper bounds, we remark that the MILP formulation provides far better upper bounds than MIQP, as illustrated in Table 3.3.

3.5.3 Heuristic's computational results

The heuristic algorithm is implemented in C++ using GLPK 4.31 and tested on the same set of instances.

Experiments are run for $n \times \bar{p}$ iterations and a time limit of 1 hour for classes 10x3, 20x5, 30x5 and 3 hours for classes 40x5, 50x8, 60x13. The internal tabu search has a maximum of $\eta = 30 \times n$ iterations, and the other parameters are set as follows:

- θ : tabu duration equal to $\lfloor 7.5 \log n \rfloor$;
- β : diversification intensity parameter equal to $0.015\sqrt{nm}$;
- δ : penalty adjustment parameter equal to 2.

Results are compared to the best solution found by CPLEX for either the MILP or MIQP formulation and illustrated in Tables 3.4, 3.5 and 3.6. The heuristic is able to find feasible solutions in 70 out of 72 instances, whereas CPLEX succeeds at finding feasible solutions on only 20 of the smaller instances. The two instances where the heuristic fails at finding a feasible solution are characterized by a high number of profiles per vessel ($\bar{p} = 30$). We observe that with a lower number of profiles per vessel ($\bar{p} = 10$, and $\bar{p} = 20$) the heuristic always succeeds in reaching feasibility. Furthermore, our algorithm is up to 2 order of magnitude faster, especially on small instances.

Class 10x3 is the only one where CPLEX performs slightly better than the heuristic, with an average of 99.00 and 98.59, respectively, and 3 optimums found by CPLEX. However, the heuristic is much faster, solving the problem in less than 30 seconds against the time limit of 1 hour set for CPLEX.

Class 20x5 is always solved by the heuristic in less than 5 minutes, with an average quality of the solution of 97.29, while CPLEX only solves 4 instances out of 12, in 2 hours, with lower quality (95.37 on average).

Remarkably, our heuristic performs very well also on the instances of larger size, where CPLEX generally fails. For the solved instances the quality of the solutions is always greater than 94.11 (instance 60x13:H2_20), with an average value of 96.06.

10x3				**20x5**			
Instance	CPLEX	HEUR	Time (sec)	Instance	CPLEX	HEUR	Time (sec)
H1_10	99.17	98.52	7	H1_10	-	97.26	81
H1_20	97.96	98.36	15	H1_20	94.33	97.19	172
H1_30	98.76	98.33	27	H1_30	93.74	97.37	259
H2_10	99.26	98.92	7	H2_10	-	97.27	82
H2_20	96.97	98.48	16	H2_20	96.66	97.38	173
H2_30	96.79	98.17	28	H2_30	-	97.26	274
L1_10	100.00	99.12	6	L1_10	-	97.30	74
L1_20	100.00	99.01	15	L1_20	-	97.25	158
L1_30	99.99	98.29	26	L1_30	-	97.06	254
L2_10	99.63	98.92	6	L2_10	-	97.55	80
L2_20	99.42	98.68	15	L2_20	96.74	97.39	170
L2_30	100.00	98.22	27	L2_30	-	97.25	295

Table 3.4: *Heuristic's computational results on classes 10x3 and 20x5.*

30x5				**40x5**			
Instance	CPLEX	HEUR	Time (sec)	Instance	CPLEX	HEUR	Time (sec)
H1_10	-	95.67	340	H1_10	-	97.38	1104
H1_20	-	95.31	677	H1_20	-	97.38	2234
H1_30	-	95.54	1009	H1_30	-	97.25	3387
H2_10	-	95.88	316	H2_10	-	97.40	1095
H2_20	-	95.81	684	H2_20	-	97.33	2198
H2_30	-	95.30	969	H2_30	-	97.27	3296
L1_10	-	96.55	324	L1_10	94.92	97.41	1421
L1_20	-	96.43	652	L1_20	94.47	97.14	2996
L1_30	-	96.18	966	L1_30	-	96.20	4862
L2_10	-	95.68	308	L2_10	-	97.41	1382
L2_20	-	95.12	614	L2_20	94.93	97.34	3144
L2_30	-	-	920	L2_30	94.61	96.60	4352

Table 3.5: *Heuristic's computational results on classes 30x5 and 40x5.*

50x8				60x13			
Instance	CPLEX	HEUR	Time (sec)	Instance	CPLEX	HEUR	Time (sec)
H1_10	-	96.52	3291	H1_10	-	95.40	6332
H1_20	-	96.37	6020	H1_20	-	95.07	10809
H1_30	-	96.21	9432	H1_30	-	94.76	10807
H2_10	-	96.03	3066	H2_10	-	95.54	6397
H2_20	-	95.64	6180	H2_20	-	94.11	10803
H2_30	-	95.16	9501	H2_30	-	-	10806
L1_10	-	95.97	2752	L1_10	-	95.67	5807
L1_20	-	96.04	6467	L1_20	-	95.40	10803
L1_30	-	95.80	9119	L1_30	-	94.45	10806
L2_10	-	96.18	3157	L2_10	-	95.63	5986
L2_20	-	95.96	5857	L2_20	-	95.64	10809
L2_30	-	96.27	8783	L2_30	-	95.34	10804

Table 3.6: *Heuristic's computational results on classes 50x8 and 60x13.*

3.6 Conclusions

We have studied the integration, at the tactical level, of the berth allocation problem with the assignment of quay cranes from the point of view of a container terminal, in the context of a negotiation process with shipping lines.

We have characterized this new decision problem and illustrated the concept of QC assignment profiles. Two mixed integer programming formulations have been presented, with a quadratic and a linearized objective function respectively. Both models have been validated on instances based on real data using a commercial solver.

These tests show that the problem is hardly solvable already on small instances; furthermore, we noticed that instances present symmetries that slow down the solution process. Hence we have tackled the computational complexity of TBAP by devising a two-level heuristic algorithm able to provide good feasible solutions in a reasonable amount of time.

As a next step, we are interested in developing an exact methods to provide optimal solutions and tighter upper bounds to the problem; decomposition methods seem to be a promising way to face the problem.

Chapter 4

An exact algorithm for the TBAP

This chapter is organized in two main sections that address both algorithmic and modeling aspects of the Tactical Berth Allocation Problem.

In Section 4.1 we propose an exact branch-and-price algorithm for Tactical Berth Allocation Problem and we present several accelerating techniques for the pricing and the master problem devised for this specific problem. In particular, some of these techniques can be easily generalized for other branch-and-price schemes.

In Section 4.2 we present a comparison between hierarchical and integrated planning models. The exact algorithm developed for the TBAP enables us to perform a comparative analysis of hierarchical vs integrated solution approaches for berth allocation and quay crane assignment, and we show the added value of integration in terms of cost reduction and efficient use of resources.

4.1 Branch-and-price for the TBAP

In this section we introduce the basic concepts of column generation and branch-and-price schemes. The TBAP is reformulated via Dantzig-Wolfe in section 4.1.2 and the column generation scheme is illustrated in section 4.1.3. The implementation of the branch-and-price algorithm and the devised accelerating techniques are described in section 4.1.4, while computational results are discussed in section 4.1.5.

4.1.1 Introduction

Dantzig-Wolfe (DW) decomposition was introduced by Dantzig and Wolfe (1960) in their seminal paper on decomposition principles for linear programs. The basic idea is to break the problem into smaller *subproblems* that can be solved independently at the lower level: solutions (*columns*) are then coordinated and combined at the upper level by a linear program called *master problem*, that can be seen as a centralized decision-maker. The solution method, called *column generation*, allows to cope with a huge number of variables, since only promising columns are added to the master

Applications	*References*
bin packing and cutting stock	Vance et al. (1994), Valerio de Carvalho (1999), Vanderbeck (1999), Alves and Valerio de Carvalho (2008);
integer multicommodity flows	Barnhart et al. (2000),Holberg and Yuan (2003);
location problems	Diaz and Fernandez (2002), Ceselli and Righini (2005), Senne et al. (2005);
vehicle routing	Desrochers et al. (1992), Ribeiro and Soumis (1994), Desaulniers et al. (1998), Salani (2006), Dell'Amico et al. (2006), Fukasawa et al. (2006);
crew scheduling	Desrochers and Soumis (1989), Vance et al. (1997), Cohn and Barnhart (2003), Huisman (2007);
airline operations	Gamache et al. (1999), Klabjan (2005), Lan et al. (2006), Eggenberg (2009);
railways operations	Ceselli et al. (2008), Peeters and Kroon (2008);
maritime transport	Hwang et al. (2008), Brønmo et al. (2010), Grønhaug et al. (2010).

Table 4.1: *Column generation and branch-and-price applications.*

problem: decision is based on the reduced cost of columns, that are priced out in the subproblem (*pricing*).

Authors give credit to Ford and Fulkerson (1958), who propose a method to solve a multicommodity flow problem by treating variables only implicitly in the solution process. This technique was firstly used by Gilmore and Gomory (1961; 1963) to solve the cutting-stock problem, and was successfully embedded in a branch-and-bound algorithm by Desrosiers et al. (1984) for solving a vehicle routing problem with time windows.

Over the last two decades, Dantzig-Wolfe decomposition and column generation for integer programs have been widely studied (Vanderbeck and Wolsey, 1996; Barnhart et al., 1998) and applied to a variety of problems. A non-exhaustive list of major applications and references, with emphasis on transportation problems, is reported in Table 4.1. The general framework for solving integer programs is called *branch-and-price* and combines column generation and branch-and-bound techniques. At each node of the search tree, the master problem is initialized with a restricted number of columns (restricted master problem) such that a feasible solution of the linear relaxation exists. Assuming to solve a minimization problem, at each iteration the pricing subproblem provides the minimum reduced cost column with respect to the current dual master solution: if this column has a non-negative reduced cost, then the current master solution is proved to be optimal and we stop; otherwise, the column is added to the master and we iterate the column generation process. The method has proved to be very efficient and presents several advantages.

The Dantzig-Wolfe reformulation usually yields to better linear-relaxation bounds.

The main assumption to apply DW decomposition is that the *original* or *compact* formulation has a special constraint-matrix structure that can be exploited. Consider the following integer linear program:

$$z_{IP} = \min \quad c^T x \tag{4.1}$$
$$\text{s.t.} \quad Ax \geq b, \tag{4.2}$$
$$x \in X, \tag{4.3}$$
$$x \text{ binary .} \tag{4.4}$$

with X bounded. We assume that set $X^* = \{x \in X : x \text{ binary }\}$ has a particular structure that can be "convexified": it means that x can be expressed as a convex combination of the extreme points of the convex hull $conv(X^*)$. The DW reformulation is usually referred to as *extensive formulation*. The linear relaxation of the extensive formulation is then solved via column generation: the objective function $\min c^T x$ and constraints $Ax \geq b$ are taken into account in the master problem, where they are expressed in terms of extreme points of $conv(X^*)$; the pricing subproblem accounts for the generation of extreme points (columns) according to reduced cost arguments.

As mentioned, the linear relaxation of the DW reformulation, i.e., the optimal master problem, typically provides better bounds than the linear relaxation of the original formulation. The gain in terms of bounds is due to convexification, since the subproblem takes into account the integrality requirements on x. In fact, it is desirable to have a subproblem without the *integrality property* in order to have the potential to exploit the integrality gap (Lübbecke and Desrosiers, 2005); otherwise the reformulation doesn't yield any improvement on the bound. This also holds for Lagrangian relaxation (Geoffrion, 1974); a typical example is the shortest path problem. We recall that an integer program satisfies the integrality property if the optimal solution is unchanged when the integrality requirements on the variables are relaxed.

One may argue that the complexity is simply moved from the original formulation to the subproblem, since we are still solving an integer program. On the contrary, one of the main advantages of column generation is to enhance the solution process by identifying subproblems with special structures, that are well studied in the literature and for which efficient (pseudo-polynomial) specialized algorithms are available. We can identify two main classes of subproblems.

Knapsack subproblems Typically originated by cutting stock, bin packing and location problems. The *Knapsack Problem* is NP-hard but it can be solved in pseudo-polynomial time via dynamic programming (Martello and Toth, 1990). Furthermore, many efficient algorithms are available nowadays that show linear computing time in practice (Martello et al., 2000; Pisinger, 2005).

Shortest path subproblems with resource constraints Usually associated with vehicle routing and crew scheduling problems. The *Resource Constrained Shortest Path Problem* (RCSPP) is NP-Complete even in the case of one resource (Garey and Johnson, 1979; Handler and Zang, 1980) but it can be solved in pseudo-polynomial time via dynamic programming. However, many applications require elementarity of paths to be satisfied, i.e., no nodes must appear more than once in the path. Opposite to the non-elementary case, the *Resource Constrained Elementary Shortest Path Problem* (RCESPP) is strongly NP-Hard when negative cost cycles may appear (Dror, 1994). Dynamic programming represents the dominant solution method for the RCESPP, and effective accelerating techniques have improved the label algorithm in the recent years (Boland et al., 2006; Righini and Salani, 2006; Righini and Salani, 2008). However, when long paths with respect to the number of visited nodes are feasible for the problem, dynamic programming can be very time consuming. We remark that resources allow to model many real-world features of problems, such as complex costs structures, operational rules and constraints, non-linearities (Irnich, 2008).

4.1.2 Dantzig-Wolfe reformulation

Our reformulation of the Tactical Berth Allocation Problem is mainly based on the concept of *berth sequence*, a sequentially ordered subset of ships in a berth with an assigned quay crane profile.

The MILP formulation introduced in section 3.3.4 represents the original or compact formulation, that we decompose according to Dantzig-Wolfe. In particular, for every berth $k \in M$ we convexify the subset $R^k = \{(x, y, \lambda, T, \gamma, \rho)$ such that $(3.3) - (3.17), (3.19) - (3.24)$ is satisfied for $k\}$. The set R^k is represented by a finite set of vectors Ω^k, that are the extreme points of its convex hull $conv\{R^k\}$. Each extreme point $r_k \in \Omega^k$ represents a sequence of vessels moored at berth k, feasible with respect to time windows constraints and with a unique quay crane profile assigned to each vessel.

As mentioned in the introduction, the basic concept of the Dantzig-Wolfe reformulation is that every point $(x, y, \lambda, T, \gamma, \rho) \in R^k$ can be represented as a convex combination of extreme points $r_k \in \Omega^k$.

In order to reformulate the problem, we define the following additional notation:

- x_{ijr_k} binary coefficient equal to 1 if vessel j follows vessel i in sequence r_k;
- y_{ir_k} binary coefficient equal to 1 if vessel i is moored at berth k in sequence r_k;
- $\lambda^p_{ir_k}$ binary coefficient equal to 1 if profile p is assigned to vessel i in sequence r_k;
- $q^h_{r_k}$ counts the number of quay cranes used by sequence r_k at time step h;
- ν_{r_k} value of sequence r_k, defined as $\nu_{r_k} = \sum_{i \in N} \sum_{p \in P_i} \lambda^p_{ir_k} \nu^p_i$.

By applying Dantzig-Wolfe decomposition to the original formulation for TBAP,

we obtain the following extensive formulation:

$$\min \sum_{i,j\in N}\sum_{k,w\in M} f_{ij}d_{kw}z_{ij}^{kw} - \sum_{k\in M}\sum_{r_k\in\Omega^k} v_{r_k}s_{r_k} \tag{4.5}$$

$$\sum_{k\in M}\sum_{r_k\in\Omega^k} y_{ir_k}s_{r_k} = 1 \quad \forall i \in N, \tag{4.6}$$

$$\sum_{k\in M}\sum_{r_k\in\Omega^k} q_{r_k}^h s_{r_k} \le Q^h \quad \forall h \in H, \tag{4.7}$$

$$\sum_{r_k\in\Omega^k} s_{r_k} \le 1 \quad \forall k \in M, \tag{4.8}$$

$$\sum_{k\in M}\sum_{w\in M} z_{ij}^{kw} = g_{ij} \quad \forall i,j \in N, \tag{4.9}$$

$$\sum_{r_k\in\Omega^k} y_{ir_k}s_{r_k} - z_{ij}^{kw} \ge 0 \quad \forall i,j \in N, \forall k,w \in M, \tag{4.10}$$

$$\sum_{r_w\in\Omega^w} y_{jr_w}s_{r_w} - z_{ij}^{kw} \ge 0 \quad \forall i,j \in N, \forall k,w \in M, \tag{4.11}$$

$$\sum_{r_k\in\Omega^k} x_{ijr_k}s_{r_k} = x_{ij}^k \quad \forall i,j \in N, \forall k \in M, \tag{4.12}$$

$$\sum_{r_k\in\Omega^k} y_{ir_k}s_{r_k} = y_i^k \quad \forall i \in N, \forall k \in M, \tag{4.13}$$

$$\sum_{k\in M}\sum_{r_k\in\Omega^k} \lambda_{ir_k}^p s_{r_k} = \lambda_i^p \quad \forall i \in N, \forall p \in P_i, \tag{4.14}$$

$$s_{r_k} \ge 0 \quad \forall r_k \in \Omega^k, \forall k \in M, \tag{4.15}$$

$$x_{ij}^k \in \{0,1\} \quad \forall k \in M, \forall (i,j) \in A^k, \tag{4.16}$$

$$y_i^k \in \{0,1\} \quad \forall k \in M, \forall i \in N, \tag{4.17}$$

$$\lambda_i^p \in \{0,1\} \quad \forall p \in P_i, \forall i \in N, \tag{4.18}$$

$$z_{ij}^{kw} \in \{0,1\} \quad \forall i,j \in N, \forall k,w \in M. \tag{4.19}$$

where s_{r_k} are the decision variables associated with sequences $r_k \in \Omega^k$.

The objective function (4.5) is equivalent to (3.28) and maximizes the total value of sequences, i.e., the total value of selected profiles, while minimizing the total housekeeping cost generated by the berth allocation plan. Constraints (4.6) ensure that every ship is assigned to exactly one sequence, and thus to one berth, while constraints (4.7) ensure that the quay crane capacity is not violated. Constraints (4.8) select at most one sequence for each berth, while (4.9)–(4.11) express the linearization constraints (3.25)-(3.27) in terms of decision variables s_{r_k}. As mentioned, the decision variables of the original formulation are expressed as a convex combination of extreme points of $conv\{R^k\}$ in constraints (4.12)-(4.15). Finally, the integrality of the solution is ensured by constraints (4.16)-(4.19).

4.1.3 Column generation

We solve the linear relaxation of the extensive formulation via column generation, by the means of a master problem and a pricing subproblem.

Master problem

The linear relaxation of the extensive formulation (4.5)-(4.19) usually yields to better bounds than the original formulation. In particular, if we relax the integrality requirements (4.16)-(4.19), constraints (4.12)-(4.14) also become redundant and we obtain the following master problem:

$$\min \sum_{i,j\in N} \sum_{k,w\in M} f_{ij} d_{kw} z_{ij}^{kw} - \sum_{k\in M} \sum_{r_k\in\Omega^k} v_{r_k} s_{r_k} \tag{4.20}$$

$$\sum_{k\in M} \sum_{r_k\in\Omega^k} y_{ir_k} s_{r_k} = 1 \quad \forall i \in N, \tag{4.21}$$

$$\sum_{k\in M} \sum_{r_k\in\Omega^k} q_{r_k}^h s_{r_k} \leq Q^h \quad \forall h \in H, \tag{4.22}$$

$$\sum_{r_k\in\Omega^k} s_{r_k} \leq 1 \quad \forall k \in M, \tag{4.23}$$

$$\sum_{k\in M} \sum_{w\in M} z_{ij}^{kw} = g_{ij} \quad \forall i,j \in N, \tag{4.24}$$

$$\sum_{r_k\in\Omega^k} y_{ir_k} s_{r_k} - z_{ij}^{kw} \geq 0 \quad \forall i,j \in N, \forall k,w \in M, \tag{4.25}$$

$$\sum_{r_w\in\Omega^w} y_{jr_w} s_{r_w} - z_{ij}^{kw} \geq 0 \quad \forall i,j \in N, \forall k,w \in M, \tag{4.26}$$

$$z_{ij}^{kw} \geq 0 \quad \forall i,j \in N, \forall k,w \in M, \tag{4.27}$$

$$s_{r_k} \geq 0 \quad \forall r_k \in \Omega^k, \forall k \in M. \tag{4.28}$$

The resulting linear program usually involves a huge number of variables (columns). Therefore, the column generation scheme starts solving a *restricted master problem*, defined on a subset of columns and, at each iteration, it generates new profitable columns to be added to the formulation, if any.

Pricing subproblem

Let π, μ, ξ, θ and η be the dual vectors associated with constraints (4.21), (4.22), (4.23), (4.25) and (4.26), respectively. Given an optimal solution of the restricted

master problem, the reduced cost of sequence $r \in \Omega^k$ is given by:

$$\tilde{v}_{r_k} = -v_{r_k} - \sum_{i \in N} \pi_i y_{ir_k} - \sum_{h \in H} \mu^h q^h_{r_k} - \xi_k - \sum_{i,j \in N} \sum_{w \in M} \theta^{kw}_{ij} y_{ir_k} - \sum_{i,j \in N} \sum_{w \in M} \eta^{kw}_{ij} y_{jr_w}.$$

The pricing subproblem identifies, for every berth $k \in M$, the column r^*_k with the minimum reduced cost and can be formulated as follows:

$$\min \; -\sum_{i \in N} (v^p_i \lambda^p_i + \pi_i y_i) - \sum_{i \in N} \sum_{p \in P_i} \sum_{h=1 \ldots t^p_i} \mu^{h+T_i} q^{ph}_i \lambda^p_i - \xi_k - \sum_{i,j \in N} \sum_{w \in M} (\theta^{kw}_{ij} y_i + \eta^{kw}_{ij} y_j) \tag{4.29}$$

$$\sum_{j \in N \cup \{d\}} x_{o,j} = 1, \tag{4.30}$$

$$\sum_{i \in N \cup \{o\}} x_{i,d} = 1, \tag{4.31}$$

$$\sum_{j \in N \cup \{d\}} x_{ij} - \sum_{j \in N \cup \{o\}} x_{ji} = 0 \qquad \forall i \in N, \tag{4.32}$$

$$\sum_{j \in N \cup \{d\}} x_{ij} = y_i \qquad \forall i \in N, \tag{4.33}$$

$$T_i + \sum_{p \in P_i} t^p_i \lambda^p_i - T_j \leq (1 - x_{ij}) M1 \qquad \forall i \in N, \forall j \in N \cup \{d(k)\}, \tag{4.34}$$

$$T_o - T_j \leq (1 - x_{o,j}) M2 \qquad \forall j \in N, \tag{4.35}$$

$$a_i y_i \leq T_i \qquad \forall i \in N, \tag{4.36}$$

$$T_i \leq b_i y_i \qquad \forall i \in N, \tag{4.37}$$

$$a^k \leq T_o, \tag{4.38}$$

$$T_d \leq b^k, \tag{4.39}$$

$$\sum_{p \in P_i} \lambda^p_i = y_i \qquad \forall i \in N, \tag{4.40}$$

$$\sum_{h \in H^s} \gamma^h_i = \sum_{p \in P^s_i} \lambda^p_i \qquad \forall i \in N, \forall s \in S, \tag{4.41}$$

$$T_i - b^h \leq (1 - \gamma^h_i) M3 \qquad \forall h \in H, \forall i \in N, \tag{4.42}$$

$$a^h - T_i \leq (1 - \gamma^h_i) M4 \qquad \forall h \in H, \forall i \in N, \tag{4.43}$$

$$x_{ij} \in \{0, 1\} \qquad \forall (i, j) \in A^k, \tag{4.44}$$

$$y_i \in \{0, 1\} \qquad \forall i \in N, \tag{4.45}$$

$$\gamma^h_i \in \{0, 1\} \qquad \forall h \in H, \forall i \in N, \tag{4.46}$$

$$\lambda^p_i \in \{0, 1\} \qquad \forall p \in P_i, \forall i \in N, \tag{4.47}$$

$$T_i \geq 0 \qquad \forall i \in N \cup \{o, d\}, \tag{4.48}$$

where M1, M2, M3 and M4 represent large positive constants.

We remark that the index k has disappeared from decision variables x, y and T with respect to the notation introduced in section 3.3.3, since the pricing subproblem is solved for a fixed berth k. For the description of constraints and decision variables, we refer the reader to section 3.3.3.

At each iteration of column generation, we solve m subproblems, one for every berth $k \in M$. If $\tilde{v}_{r_k^*} < 0$ for some k, we add column $s_{r_k^*}$ to the restricted master problem and we iterate the process; otherwise, the current solution of the master problem is proven to be optimal and we stop.

Dynamic programming The pricing subproblem (4.29)-(4.48) can be cast to a Resource Constrained Elementary Shortest Path Problem, where the resource is represented by time, and it is solved by the means of dynamic programming. The underlying network $G(V, A)$ has one vertex for every vessel $i \in N$, for every profile $p \in P_i$ and for every time step $h \in H$, and transit time on arcs equal to the length of profile p assigned to vessel i. Vertex (i, h, p) represents vessel i berthed at time step h and operated by quay crane profile p. The graph has two additional vertices o, d associated with the specific berth $k \in M$ for which the pricing is solved, representing the origin and the destination of the path.

The RCESPP aims to find a minimum-cost elementary path from o to d that satisfies the constraints on resources: the objective function of the RCESPP associated with the pricing subproblem corresponds to (4.29), while the resource constraint requires not to exceed the given time horizon. We remark that the resulting graph presents negative arc costs, as we are minimizing the reduced cost.

The dynamic programming (DP) algorithm iteratively extends states. A *state* for vertex (i, h, p) represents a path from o to (i, h, p); many states are associated with the same vertex (i, h, p), representing different paths. Each state is encoded by a *label* of the form (S, τ, C, i, h, p), that is a path from o to (i, h, p) with time consumption τ and cost C; furthermore, to ensure elementarity, set S keeps tracks of vessels visited along the path (Beasley and Christofides, 1989). The optimal solution is given by the minimum cost state associated with the destination vertex d.

At vertex o, time consumption τ is initialized at 0 and $S = \{o\}$; cost C is initialized to ξ_k, according to the berth k for which the pricing problem is being solved. When extending state $(S, \tau, C, i, h_i, p_i)$ to another feasible state $(S', \tau', C', j, h_j, p_j)$, the label is updated according to the formula:

$$S' = S + \{j\} \tag{4.49}$$

$$\tau' = h_j + t_j^{p_j} \tag{4.50}$$

$$C' = C - \nu_{p_j} - \pi_j - \sum_{h=h_j}^{h_j + t_j^{p_j}} \mu^h q_j^{p_j(h-h_j)} - \sum_{n \in N} \sum_{w \in M} (\theta_{jn}^{kw} + \eta_{nj}^{wk}) \tag{4.51}$$

The extension is feasible if $j \notin S$ (elementarity), $\tau' < |H|$ (total duration) and h_j satisfies time windows $[a_j, b_j]$.

The efficiency of dynamic programming strongly depends on the effectiveness of *dominance* rules that are used to fathom feasible, yet non-optimal states. In particular, dominated states are not extended further.

State $(S', \tau', C', j, h_j, p_j)$ dominates $(S'', \tau'', C'', j, h_j, p_j)$ if:

$$|S'| \leq |S''| \tag{4.52}$$
$$\tau' \leq \tau'' \tag{4.53}$$
$$C' \leq C'' \tag{4.54}$$

and at least one of these inequalities is strictly satisfied.

4.1.4 Implementation

In order to obtain integer solutions, we implement a branch-and-price algorithm where column generation is applied at every node of the search tree. The search tree is explored according to a best-first strategy with respect to the lower bound associated with the node. The algorithm makes use of a column pool that keeps track of all columns generated in different nodes of the search tree.

In the remainder of this section we illustrate the branching rules as well as accelerating techniques both for the pricing and the master problem.

Branching scheme

In the search tree, branching is required when the master problem is solved at optimality and the corresponding solution in terms of original formulation's variables is not integer. We implement a branching scheme consisting of four hierarchical levels:

1. if the total number of berths $\widetilde{K} = \sum_{k \in M} \sum_{r_k \in \Omega^k} s_{r_k}$ is fractional, then branching requires an additional constraint to be added in the master problem:

   - $\sum_{k \in M} \sum_{r_k \in \Omega^k} s_{r_k} \leq \lfloor \widetilde{K} \rfloor$ on the first child node;
   - $\sum_{k \in M} \sum_{r_k \in \Omega^k} s_{r_k} \geq \lceil \widetilde{K} \rceil$ on the second child node.

   This branching requires the dual value associated with the additional constraint, denoted by π_0, to be collected and accounted in the pricing subproblem. We remark that π_0 is a constant, regardless of the berth. In particular, the additional constraint in the master problem modifies the objective function of the pricing subproblem as follows:

$$\min \nu_{r_k} - \sum_{i \in N} \pi_i y_{ir_k} - \sum_{h \in H} \mu^h q^h_{r_k} - \xi_k - \sum_{i,j \in N} \sum_{w \in M} (\theta^{kw}_{ij} y_{ir_k} + \eta^{kw}_{ij} y_{jr_w}) - \pi_0.$$

2. if some vessel $i \in N$ is assigned fractionally to some berth $k \in M$, i.e., quantity $\widetilde{Y}_i^k = \sum_{r_k \in \Omega^k} y_i^{r_k} s_{r_k}$ is fractional, then the branching requires an additional constraint to be added to the master problem for the given vessel i and berth k:

   - $\sum_{r_k \in \Omega^k} y_i^{r_k} s_{r_k} = 0$ on the first child node;
   - $\sum_{r_k \in \Omega^k} y_i^{r_k} s_{r_k} = 1$ on the second child node.

   This branching requires the dual value associated with the additional constraint, denoted by φ_i^k, to be taken into account in the pricing subproblem. We remark that φ_i^k is collected in the pricing for berth k if vessel i is visited by the sequence. In particular, the additional constraints in the master problem modify the objective function of the pricing subproblem as follows:

$$\min \nu_{r_k} - \sum_{i \in N} \pi_i y_{ir_k} - \sum_{h \in H} \mu^h q_{r_k}^h - \xi_k - \sum_{i,j \in N} \sum_{w \in M} (\theta_{ij}^{kw} y_{ir_k} + \eta_{ij}^{kw} y_{jr_w}) - \pi_0 - \sum_{i \in N} \varphi_i^k y_{ir_k}.$$

3. if some profile $p \in P_i$ is assigned fractionally to some vessel $i \in N$, i.e., quantity $\widetilde{\Lambda}_i^p = \sum_{k \in M} \sum_{r_k \in \Omega^k} \lambda_i^{pr_k} s_{r_k}$ is fractional, then branching is handled directly in the pricing subproblem by modifying the set P_i of feasible profiles for vessel i. On the first node child, we enforce profile p to be assigned to vessel i by removing all other feasible profiles from set P_i; this branching corresponds to enforce $\lambda_i^p = 1$ in the original formulation. On the second child node, we prevent profile p to be used by removing it by set P_i; this branching corresponds to enforce $\lambda_i^p = 0$ in the original formulation. We remark that neither the master nor the pricing formulation is modified by this branching in terms of objective function and additional constraints. However, infeasible columns must be removed from the master problem, accordingly to the branching decision associated with the analyzed node.

4. if none of the above conditions holds, then there exist some vessel $i \in N$ such that the quantity $\widetilde{T}_i^h = \sum_{k \in M} \sum_{r_k \in \Omega^k} T_i^{hr_k} s_{r_k}$ is fractional for some $h^* \in H$, where $T_i^{hr_k}$ is a binary coefficient equal to 1 if vessel i arrives at time step h in sequence r_k. In this case, branching is handled in the pricing subproblem, by modifying the time windows $[a_i, b_i]$ associated with vessel i (Gélinas et al., 1995). We denote as t_i^* the arrival time associated with time step h^*. On the first child node, we enforce the vessel to arrive before time step h^*: the new time windows for vessel i are therefore $[a_i, t_i^* - \epsilon]$ and this corresponds to enforce $\gamma_i^h = 0 \ \forall h \geq h^*$ in the original formulation. On the second child node, we enforce the vessel to arrive at or after time step h^*: the new time windows for vessel i are therefore $[t_i^*, b_i]$ and this corresponds to enforce $\gamma_i^h = 0 \ \forall h <$

h^* in the original formulation. We remark that neither the master nor the pricing formulation is modified by this branching in terms of objective function and additional constraints. However, infeasible columns must be removed from the master problem, accordingly to the branching decision associated with the analyzed node.

The order of branching is determined by the increasing complexity of the branching rules for what concerns additional constraints in the master problem or additional complexity of the pricing problem.

Accelerating techniques

Solving exact dynamic programming We implement state-of-the-art techniques for solving the RCESPP such as bounded bidirectional dynamic programming and decremental state space relaxation.

Bounded bidirectional DP (Righini and Salani, 2006) consists of two steps: firstly, states are extended in forward and backward direction until half of the so-called *critical resource* (time, in our case) is consumed; secondly, forward and backward paths are joined to produce feasible sequences. Bounding is used to discard non-dominated non-optimal states.

The basic idea of decremental state space relaxation (Righini and Salani, 2008) is to start checking elementarity only on a subset $\bar{S}$ of S. If the final solution is non-elementary, one or more vertices violating the constraint are added to $\bar{S}$ and DP is executed again.

The implemented search policy takes into account time windows (Liberatore et al., 2010). At every iteration of dynamic programming, states are explored according to the vertices (vessels) they are associated with. We decide to order vessels according to the starting time a_i of their time windows; this search strategy proves to be important for the effectiveness of the algorithm in our tests.

Furthermore, we design an additional technique for accelerating the exact pricing, that is specifically conceived for our pricing subproblem.

Domination of (h,p) pairs. Unlike RCESPP subproblems arising in vehicle routing, where customers are visited one right after the other, in our problem it may be convenient to wait some time between the departure of a vessel and the arrival of the next one. This is due to the quay crane capacity constraint, that control the interactions between berths at the master problem level; in particular, these interactions are captured by dual vectors θ and η. More specifically, when extending a label to the next vessel j, we have as many new states as the number of feasible arrival time steps h_j; furthermore, we may have more than one profile p_j associated with a single time step h_j and viceversa. In order to reduce the number of states, preprocessing is performed at the beginning of the DP algorithm: we populate a list of non-dominated (h_j, p_j) pairs for every vessel j and we refer to this list when extending a label to vessel j.

We remark that in the special case of $\theta = 0$ and $\eta = 0$, the list has at most one pair (h_j, p_j) for every profile p_j.

Heuristic pricing The pricing subproblem is firstly solved heuristically. An exact solution is computed only if needed.

The heuristic dynamic programming algorithm (Liberatore et al., 2010) is based on a relaxed dominance rule that allows to eliminate much more states during the comparison of labels. The final solution is an elementary shortest path that satisfies resource constraints; however, optimality is no longer guaranteed. When using relaxed dominance, we have that state $(S', \tau', C', j, h_j, p_j)$ dominates $(S'', \tau'', C'', j, h_j, p_j)$ if:

$$\tau' \leq \tau'' \tag{4.55}$$
$$C' \leq C'' \tag{4.56}$$

and at least one of these inequalities is strictly satisfied. In other words, we do not compare anymore the number of vertices $|S'|, |S''|$ visited by the partial paths. As the dominance is weaker, the number of eliminated labels is greater. This results in a reduced computational effort to solve the pricing.

Furthermore, the following accelerating techniques are implemented with the main purpose of avoiding the call to exact pricing as much as possible.

Multiple pricing strategy. At every iteration of column generation, we firstly solve a pricing subproblem for every berth $k \in M$ using the heuristic dynamic programming algorithm; exact DP is called only if heuristic pricing cannot provide a negative reduced cost column. As soon as we find a negative reduced cost column for some berth k^*, the pricing terminates. However, columns generated for k^* are evaluated for all berths $k \neq k^*, k \in M$: if a column is feasible for another berth, say $\bar{k}$, and its reduced cost re-computed for berth $\bar{k}$ is negative, then the column is duplicated and added to the master problem also for berth $\bar{k}$.

Incremental heuristic dynamic programming. The basic idea is to incrementally strengthen the relaxed dominance rule introduced for the heuristic pricing, in order to increase the probability of finding a negative reduced cost column and therefore avoid calling exact DP. We define the set of *critical vertices* $\widetilde{N} \subset N$ for which exact dominance is required, similarly to decremental state space relaxation (Righini and Salani, 2008); the set $\widetilde{N}$ is initialized with the empty set, and it is iteratively incremented until a given percentage δ of vertices is reached. At each iteration, βN critical vertices are chosen among those visited more than once by the resulting path and added to $\widetilde{N}$. The dominance rule is the one described in section 4.1.3, except for the definition of set S: a vertex j belongs to S if it is visited by the partial path and if $j \in \widetilde{N}$. The first iteration, when $\widetilde{N} = \{\emptyset\}$, corresponds to the heuristic DP algorithm outlined at the beginning of this subsection; the special case of $\delta = 1$ corresponds to exact dynamic programming. In our tests, we fix $\beta = 0.2$ and $\delta = 0.4$.

Dual stabilization Column generation is known to suffer slow convergence (*tailing-off effect*) mainly due to stability problem. Degeneracy of the master problem implies an infinite number of dual optimal solutions: the simplex method typically provides an extreme dual optimal vector, whereas interior dual vectors could be more suitable for generating good paths in the pricing subproblem. Stabilization methods try to overcome this issue by providing a better approximation of optimal dual values (du Merle et al., 1999; Rousseau et al., 2007).

Our stabilized version of column generation is inspired by Addis et al. (2009). The basic idea is the following: a dual optimal solution π to the restricted master problem can be either feasible, and thus optimal, or infeasible for the dual of the complete master problem. We are mainly interested in pricing out with a dual vector close to the optimal dual, thus close to feasibility.

We define the *stability center* $\bar{\pi}$ that represents our current best guess for the optimal dual. At each iteration of column generation, we modify the dual vector provided by the restricted master problem and we obtain a new vector $\tilde{\pi}$ that we use in the pricing problem. The update formula is clear and simple:

$$\tilde{\pi} = \alpha\pi + (1 - \alpha)\bar{\pi} \tag{4.57}$$

where α is a parameter between 0 and 1. If no negative reduced cost columns are found with a given $\tilde{\pi}$, the value of α is increased by step α_k; furthermore, the current $\tilde{\pi}$ is feasible and improving, therefore we update the stability center. The process is repeated until $\alpha = 1$ and no negative reduced cost columns can be found.

In our experiments, we set $\alpha = 0.5$ and $\alpha_k = 0.1$.

Primal heuristic Integer feasible solution are rarely produced in column generation, as the optimal solutions of restricted master problems are typically fractional. Therefore we implement a primal heuristic in order to identify feasible integer solutions during the search process: the main purpose is to improve the primal bound, and thus increase the pruning in the search tree.

The heuristic algorithm takes as input a fractional optimal solution to a restricted master problem and identifies the variable $s^*_{r_k}$ with the highest fractional value strictly lower than 1; variable $s^*_{r_k}$ is set equal to 1 and the linear program is solved again. The procedure is repeated until either a integer solution is found or the linear problem becomes infeasible.

Although very simple, the primal heuristic has proved to be helpful in finding integer solutions especially for larger instances.

4.1.5 Computational results

In this section we provide computational results for the Tactical Berth Allocation Problem.

The branch-and-price algorithm for the TBAP is implemented in C++ and compiled with gcc 4.1.2. All restricted master problems are solved using ILOG CPLEX version 12. Computational experience is run under a linux operating system on a 2Ghz Intel processor equipped with 2GB of RAM.

Although in this chapter the TBAP is formulated as a minimization problem for simplicity of illustration, the results are expressed in terms of a maximization problem, in order to be consistent with the original formulation presented in Chapter 3.

Instances

Computational experiments are performed on instances derived by the test set introduced in Chapter 3.

As already remarked, those instances present symmetries and this usually slows down the convergence of exact algorithms, as proving optimality may be very difficult. We observe that a source of symmetry is given by the cost structure associated with berths (d_{kw}). The cost matrix, for an illustrative example of $|M| = 3$ berths, shows the following structure:

d_{kw}	1	2	3
1	a	b	c
2	b	a	b
3	c	b	a

with $a < b$ and $b < c$. Indeed, this cost matrix only takes into account relative distances between couples of berths and it originates a lot of symmetries in the problem. Therefore we break the symmetry in order to speed up the solution process. In particular, without loss of generality, we perturbate the diagonal of the cost matrix in order to give priority to the first berth, then to the second berth, and so on. This can always be imposed when the time windows $[a^k, b^k]$ on berths are equivalent, since it means that berths are identical. The resulting perturbated matrix has the following structure:

$\tilde{d}_{kw}$	1	2	3
1	a	b	c
2	b	a+ϵ	b
3	c	b	a+2ϵ

In order to maintain equivalence of optimal solutions, it is sufficient to ensure that $a + (|M| - 1)\epsilon < b$ by choosing an appropriate positive small value for ϵ.

The effects of perturbation on symmetry are illustrated by the example in Figure 4.1, where four optimal solutions for TBAP are defined. Configurations A, B, C and D are equivalent with respect to the original cost matrix. On the contrary, with the perturbated cost matrix, solutions C and D are discarded a priori, since the couple of berths (1,2) is always preferred to the couple (2,3). Furthermore, depending on

CONFIGURATION A

TIME	h=1	h=2	h=3	h=4	h=5	h=6	h=7	h=8
berth 1	ship 1				ship 2			
	3	2	2		4	4	5	5
berth 2		ship 3				ship 4		
		4	5			3	3	3
berth 3								
QCs	3	6	7	0	4	7	8	8

CONFIGURATION B

TIME	h=1	h=2	h=3	h=4	h=5	h=6	h=7	h=8
berth 1		ship 3				ship 4		
		4	5			3	3	3
berth 2	ship 1				ship 2			
	3	2	2		4	4	5	5
berth 3								
QCs	3	2	2	0	4	4	5	5

CONFIGURATION C

TIME	h=1	h=2	h=3	h=4	h=5	h=6	h=7	h=8
berth 1								
berth 2	ship 1				ship 2			
	3	2	2		4	4	5	5
berth 3		ship 3				ship 4		
		4	5			3	3	3
QCs	3	6	7	0	4	7	8	8

CONFIGURATION D

TIME	h=1	h=2	h=3	h=4	h=5	h=6	h=7	h=8
berth 1								
berth 2		ship 3				ship 4		
		4	5			3	3	3
berth 3	ship 1				ship 2			
	3	2	2		4	4	5	5
QCs	3	6	7	0	4	7	8	8

Figure 4.1: *TBAP equivalent solutions with the original cost matrix.*

the flow matrix, either configuration A or B can be dominated. To this purpose, we define the following quantities: $F_{12} = f_{11} + f_{12} + f_{21} + f_{22}$ and $F_{34} = f_{33} + f_{34} + f_{43} + f_{344}$. If $F_{12} > F_{34}$, then it is more convenient to assign vessels 1 and 2 to berth 1, and only configuration A is optimal. Otherwise, if $F_{12} < F_{34}$, configuration B dominates configuration A.

In order to maintain a fair comparison, all the results presented in the next sections, both for the MIP formulation solved by CPLEX and the heuristic algorithm, have been re-performed with respect to Chapter 3.

Results

Table 4.2 provides a comparison between the upper bound of the linear relaxation of the original MILP formulation (z_{LP}) and the upper bound obtained via Dantzig-Wolfe reformulation (z_{DW}), i.e., the optimal value of the master problem at the end of the root node, for instances with 10 vessels and 3 berths over a time horizon of one week. Computational times are not reported, as they are negligible for both cases. Column '% z_{DW}' reports the percentage of the bound improvement, that is always less than 0.5%, thus not very significant. However, the DW formulation proves to be much stronger than MILP when embedded into a branch-and-bound framework. Indeed, we notice that, after one hour of computation, the CPLEX bound is unchanged, despite of the depth of the search tree and the several branching decisions that have been made. On the contrary, it is often sufficient to perform a few branching decisions to see an improvement in the bound provided by the master problem.

The superiority of our approach is clearly shown in Table 4.3, that compares our branch-and-price algorithm to the general-purpose MIP solver on instances with 10 vessels; for completeness, we also report experimental results obtained with the heuristic algorithm. The branch-and-price algorithm is very efficient, as it always provides the optimal solution (opt_sol) relatively fast; column t(s) reports the computational time expressed in seconds. The best feasible solutions (best_sol) and the gap with respect to the optimal solution value are provided both for CPLEX and the heuristic algorithm. The time limit is set to one hour for the CPLEX MIP solver, while the heuristics stops according to the termination criteria explained in Chapter 3. The branch-and-price algorithm clearly outperforms CPLEX: it always provides the optimal solution in a few seconds, whereas the MIP solver often produces feasible solutions within a gap of 3%. In three cases, the MILP formulation cannot find a feasible solution in one hour. Furthermore, we remark that CPLEX fails to prove optimality for instances L1_p10 and L2_p10 because of the poor linear relaxation bound, that cannot be improved during the search in the branch-and-bound tree. Remarkably, for this class of instances, our branch-and-price shows computational times comparable (or even smaller) than the heuristic algorithm, while ensuring optimality of the solutions.

Tables 4.4 and 4.5 report the results for instances with 20 vessels and 5 berths over a time horizon of one week. We start analyzing the case of $\bar{p} = 10$, i.e., instances

10 x 3 Instance	z_{LP}	z_{DW}	% z_{DW}
H1_p10	800614	797594	0.38%
H1_p20	800890	797870	0.38%
H1_p30	800924	798136	0.35%
H2_p10	740947	738540	0.32%
H2_p20	741487	739190	0.31%
H2_p30	741523	739417	0.28%
L1_p10	519319	518334	0.19%
L1_p20	519354	518750	0.12%
L1_p30	519389	518785	0.12%
L2_p10	568152	566976	0.21%
L2_p20	568188	567012	0.21%
L2_p30	568224	567146	0.19%

Table 4.2: *Linear relaxation results for 10 ships and 3 berths over 1 week.*

10 x 3	**B&P**		**CPLEX (1h)**		**HEUR**		
Instance	**opt_sol**	**t(s)**	**best_sol**	**GAP**	**best_sol**	**GAP**	**t(s)**
H1_p10	790735	21	x	∞	786439	0.54%	7
H1_p20	791011	25	x	∞	785460	0.70%	21
H1_p30	791045	10	780722	1.30%	784658	0.81%	39
H2_p10	733276	2	712669	2.81%	732101	0.16%	8
H2_p20	735646	7	x	∞	729472	0.84%	20
H2_p30	735682	9	723818	1.61%	727443	1.12%	33
L1_p10	515902	7	515902	0.00%	513941	0.38%	7
L1_p20	518049	5	515991	0.40%	513847	0.81%	18
L1_p30	518084	27	513731	0.84%	509617	1.63%	37
L2_p10	564831	9	564831	0.00%	560915	0.69%	8
L2_p20	564867	7	561504	0.60%	559595	0.93%	18
L2_p30	564903	8	559389	0.98%	556998	1.40%	36

Table 4.3: *Branch-and-price results for 10 ships and 3 berths over 1 week.*

with 10 feasible quay crane assignment profiles per ship.

In Table 4.4 we can observe that the improvement in terms of linear relaxation bound has slightly increased (from 0.5% to about 1% on average) for this class of instances. However, the computational effort required by column generation is not

20 x 5 Instance	z_{LP}	z_{DW}	% z_{DW}	**t(s)**
H1_p10	1383614	1369818	1.00%	721
H2_p10	1474082	1459341	1.00%	504
L1_p10	1298356	1287080	0.87%	520
L2_p10	1103212	1094480	0.79%	640

Table 4.4: *Linear relaxation results for 20 ships and 5 berths over 1 week.*

20 x 5	**B&P (3h)**		**CPLEX (3h)**		**HEUR**		
Instance	**best_sol**	**GAP**	**best_sol**	**GAP**	**best_sol**	**GAP**	**t(s)**
H1_p10	x	∞	x	∞	1335625	2.50%	94
H2_p10	x	∞	x	∞	1428890	2.09%	84
L1_p10	1256529	2.37%	1221191	5.12%	1258098	2.25%	100
L2_p10	1059231	3.22%	x	∞	1070543	2.19%	89

Table 4.5: *Branch-and-price results for 20 ships and 5 berths over 1 week.*

negligible, as it takes about 10 minutes on average to close the root node. On the contrary, the MILP linear relaxation is solved in fractions of a second.

Table 4.5 compares our branch-and-price algorithm to CPLEX: in both cases, the time limit is set to three hours. Branch-and-price still performs better in terms of number of feasible solutions found (2 vs 1) and quality. Expectedly, the heuristic algorithm performs well: good quality solutions, with a gap of about 2% on average, are provided in less than 2 minutes.

We clearly notice that the increased size of the problem has a significant impact on the algorithm efficiency, and in three hours of computation no instance is solved at optimality.

Additional tests

In order to further investigate the sources of complexity of the problem, we design additional instances.

Firstly, we study an intermediate class of instances between the easy class (10 vessels, 3 berths) and the difficult class (20 vessels, 5 berths), by defining a new set of instances composed of 15 vessels and 3 berths, over a time horizon of one week. These instances are obtained by considering the first 15 vessels and the first 3 berths of the class 20 x 3. Computational results are reported in Tables 4.6 and 4.7. The improvement of the linear relaxation bound is comparable to the one obtained for class 10 x 3 (about 0.5%), and the computational time, although not negligible, is still reasonable (except for instance L2_p10). The branch-and-price algorithm always finds the optimal solution, opposite to CPLEX that is able to provide a feasible solution only for instance H2_p10 within 3 hours of computational time. However, the branch-and-price algorithm requires a significant effort in terms of computational time (about 2 orders of magnitude higher than for class 10 x 3). Surprisingly, the

15 x 3 Instance	z_{LP}	z_{DW}	% z_{DW}	**t(s)**
H1_p10	1189962	1183344	0.56%	11
H2_p10	1292357	1285075	0.56%	11
L1_p10	1110159	1105395	0.43%	7
L2_p10	899876	896483	0.38%	320

Table 4.6: *Linear relaxation results for 15 ships and 3 berths over 1 week.*

15 x 3	**B&P (3h)**		**CPLEX (3h)**		**HEUR**		
Instance	**opt_sol**	**t(s)**	**best_sol**	**GAP**	**best_sol**	**GAP**	**t(s)**
H1_p10	1170783	3507	x	∞	1163063	2.26%	34
H2_p10	1272247	3787	1250124	3.27%	1265782	2.06%	32
L1_p10	1098411	1203	x	∞	x	∞	27
L2_p10	890211	8975	x	∞	888112	1.31%	28

Table 4.7: *Branch-and-price results for 15 ships and 3 berths over 1 week.*

heuristic algorithm fails at finding a solution for instance L1_p10 with the default settings; however, by allowing a higher number of iterations, a near-optimal solution (0.06% gap with respect to the optimal solution) is found in about 20 minutes. For the remaining three instances, the heuristic performs well, providing relatively good quality solutions in less than one minute.

In the second set of instances we shorten the time horizon of class 20 x 5, from seven to four working days. In order to maintain feasibility, time windows are relaxed for every vessel; this new class is denoted by the suffix 4d in the instance name.

Computational results are reported in Tables 4.8 and 4.9. The linear relaxation bound is improved on average by 2%; remarkably, the MILP linear relaxation bound is unchanged, despite of the time horizon reduction and modified time windows. This gives an insight of how much "fractional" the MILP linear relaxation solution is. The root node is closed in about half the time needed previously for class 20 x 5 over one week. In Table 4.9 we report the best solutions found by the exact methods in three hours of computation: CPLEX is not able to provide any feasible solution, whereas the branch-and-price algorithm always provides solution with a gap between 3% and 5%. As previously, the heuristic performs well, although no feasible solution is provided for instance H2_p10_4d with the default settings; by allowing a higher number of iterations, a solution with a gap of 3.5% is found in about 10 minutes.

Discussion of results

Computational experiments show that our specialized branch-and-price algorithm outperforms the general-purpose solver in terms of quality of solutions and computational time.

The results confirm that designing sophisticated algorithms and exploiting the problem structure is crucial when tackling large-scale optimization problems as the

20 x 5 Instance	z_{LP}	z_{DW}	% z_{DW}	**t(s)**
H1_p10_4d	1383614	1356460	1.96%	315
H2_p10_4d	1474082	1444042	2.04%	136
L1_p10_4d	1298356	1274821	1.81%	483
L2_p10_4d	1103212	1084936	1.66%	373

Table 4.8: *Linear relaxation results for 20 ships and 5 berths over 4days.*

20 x 5	**B&P (3h)**		**CPLEX (3h)**		**HEUR**		
Instance	**best_sol**	**GAP**	**best_sol**	**GAP**	**best_sol**	**GAP**	**t(s)**
H1_p10_4d	1293184	4.66%	x	∞	1305216	3.78%	65
H2_p10_4d	1379208	4.49%	x	∞	x	∞	63
L1_p10_4d	1224458	3.95%	x	∞	1230409	3.48%	66
L2_p10_4d	1045778	3.61%	x	∞	1050171	3.20%	70

Table 4.9: *Branch-and-price results for 20 ships and 5 berths over 4days.*

TBAP. However, the instances' size still represents an issue and additional advanced techniques should be further investigated to overcome the complexity of the problem.

Most of the implemented accelerating techniques concern the pricing problem. This is motivated by the fact that preliminary results produced with a "basic" implementation of the branch-and-price algorithm (that included only heuristic pricing as accelerating technique) point out that about 99% of the computational time was spent in the pricing. The development of sophisticated and specialized techniques for the pricing problem is extremely successful: we reduced the computational time by 90% on average, as shown in Table 4.10.

We are currently working on a new method that solves the problem on a reduced-size space maintaining optimality. The framework is introduced and discussed in Chapter 6.

Furthermore, we think that future research should focus more on improving the master problem. Our guess is that linearization variables z_{ij}^{kw} significantly slow down the master problem as soon as the problem size increases; alternative linearizations should be investigated.

Finally, cutting planes for the problem should be studied, in order to improve the linear relaxation bound. A candidate class of valid inequalities are the so called *lifted cover inequalities* (Kaparis and Letchford, 2010) conceived for knapsack polytopes. Indeed, constraints (4.22) can be seen as knapsack constraints when variables s_{r_k} take binary values.

The addition of valid inequalities to the master problem generates new dual variables to be accounted in the pricing problem. Some attempts in this direction exist in literature for the multicommodity flow problem solved via column generation (Barnhart et al., 2000).

Preliminary investigation for the TBAP shows that the fractional solution of the

10 x 3	Basic B&P		Improved B&P		
Instance	**t(s)**	**% pricing**	**t(s)**	**% pricing**	**speed up(%)**
H1_p10	114	97%	21	10%	82%
H1_p20	995	97%	25	12%	97%
H1_p30	557	99%	10	18%	98%
H2_p10	12	82%	2	10%	83%
H2_p20	29	90%	7	11%	76%
H2_p30	25	92%	9	13%	65%
L1_p10	4054	99%	7	42%	100%
L1_p20	761	99%	5	62%	99%
L1_p30	470	99%	27	93%	94%
L2_p10	4697	99%	9	54%	100%
L2_p20	1573	99%	7	62%	100%
L2_p30	2680	99%	8	63%	100%

Table 4.10: *Reduction of computational time obtained with the accelerating techniques.*

root node provided by the column generation scheme actually violates some lifted cover inequalities. Therefore, our insight is that cuts would help in closing the gap faster, as up to now the linear relaxation bound improves slowly throughout the search tree.

4.2 Hierarchical vs integrated planning models

The branch-and-price algorithm introduced in the previous section enables us to perform a comparative analysis between hierarchical and integrated optimization approaches for the Tactical Berth Allocation Problem.

The aim of this study is to experimentally analyze the impact of integrated planning in terms of quality of the solutions and increased complexity. For this purpose, the Tactical Berth Allocation Problem represents the perfect case study, as it directly arises from the integration of two highly related decision problems, the berth allocation and the quay crane assignment, that are currently solved hierarchically in container terminals.

The hierarchical solution approach is composed of two models that are introduced in the next section. Computational results are presented and the potential benefits of integration are discussed.

4.2.1 The hierarchical BAP + QCAP approach

In the hierarchical approach, the berth allocation and the quay crane assignment are solved sequentially. The main assumption concerns the handling time of vessels, that is assumed to be known in advance. In practice, the expected handling time is provided by terminal planners, that base their estimations on quantitative data such as vessel's workload, average QC productivity, availability of transfer equipment, vessel's priority, as well as on their experience. In particular, some extra time can also be included in the estimation in order to guarantee more robustness and flexibility to the schedule.

The expected handling time of vessels is the necessary input of the whole hierarchical optimization process, that consists of the following two sequential steps:

1. **Berth Allocation Problem (BAP)**
   In this step, every vessel is assigned to a berth. For every berth, vessels are scheduled over the time horizon according to their time windows and expected handling times. The objective function aims to minimize the yard housekeeping costs generated by the resulting berth template.

2. **Quay Crane Assignment Problem (QCAP)**
   In this step, a quay crane profile (representing the number of quay cranes operating on the vessel during the working shifts associated with the allocated handling time) is assigned to every vessel. The main constraint is represented by the total quay crane capacity of the terminal, that is limited and must not be exceeded, especially when two or more vessels are serviced simultaneously. The berth allocation plan and scheduling determined at step 1 is therefore a necessary input for solving the QCAP. The objective function aims to maximize the monetary value associated with the quay crane profiles assigned to vessels.

The hierarchical approach is based on two separated models for the BAP and the QCAP. The objective functions of the two models are consistent with the global objective of the integrated TBAP model, in order to allow for comparison between the two approaches.

BAP model

The BAP model minimizes the housekeeping costs generated by the berth allocation and it requires as input the expected handling time t_i^{exp} for every vessel $i \in N$. The

formulation is the following:

$$\min \quad \frac{1}{2}\sum_{i\in N}\sum_{k\in M} y_i^k \sum_{j\in N}\sum_{w\in M} f_{ij} d_{kw} y_j^w \tag{4.58}$$

$$\text{s.t.} \quad \sum_{k\in M} y_i^k = 1 \qquad \forall i \in N, \tag{4.59}$$

$$\sum_{j\in N\cup\{d(k)\}} x^k_{o(k),j} = 1 \qquad \forall k \in M, \tag{4.60}$$

$$\sum_{i\in N\cup\{o(k)\}} x^k_{i,d(k)} = 1 \qquad \forall k \in M, \tag{4.61}$$

$$\sum_{j\in N\cup\{d(k)\}} x^k_{ij} - \sum_{j\in N\cup\{o(k)\}} x^k_{ji} = 0 \qquad \forall k \in M,\ \forall i \in N, \tag{4.62}$$

$$\sum_{j\in N\cup\{d(k)\}} x^k_{ij} = y^k_i \qquad \forall k \in M,\ \forall i \in N, \tag{4.63}$$

$$T^k_i + t^{exp}_i - T^k_j \le (1 - x^k_{ij})M1 \qquad \forall k \in M,\ \forall i \in N, \forall j \in N \cup \{d(k)\}, \tag{4.64}$$

$$T^k_{o(k)} - T^k_j \le (1 - x^k_{o(k),j})M2 \qquad \forall k \in M,\ \forall j \in N, \tag{4.65}$$

$$a_i y^k_i \le T^k_i \qquad \forall k \in M,\ \forall i \in N, \tag{4.66}$$

$$T^k_i \le b_i y^k_i \qquad \forall k \in M,\ \forall i \in N, \tag{4.67}$$

$$a^k \le T^k_{o(k)} \qquad \forall k \in M, \tag{4.68}$$

$$T^k_{d(k)} \le b^k \qquad \forall k \in M, \tag{4.69}$$

$$x^k_{ij} \in \{0,1\} \qquad \forall k \in M,\ \forall (i,j) \in A^k, \tag{4.70}$$

$$y^k_i \in \{0,1\} \qquad \forall k \in M,\ \forall i \in N, \tag{4.71}$$

$$T^k_i \ge 0 \qquad \forall k \in M,\ \forall i \in N \cup \{o(k), d(k)\}. \tag{4.72}$$

where M1 and M2 represent large positive constants.

We remark that decision variables λ, γ, ρ are not needed anymore, since the profile assignment and the quay crane capacity constraint are not taken into account at this stage. Furthermore, constraints (4.64) rely on the estimation of the handling time t^{exp}_i, that must be provided as input.

The BAP model can be linearized and reformulated via Dantzig-Wolfe, similarly to the procedure previously illustrated for the TBAP model. In particular, the resulting master problem differs from (4.20)-(4.28) for the QC capacity constraint (4.22), that is not taken into account in the BAP model. Furthermore, the network associated with the BAP pricing subproblem is reduced, having one node for every vessel $i \in N$ and for every arrival time step $h \in H$. In other words, the dimension associated with the quay crane profiles disappear. Similarly, the branch-and-price algorithm for TBAP can be easily adapted to solve the berth allocation problem.

QCAP model

The QCAP model relies on the output of the BAP model, especially on the arrival time T_i^{arr} of vessels $i \in N$. Furthermore, some preprocessing of data is needed, according to the berth allocation plan; in particular, only profiles of length $t_i^p \leq t_i^{exp}$ are feasible for the QCAP. We define the following additional notation:

$\widetilde{P}_i$ subset of profiles $p \in P_i, i \in N$ such that $t_i^p \leq t_i^{exp}$;

T_i^p berthing time associated with profile $p \in \widetilde{P}_i, i \in N$.

The parameter T_i^p is computed according to the starting time step of the profile within a shift (sets $P_i^s,\ s \in S$ defined in section 3.3.3), and to the arrival time T_i^{arr}.

The QCAP model can be formulated as follows:

$$\max \sum_{i \in N} \sum_{p \in \widetilde{P}_i} v_i^p \lambda_i^p \tag{4.73}$$

$$\text{s.t.} \sum_{p \in \widetilde{P}_i} \lambda_i^p = 1 \quad \forall i \in N, \tag{4.74}$$

$$\sum_{i \in N} \sum_{\substack{p \in \widetilde{P}_i \\ 0 \leq h - T_i^p < t_i^p}} \lambda_i^p q_i^{p(h - T_i^p + 1)} \leq Q^h \quad \forall h \in H, \tag{4.75}$$

$$\lambda_i^p \in \{0, 1\} \qquad \forall i \in N, \forall p \in \widetilde{P}_i. \tag{4.76}$$

As observed by Bierwirth and Meisel (2010) the quay crane assignment problem is relatively easy to solve compared to the berth allocation problem, that is NP-Hard (Lim, 1998).

4.2.2 Comparative analysis

The branch-and-price algorithm for TBAP presented in section 4.1.4 is further adapted to solve the BAP model in the hierarchical approach. The QCAP model is solved using a general-purpose MIP solver.

Handling time estimation

As mentioned, the hierarchical approach requires an estimation of the handling time, usually provided by the terminal's planners. Among the available TBAP data, we are given the set of feasible quay crane assignment profiles defined for every vessel and known in advance; in particular, we know the duration in terms of working shifts of every feasible QC profile.

In order to start the entire hierarchical optimization process, we produce two different estimations for the handling time, both motivated by the practice:

Scenario A for every vessel, the handling time is given by the longest feasible quay crane assignment profile;

Scenario B for mother vessels, the handling time is given by the shortest feasible quay crane assignment profile whereas for feeders, the handling time is given by the longest feasible quay crane assignment profile.

Scenario A is very conservative and somehow represents the worst-case scenario, when all vessels are serviced at the lowest rate. However, this handling time estimation may be useful to produce robust schedules.

Scenario B can be considered more realistic, since mother vessels typically have higher priority than feeders. In particular, we expect the terminal to operate as fast as possible mother vessels in order to minimize their stay at the port.

We remark that both scenarios are realistic and reasonable in practice.

Computational results

Computational experiments are performed on the set of perturbated instances introduced in section 4.1.5.

Table 4.11 compares the optimal solutions for the hierarchical approach under scenarios A and B to the integrated TBAP approach. We consider instances with 10 vessels and 3 berths over a time horizon of one week. For all solutions we report the value of the objective function ($\mathtt{opt_sol}$), the number of used berths (K) and the computational time in seconds ($\mathtt{t(s)}$). Columns '%(A)', '%(B)' indicate the improvement of the integrated solution with respect to the hierarchical approach under scenarios A, B respectively. We remark again that the global objective function refers to a maximization problem.

As expected, the integrated TBAP provides better solutions, although it seems that these specific instances do not allow for a significant gain in terms of objective function. The average improvement is 0.68% over scenario A and 0.36% over scenario B. Surprisingly, the computational effort required by the integrated approach is comparable to the hierarchical approach: optimal integrated solutions are therefore produced in a fast and efficient way.

These results seem to indicate that the tested instances are not very tight. Therefore, we perform additional tests by defining more congested instances. Such instances are obtained by reducing the time horizon from seven to four working days; in order to maintain feasibility, vessels' time windows are also relaxed and the quay crane capacity increases from $\mathsf{Q} = 8$ to $\mathsf{Q} = 10$.

Results for instances with a time horizon of four days are illustrated in Table 4.12. As soon as instances become more congested, the hierarchical approach clearly shows its drawbacks: first of all, the hierarchical approach is not always able to provide a feasible solution. More specifically, the quay crane assignment may not be feasible for a given berth allocation plan, due to the QC capacity constraint; this is often the case for scenario B, where the shortest handling time is assigned to mother vessels, and

10 x 3	Scenario A			Scenario B			Integrated TBAP				
Inst.	opt_sol	K	t(s)	opt_sol	K	t(s)	opt_sol	K	t(s)	%(A)	%(B)
H1_10	786841	2	14	789478	2	19	790735	2	21	0.49%	0.16%
H1_20	787689	2	14	789754	2	14	791011	2	25	0.42%	0.16%
H1_30	787723	2	9	789788	2	8	791045	2	10	0.42%	0.16%
H2_10	730702	2	3	733276	2	2	733276	2	2	0.35%	0.00%
H2_20	730418	2	6	732659	2	6	735646	2	7	0.72%	0.41%
H2_30	730454	2	4	732695	2	6	735682	2	9	0.72%	0.41%
L1_10	513661	2	5	515017	2	5	515902	2	7	0.44%	0.17%
L1_20	513696	2	5	515052	2	6	518049	2	5	0.85%	0.58%
L1_30	513731	2	7	515087	2	4	518084	2	27	0.85%	0.58%
L2_10	559683	2	7	561705	2	7	564831	2	9	0.92%	0.56%
L2_20	559719	2	4	561741	2	10	564867	2	7	0.92%	0.56%
L2_30	559755	2	4	561777	2	6	564903	2	8	0.92%	0.56%

Table 4.11: *Optimal solutions for 10 vessels and 3 berths over 1 week.*

10 x 3	Scenario A			Scenario B			Integrated TBAP				
Inst.	opt_sol	K	t(s)	opt_sol	K	t(s)	opt_sol	K	t(s)	%(A)	%(B)
H1_10_4d	x			x			777398	3	39	∞	∞
H1_20_4d	776331	3	10	x			779674	3	47	0.43%	∞
H1_30_4d	776365	3	13	x			782300	3	66	0.76%	∞
H2_10_4d	718900	3	8	x			722431	3	28	0.49%	∞
H2_20_4d	719987	3	11	x			724345	3	36	0.61%	∞
H2_30_4d	719701	3	13	x			725585	3	28	0.82%	∞
L1_10_4d	507422	3	5	508657	3	7	512533	2	4	1.01%	0.76%
L1_20_4d	507304	3	4	508505	3	6	512533	2	19	1.03%	0.79%
L1_30_4d	507339	3	4	508540	3	6	512991	2	10	1.11%	0.88%
L2_10_4d	553971	3	6	x			558750	2	16	0.86%	∞
L2_20_4d	554380	3	6	556272	3	4	558786	2	25	0.79%	0.45%
L2_30_4d	554380	3	6	556280	3	4	558822	2	6	0.80%	0.46%

Table 4.12: *Optimal solutions for 10 vessels and 3 berths over 4 days.*

therefore a higher number of cranes is used. In particular, the hierarchical approach fails in providing a solution for all the high-load instances (H1 and H2) under scenario B. On the contrary, the integrated approach finds the optimal solution for all tested instances in a reasonable time (always less than one minute).

The gain in terms of objective function is still modest (at most 1%); however, it is interesting to notice that the integrated solution makes use, in some cases, of one berth less than the solution provided by the hierarchical approach.

10 x 3	Housekeeping					Profiles				
Inst.	HK_A	HK_B	HK_{TB}	%(A)	%(B)	PV_A	PV_B	PV_{TB}	%(A)	%(B)
H1_10	118713	117138	115881	-2.4%	-1.1%	906616	905554	906616	0.0%	0.1%
H1_20	118713	117138	115881	-2.4%	-1.1%	906892	906402	906892	0.0%	0.1%
H1_30	118713	117138	115881	-2.4%	-1.1%	906926	906436	906926	0.0%	0.1%
H2_10	108806	106232	106232	-2.4%	0.0%	839508	839508	839508	0.0%	0.0%
H2_20	108806	107261	104402	-4.0%	-2.7%	840048	839224	839920	0.0%	0.1%
H2_30	108806	107261	104402	-4.0%	-2.7%	840084	839956	839956	0.0%	0.0%
L1_10	74805	73449	72564	-3.0%	-1.2%	588466	588466	588466	0.0%	0.0%
L1_20	74805	73449	70386	-5.9%	-4.2%	588435	588501	588501	0.0%	0.0%
L1_30	74805	73449	70386	-5.9%	-4.2%	588470	588536	588536	0.0%	0.0%
L2_10	84145	82123	78997	-6.1%	-3.8%	643828	643828	643828	0.0%	0.0%
L2_20	84145	82123	78997	-6.1%	-3.8%	643864	643864	643864	0.0%	0.0%
L2_30	84145	82123	78997	-6.1%	-3.8%	643900	643900	643900	0.0%	0.0%

Table 4.13: *Housekeeping and profiles' value for 10 vessels and 3 berths over 1 week.*

10 x 3	Housekeeping					Profiles				
Inst.	HK_A	HK_B	HK_{TB}	%(A)	%(B)	PV_A	PV_B	PV_{TB}	%(A)	%(B)
H1_10_4d	129717	127524	127218	-1.9%	-0.2%	x	x	906616		
H1_20_4d	129717	127524	127218	-1.9%	-0.2%	906054	x	906892	0.1%	
H1_30_4d	129717	127524	124626	-3.9%	-2.3%	906134	x	906926	0.1%	
L2_10_4d	119666	117077	117077	-2.2%	-0.0%	838566	x	839508	0.1%	
L2_20_4d	119666	117077	115702	-3.3%	-1.2%	839653	x	840048	0.0%	
L2_30_4d	119666	117077	114371	-4.4%	-2.3%	839367	x	839956	0.1%	
H2_10_4d	80511	79809	75933	-5.7%	-4.9%	587933	588466	588466	0.1%	0.0%
H2_20_4d	80820	79809	75444	-6.7%	-5.5%	588124	588314	588435	0.1%	0.0%
H2_30_4d	80820	79809	75444	-6.7%	-5.5%	588159	588349	588470	0.1%	0.0%
L1_10_4d	89137	87430	85078	-4.6%	-2.7%	643471	x	643828	0.1%	
L1_20_4d	89137	87430	85078	-4.6%	-2.7%	643598	643702	643864	0.0%	0.0%
L1_30_4d	89137	87430	85078	-4.6%	-2.7%	643598	643710	643900	0.0%	0.0%

Table 4.14: *Housekeeping and profiles' value for 10 vessels and 3 berths over 4days.*

We remark that the global objective function is composed of two terms, the value of profiles and the housekeeping costs. In Tables 4.13 and 4.14 we report the absolute values of housekeeping costs (HK) and profiles' total value (PV) for scenario A, scenario B and the TBAP integrated approach (denoted by subscript TB). We also provide the percentage improvement of the integrated approach on scenario A (%(A)) and scenario B (%(B)) for the two components of the objective function. In particular, TBAP reduces the housekeeping costs and increases the profiles' value.

We can observe that housekeeping costs are decreased by 4% on average with respect to scenario A and by 2% on average with respect to scenario B. On the contrary, profiles' total value is rarely improved. Unfortunately, the improvement obtained by the integrated approach is almost completely hidden when considering the global objective function, since for these specific instances the profiles value is about 8 times higher than housekeeping costs in the final objective function. Our insight is that, depending on data, the improvement would be much more emphasized.

To conclude, we think that the additional computational effort required by the integrated solution approach is worth it, especially for congested instances (that are those closer to reality). In particular, the integrated TBAP provides a more efficient use of terminal's resources.

4.3 Conclusions

In this chapter we have presented a branch-and-price algorithm for the Tactical Berth Allocation Problem. We have reformulated the problem via Dantzig-Wolfe decomposition and applied column generation. In order to obtain integer solutions, a branch-and-price scheme has been implemented and algorithmic details and accelerating techniques are discussed.

In particular, we presented advanced techniques specifically conceived for our problem, that proved to be very useful and that can be easily generalized for other branch-and-price schemes in various applications.

Computational tests prove that our exact algorithm outperforms commercial solvers: especially on small instances, branch-and-price always provides optimal solutions relatively fast.

However, the computational complexity of TBAP is not completely overcome and new exact solution approaches are worth being investigated to tackle difficult large-scale optimization problems, such as TBAP.

Finally, the developed branch-and-price algorithm enables us to provide an experimental comparison between the traditional hierarchical approach solving sequentially berth allocation and quay crane assignment, and our proposed integrated TBAP model. Computational tests confirm the added value of integration in terms of cost reduction and efficient use of resources.

Part II

From Methods to Applications

Part II is devoted to general models and algorithms for large-scale optimization problems, with a specific focus on methods.

In Part I we have seen that solving complex problems, such as the Tactical Berth Allocation Problem, is difficult even for sophisticated large scale optimization techniques such as standard column generation and branch-and-price.

In the second part of this dissertation we propose a new concept in the context of column generation and Dantzig-Wolfe decomposition able to handle such complex problems. We design a framework that is transferable across applications and that provides general results for large scale optimization problems.

The novel approach, called two-stage column generation, represents a major contribution of our work. The proposed methodology is validated on a new class of split delivery vehicle routing problems that generalizes specific characteristics of the Tactical Berth Allocation Problem and other applications in transportation, telecommunication and logistics.

This generalization results in the definition of the Discrete Split Delivery Vehicle Routing Problem with the following modeling features:

- discrete demand delivered in discrete orders: *in the TBAP, the demand is represented by the amount of quay cranes (discrete* items*) needed to complete the workload of a vessel; quay cranes are "delivered" as discrete combinations of items, called quay crane profiles (discrete* orders*);*

- service time dependent on the delivered quantity: *in the TBAP, a handling time is associated with every quay crane profile; according to the amount of cranes that are operating the vessel, the service has a different length (service time) that depends on the "delivered quantity" of cranes;*

- *we remark that the TBAP can be formulated as a multi-depot vehicle routing problem, according to Section 3.3.3.*

The mentioned modeling features are relevant also to other domains of application, such as telecommunication. In particular, we have found interesting similarities with the Field Technician Scheduling Problem (Xu and Chiu, 2001) and the Technician and Task Scheduling Problem (Cordeau et al., 2010).

Chapter 5

The Discrete Split Delivery VRPTW

5.1 Introduction

The capacitated Vehicle Routing Problem (VRP) consists of designing the optimal routes for a set of vehicles with given capacity in order to serve a set of customers. Customer's demand must be delivered by exactly one vehicle and vehicles' capacity cannot be violated. In the VRP with Time Windows (VRPTW) the service at any customer must start within a given time interval.

The Split Delivery Vehicle Routing Problem (SDVRP) is a relaxed version of the capacitated VRP in which the number of visits to customer locations is no longer constrained to be exactly one. In the SDVRP each customer can be visited by more than one vehicle which serves a fraction of its demand. It has been shown that this relaxation could yield to substantial savings on the total traveled distance, up to 50% in some instances, as well as on the number of required vehicles (Archetti, Savelsbergh and Speranza, 2006; Archetti, Savelsbergh and Speranza, 2008; Nowak et al., 2009). However, Gulczynski et al. (2009) have remarked that continuous splitting of the delivery may be not acceptable under a certain amount.

In the Discrete Split Delivery Vehicle Routing Problem (DSDVRP) the demand of a customer consists of several items that cannot be split further. The problem belongs to the class of split delivery problems since each customer's demand can be fractionated and each customer can be visited by more than one vehicle.

In this chapter we study the Discrete Split Delivery Vehicle Routing Problem with Time Windows (DSDVRPTW). We assume that demand can be split in predetermined discrete orders, i.e., feasible combinations of items (the unit request of a customer), that each vehicle can serve at most one order per customer and that service time at customer's location depends on the delivered combination of items. Remarkably, this is a modeling feature rarely found in literature, where service times are usually assumed to be independent of the delivered quantities. We refer e.g., to

Gendreau et al. (2006) and Desaulniers (2010), who make the simplifying assumption of constant service times: although it may be acceptable in applications where the unloading time is negligible, this is not an appropriate modeling assumption for applications where the unloading time is largely affected by the size of the delivery. Furthermore, discrete orders are particularly suited to specify any (possibly non-linear) relation among items, such as complicating constraints, incompatibilities and cost functions depending on orders.

Our main contribution is the definition of a new model for the Split Delivery VRPTW that takes into account discrete demands and delivery-dependent service times. Our model provides a more realistic representation of logistic problems, especially when deliveries cannot be treated as continuous quantities. Furthermore, delivery-dependent service times allow to understand to what extent delivery splitting can be advantageous in real cases: in our experiments we observe that delivery splitting into small fractions is rarely used in optimal solutions. The assumption of delivery-dependent service times affects some properties of optimal solutions with respect to known properties of Split Delivery VRPs that have been exploited in the literature to design and implement optimization algorithms. In particular, when column generation is used to solve the problem, as in our case, constant service times imply that no two split customers are shared among two vehicles' routes and therefore columns generated by the pricing problem have at most one split customer (Desaulniers, 2010). This is no more the case in our pricing subproblem. We study the properties of the DSDVRPTW and design a branch-and-price algorithm that exploits its specific structure. We extend state-of-the-art algorithms to cope with the additional complexity of the problem and manage to solve to optimality instances with up to 50 customers and a total of 350 orders. To provide a qualitative comparison, we recall that a DSDVRPTW instance is comparable, in terms of size of the associated network, to a VRPTW instance with as many customers as the total number of orders. To the best of our knowledge, instances of the VRPTW with up to 200 customers are solved to optimality. Our implementation largely outperforms a general-purpose commercial solver in terms of computational time and number of instances solved.

The increased difficulty of delivery-dependent service times is confirmed by computational experiments. Solving the same set of instances assuming constant service times, it results that those instances are much easier to solve. In particular, when service time is constant, several delivery splittings are easily dominated or unfeasible because of time windows limitations. Remarkably, considering delivery-dependent service time allows to produce some savings. In our experiments the solution of 14 instances is improved when delivery-dependent service times are considered.

The remainder of this chapter is organized as follows. In Section 5.2 we review scientific contributions relevant to the DSDVRPTW. In Section 5.3 we recall some known properties of split deliveries and show how they extend to the DSDVRPTW case. Section 5.4 provides an arc-flow formulation for the DSDVRPTW. In Section 5.5 we reformulate the problem using Dantzig-Wolfe decomposition and we illustrate

the column generation scheme. The branch-and-price implementation is presented in Section 5.6 and computational results are discussed in Section 5.7.

5.2 Literature review

The idea of obtaining some savings thanks to delivery splitting was introduced by Dror and Trudeau (1989), further developed by Dror and Trudeau (1990), who also provided some fundamental properties of optimal solutions of the SDVRP, exploited by other contributions to design effective optimization algorithms. Next, Dror et al. (1994) introduced a mathematical formulation based on integer programming, solved through a cutting plane approach able to provide bounds within 9% gap from optimum for instances with up to 20 customers. Lower bounds obtained by additional valid inequalities have been proposed by Belenguer et al. (2000); further exact methods (Gueguen, 1999; Jin et al., 2007) as well as heuristic algorithms (Archetti, Speranza and Hertz, 2006; Chen et al., 2007; Jin et al., 2008; Archetti, Speranza and Savelsbergh, 2008) have been proposed.

Gendreau et al. (2006) and Desaulniers (2010) address the problem with time windows (SDVRPTW) and present exact approaches based on column generation and branch-and-bound techniques. In particular, the method by Desaulniers (2010) exploits some known properties introduced by Dror and Trudeau (1989). In his column generation scheme, the author could limit the exploration of the pricing to columns visiting at most one split customer. The key idea of this paper is to allow so-called skip customers to be included in each route. The convex combination of such routes which is performed by the master problem can successfully lead to feasible integral solutions.

In all column generation schemes reviewed so far the decision of the amount to be delivered to each customer is left to the master problem. Ceselli et al. (2009b) propose lower bounds for the SDVRPTW based on a different formulation in which the amount to be delivered is left to the pricing. This bound is neither dominated nor dominates those of Gendreau et al. (2006) and Desaulniers (2010).

Integral solutions for instances with up to 100 customers have been provided by Ho and Haugland (2004) using tabu search. Finally, we refer the reader to the surveys by Archetti and Speranza (2008) and Chen et al. (2007).

We remark that in the papers reviewed so far, the following assumptions hold: firstly, the amount of demand to be delivered to each customer can be split into any fraction; secondly, for the problems with time windows, the service time to customers is assumed to be fixed and thus delivery independent. However, these assumptions may be unrealistic in real applications and this has motivated us to include in our model features such as discrete demand and orders, as well as delivery-dependent service time.

A first attempt to partially relax these assumptions is made by Gulczynski et al. (2009), who define a lower limit on the amount delivered to each customer motivated

by practical applications. Authors discussed the implications of such limit on the properties of optimal solutions of SDVRP. Beyond that limit the delivery can be split in any fractional manner.

Discrete demands have been introduced by Nakao and Nagamochi (2007), who present the discrete version of the SDVRP and propose a dynamic programming based heuristic able to solve real world instances with up to 77 customers. The algorithm is compared to other existing heuristics for the VRP. Indeed, many real world applications often present discrete split delivery characteristics: Ceselli et al. (2009a) present an exact approach to a real-world VRP in which customers' orders can be split among several vehicles in a discrete fashion. The authors propose a three level order aggregation that ends up, at the last level, in considering any possible combination of items.

Predetermined discrete orders and delivery-dependent service time are particularly suited to model real applications, such as the Field Technician Scheduling Problem (Xu and Chiu, 2001; Cordeau et al., 2010). A set of tasks of different types and requiring different skills must be assigned to technicians; every technician is able to deliver only the subset of tasks for which he is specialized. As each technician has different skills, the processing time of each task depends on the final task assignment. However, authors do not specifically relate their problems to the DSDVRP. Other contributions on real world routing with delivery splitting can be found in Belfiore et al. (2009) and Bolduc et al. (2010).

Finally, Giallombardo et al. (2010) describe a real-world problem in container terminal management where a non-linear monetary value is associated with combination of items. In this application, "orders" represent feasible assignments of quay cranes to vessels over time.

We relax the limiting assumption of continuous delivery splitting exploited in the literature and we model delivery dependent service times. We show that state-of-the-art techniques can be appropriately adapted to cope with this additional complexity and that the designed algorithm is able to compete in terms of solved instances with the best known algorithms.

5.3 Known properties of SDVRP extended to DSDVRPTW

In this section we recall some known properties of the VRP with Split Deliveries, firstly introduced by Dror and Trudeau (1990), extended to the variant with time windows by Gendreau et al. (2006). In particular, we discuss the implications of the new modeling feature introduced in this chapter, the delivery-dependent service time.

Property 1 *The SDVRP(TW) is a relaxation of the corresponding VRP(TW).*
The proof is given by Dror and Trudeau (1990). Let z_s^* be the value of the optimal

solution for the SDVRP(TW) and let z_f^* be the value of the optimal solution for the corresponding VRP(TW). Property 1 states that $z_s^* \leq z_f^*$. Clearly, $z_s^* \not> z_f^*$ for any problem instance since any VRP(TW) solution (and in particular, the optimal one) is a feasible solution for the corresponding SDVRP(TW). Furthermore, there exists instances such that $z_s^* < z_f^*$, as for the following example.

Dror and Trudeau's example We consider three demand points with $d_1 = 3$, $d_2 = 4$ and $d_3 = 3$; the distances between the points including the depot (node 0) are $c_{0i} = 2M$ for $i = 1, 2, 3$; $c_{12} = c_{23} = \epsilon$ and $c_{13} = 2\epsilon$. All vehicles have a capacity of five units. The VRP solution has a total cost of $12M$ and requires 3 vehicles, whereas the SDVRP solution has a total cost of $8M + 2\epsilon$ and requires only 2 vehicles (cf. Fig. 5.1).

Property 1.1 *The DSDVRP(TW) is a relaxation of the corresponding VRP(TW) when the unsplit order is defined for each customer.* We define the unsplit order as the order in which all items requested by a customer are delivered in a unique order. The proof easily derives from property 1 as any VRP(TW) solution is a feasible solution for the corresponding DSDVRP(TW).

Property 2 *Both SDVRP(TW) and DSDVRP(TW) are NP-Hard in the strong sense.* The NP-Hardness of SDVRPTW has been proven by Gendreau et al. (2006) who proposed a polynomial reduction from the Traveling Salesman Problem to the SDVRPTW. The same reduction applies for DSDVRPTW. An alternative proof can be obtained observing that the VRP(TW) is a special case of the DSDVRP(TW) when the unsplit order is defined for each customer. Therefore, DSDVRP(TW) is as hard as the VRP(TW), that is NP-Hard (Toth and Vigo, 2002).

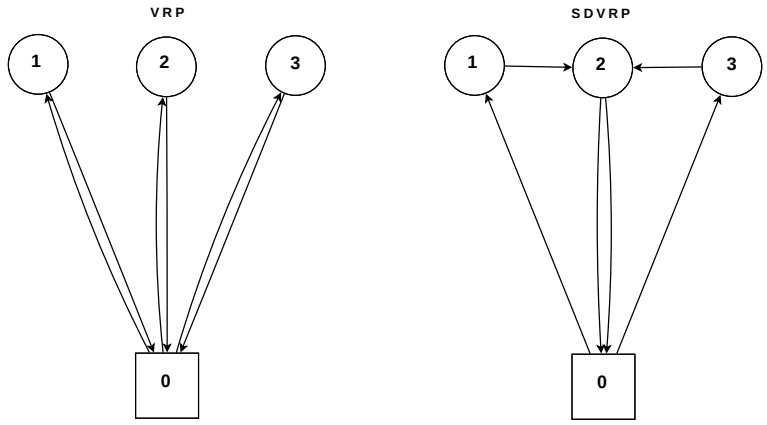

Figure 5.1: *Dror and Trudeau's example.*

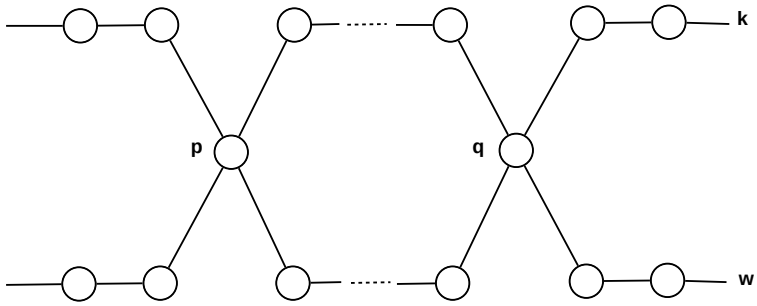

Figure 5.2: *Dror and Trudeau's two-route two-split example.*

Property 3 *When the cost matrix satisfies the triangular inequality, there exists an optimal solution of the SDVRP in which no two routes have more than one split demand in common.* The proof is given in Dror and Trudeau (1990). Consider an optimal SDVRP solution where two customers p and q are serviced by the same two routes k and w with split deliveries d_p^k, d_p^w, d_q^k and d_q^w (cf. Fig. 5.2). Without loss of generality, we assume that $d_p^k = \min\{d_p^k, d_p^w, d_q^k, d_q^w\}$. It is always possible to modify the quantities delivered by k and w to drop out customer p from route k, such that demands are still fulfilled, vehicles' capacity is not violated and the objective function does not increase its value. In particular, the new quantities are $d_p^{k\prime} = 0$, $d_p^{w\prime} = d_p^w + d_p^k$, $d_q^{k\prime} = d_q^k + d_p^k$ and $d_q^{w\prime} = d_q^w - d_p^k$.

Property 3 is extended to the SDVRP with Time Windows by Gendreau et al. (2006) under the assumption of constant service times. In particular, since route w still visits customers p and q, service times are unchanged and time windows are not affected. With respect to route k, since customer p is not visited anymore, the vehicle may reach some subsequent customer earlier that allowed; in this case, the vehicle will just wait at customer's location until it is allowed to start the delivery.

While properties 1 and 2 can be extended to the DSDVRPTW, property 3 does not apply because of the assumption of delivery-dependent service times. In this case, the increased quantity delivered by route w to customer p implies an increased service time at location p. As a consequence, the arrival and the delivery to the next customers may not comply with the time windows constraints anymore.

5.4 Arc-flow formulation

In this section we present a mixed integer linear program for the DSDVRPTW based on arc-flow formulation. Let $G(V, E)$ be a complete graph with $V = \{0\} \cup N$, where vertex $\{0\}$ represents the depot and $N = \{1, ..., n\}$ is the set of customers to be served. Each arc $(i, j) \in E$ has a cost c_{ij} and a travel time t_{ij}. The set of available vehicles with identical capacity Q is denoted by K. The set of items R is defined as $R = \bigcup_{i \in N} R_i$, where R_i represents the set of items to be delivered to customer $i \in N$.

Furthermore, $R_i \cap R_j = \emptyset \; \forall i \neq j, \;\; i,j \in N$, meaning that any item $r \in R$ is univocally associated with a customer $i \in N$. Each item $r \in R$ has a size q^r and a service time t^r. Items are delivered in orders, i.e., combinations of items. The set of orders C is given and defined as $C = \bigcup_{i \in N} C_i$, where C_i represents the set of feasible orders for customer $i \in N$. Furthermore, $C_i \cap C_j = \emptyset \; \forall i \neq j, \;\; i,j \in N$, meaning that any order $c \in C$ is univocally associated with a customer $i \in N$. Each order $c \in C$ has a size $q_c = \sum_{r \in R} e_c^r q^r$ and a service time t_c. Although t_c may incorporate any non-linear specification, for illustration purposes we assume that order's service time is equal to the sum of item's unloading times plus some constants administrative paper work time, i.e., $t_c = \sum_{r \in R} e_c^r t^r + \gamma$, where parameter e_c^r is used to identify the items composing the order c and equals 1 if item $r \in R$ is delivered in order $c \in C$ and 0 otherwise and γ is a constant service time related to the administrative paper work. Interval $[a_i, b_i]$ denotes the time window for customer $i \in N$. We define the following decision variables:

x_{ij}^k binary, equal to 1 if arc $(i,j) \in E$ is used by vehicle $k \in K$;
y_c^k binary, equal to 1 if vehicle $k \in K$ delivers order $c \in C$;
T_i^k ≥ 0, represents the arrival time of vehicle $k \in K$ at customer $i \in N$.

The discrete split delivery VRPTW can be formulated as follows:

$$z_{IP}^* = \min \sum_{k \in K} \sum_{(i,j) \in E} c_{ij} x_{ij}^k \tag{5.1}$$

$$\sum_{j \in V} x_{0j}^k = 1 \qquad \forall k \in K, \tag{5.2}$$

$$\sum_{j \in V} x_{ij}^k - \sum_{j \in V} x_{ji}^k = 0 \qquad \forall k \in K, \forall i \in V, \tag{5.3}$$

$$\sum_{j \in V} x_{ij}^k = \sum_{c \in C_i} y_c^k \qquad \forall k \in K, \forall i \in N, \tag{5.4}$$

$$\sum_{k \in K} \sum_{c \in C} e_c^r y_c^k = 1 \qquad \forall r \in R, \tag{5.5}$$

$$\sum_{c \in C_i} y_c^k \leq 1 \qquad \forall k \in K, \forall i \in N, \tag{5.6}$$

$$T_i^k + \sum_{c \in C_i} t_c y_c^k + t_{ij} - T_j^k \leq (1 - x_{ij}^k) M \qquad \forall k \in K, \forall i \in N, \forall j \in V, \tag{5.7}$$

$$T_i^k + (1 - x_{0i}^k) M \geq t_{0i} \qquad \forall k \in K, \forall i \in N, \tag{5.8}$$

$$T_i^k \geq a_i \sum_{j \in V} x_{ij}^k \qquad \forall k \in K, \forall i \in N, \tag{5.9}$$

$$T_i^k + \sum_{c \in C_i} t_c y_c^k \leq b_i \sum_{j \in V} x_{ij}^k \qquad \forall k \in K, \forall i \in N, \tag{5.10}$$

$$\sum_{c \in C} q_c y_c^k \leq Q \qquad \forall k \in K, \tag{5.11}$$

$$x_{ij}^k \in \{0,1\} \qquad \forall k \in K, \forall (i,j) \in E, \tag{5.12}$$

$$y_c^k \in \{0,1\} \qquad \forall k \in K, \forall c \in C, \tag{5.13}$$

$$T_i^k \geq 0 \qquad \forall k \in K, \forall i \in N. \tag{5.14}$$

where M is a sufficiently large positive constant. The objective function (5.1) minimizes the total traveling costs. Flow conservation is ensured by constraints (5.2)–(5.4), which also link x and y variables. Demand satisfaction is ensured by constraints (5.5): all items must be delivered (but not all combinations). Constraints (5.6) ensure that every vehicle delivers at most one order per customer. Precedence, time windows and capacity constraints are ensured by constraints (5.7)–(5.8), (5.9)–(5.10) and (5.11). Finally, the domain of variables is defined by (5.12), (5.13) and (5.14).

We remark that the service time at customer location depends on the selected order. This feature is modeled by the term $\sum_{c \in C_i} t_c y_c^k$ in constraints (5.7): it increases the complexity of the model, with respect to the same type of precedence constraints in classical VRP formulations with time windows.

5.5 Column generation

In this section we reformulate the DSDVRPTW model (5.1)–(5.14) via Dantzig-Wolfe decomposition and provide the formulations of the master problem and pricing subproblem. The master problem is solved by means of column generation.

5.5.1 Master problem

Let (5.2)-(5.4) and (5.6)-(5.14) be the constraints that define the feasible region of the subproblem and let $D^k = conv\{(x^k, y^k, T^k) \,|\, (x^k, y^k, T^k)$ satisfies $(5.2) - (5.4); (5.6) - (5.14)$ for $k\}$ be the feasible bounded domain of the subproblem associated with vehicle $k \in K$. Let P^k be the set of extreme points of D^k. Each extreme point $p^k = (x_p^k, y_p^k, T_p^k)$, $p^k \in P^k$ represents a feasible route for vehicle k with respect to vehicle's capacity and customers' time windows, delivering a unique order to every customer visited by the tour.

Since vehicles $k \in K$ present identical restrictions (in this case, the same capacity), all subproblems are identical and can therefore be aggregated into a single subproblem. We denote as $D = conv\{(x, y, T) \,|\, (x, y, T)$ satisfies $(5.2) - (5.4); (5.6) - (5.14)\}$ the feasible domain of the subproblem and P the set of extreme points of D. Each extreme point $p = (x_p, y_p, T_p)$, $p \in P$ represents now a feasible route that can be covered by any vehicle among the $|K|$ available.

The definition of the master problem requires the following additional notation: we denote c_p the cost of a route $p \in P$, defined as $c_p = \sum_{(i,j) \in p} c_{ij}$, while α_p^r denotes a binary parameter equal to 1 if route $p \in P$ delivers item $r \in R$. We also define

α_p^i as a binary parameter equal to 1 if route $p \in P$ visits customer i and used later in additional constraints for the master. After some standard adjustments and aggregation, the master problem can be formulated as follows:

$$\min \sum_{p \in P} c_p \lambda_p \tag{5.15}$$

$$\sum_{p \in P} \alpha_p^r \lambda_p = 1 \quad \forall r \in R \quad (\pi_r) \tag{5.16}$$

$$\sum_{p \in P} \lambda_p \leq |K| \qquad (\pi_0) \tag{5.17}$$

$$\lambda_p \geq 0 \quad \forall p \in P. \tag{5.18}$$

where λ_p are the decision variables associated with paths $p \in P$. The dual variables associated with constraints (5.16) are denoted as π_r while π_0 is the dual variable associated with constraint (5.17). The objective function (5.15) minimizes the total traveling cost. Constraints (5.16) ensure that all items are delivered to customers, while constraint (5.17) ensures that the number of chosen routes does not exceed the number of available vehicles.

We remark that constraints (5.16) need to be modeled as partitioning constraints in the DSDVRPTW, unlike common reformulations for routing problems that generally make use of covering constraints. This is due to the fact that, for every customer $i \in N$, the set of orders C_i does not necessarily contain all subsets of items $r \in R_i$, but only the subsets that are considered feasible with respect to the problem definition (incompatibilities between specific items, restrictions on the order size, etc.). As a consequence, a partitioning solution equivalent to the optimal covering solution may not exist.

5.5.2 Pricing subproblem

We denote by $\tilde{c}_p := c_p - \sum_{r \in R} \pi_r \alpha_p^r - \pi_0$ the reduced cost of a route $p \in P$. In a column generation scheme, given a dual solution of the (restricted) master problem, the pricing subproblem identifies the route p^* with the minimum reduced cost:

$$p^* = \arg\min_{p \in P}\{\tilde{c}_p\} = \arg\min_{p \in P}\{c_p - \sum_{r \in R} \pi_r \alpha_p^r - \pi_0\}. \tag{5.19}$$

The subproblem formulation relies on variables x, y and T defined in Section 5.4 (without index k, since we have aggregated the subproblems) and can be written as follows:

$$\min \sum_{(i,j) \in E} c_{ij} x_{ij} - \sum_{r \in R} \pi_r (\sum_{c \in C} y_c e_c^r) - \pi_0 \tag{5.20}$$

$$\sum_{j \in V} x_{0j} = 1 \tag{5.21}$$

$$\sum_{j \in V} x_{ij} - \sum_{j \in V} x_{ji} = 0 \qquad \forall i \in V, \tag{5.22}$$

$$\sum_{j \in V} x_{ij} = \sum_{c \in C_i} y_c \qquad \forall i \in N, \tag{5.23}$$

$$\sum_{c \in C_i} y_c \leq 1 \qquad \forall i \in N, \tag{5.24}$$

$$T_i + \sum_{c \in C_i} t_c y_c + t_{ij} - T_j \leq (1 - x_{ij})M \qquad \forall i \in N, \forall j \in V, \tag{5.25}$$

$$T_i + (1 - x_{0i})M \geq t_{0i} \qquad \forall i \in N, \tag{5.26}$$

$$T_i \geq a_i \sum_{j \in V} x_{ij} \qquad \forall i \in N, \tag{5.27}$$

$$T_i + \sum_{c \in C_i} t_c y_c \leq b_i \sum_{j \in V} x_{ij} \qquad \forall i \in N, \tag{5.28}$$

$$\sum_{c \in C} q_c y_c \leq Q \tag{5.29}$$

$$x_{ij} \in \{0, 1\} \qquad \forall (i, j) \in E, \tag{5.30}$$

$$y_c \in \{0, 1\} \qquad \forall c \in C, \tag{5.31}$$

$$T_i \geq 0 \qquad \forall i \in N. \tag{5.32}$$

Analyzing the objective function, we can observe that two major decisions are made in the subproblem:

a) the sequence of visited customers $i \in N$ (cost component c_{ij});

b) for each customer in the route, the order $c \in C$ to be delivered, and therefore the subset of items $r \in R$ delivered by the route (cost component e_c^r).

The pricing problem (5.20)–(5.32) can be cast to an Elementary Shortest Path Problem with Resource Constraints (ESPPRC) on a network that has one node for every order $c \in C$ and whose arcs have transit time equals to $(t_{ij} + t_c)$.

5.6 Branch-and-price implementation

For solving the DSDVRPTW we implement a branch-and-price algorithm adapting state-of-the-art solution techniques for the pricing and the master problem.

The pricing problem is solved using bounded bi-directional dynamic programming (Righini and Salani, 2006) with decremental state space relaxation (Righini and Salani, 2008). The algorithm is initialized by a preprocessing phase, used to

identify and remove trivially dominated combinations, and by a simple greedy algorithm used to find a feasible solution to the problem. Such solution allows to compute an upper bound on the cost of the solution and on the number of vehicles. The search tree is explored using a best-first strategy.

5.6.1 Branching scheme

In the search tree, branching is required when the master problem is solved at optimality and the corresponding solution of the arc-flow formulation is not integer. We have implemented a branching scheme consisting of four hierarchical levels of increasing complexity for what concerns additional constraints in the master problem or additional complexity of the pricing problem:

1. if the total number of vehicles is fractional ($\sum_{p\in P}\lambda_p = \tilde{K}$), branching is performed on constraint (5.17) by enforcing $\sum_{p\in P}\lambda_p \leq \lfloor\tilde{K}\rfloor$ on the first child node and $\sum_{p\in P}\lambda_p \geq \lceil\tilde{K}\rceil$ on the second child node.

2. if the number of vehicles visiting some customer $i \in N$ is fractional ($\sum_{p\in P}\alpha_p^i\lambda_p = \tilde{K}_i$), branching requires additional constraints to be added to the master problem for customer i: $\sum_{p\in P}\alpha_p^i\lambda_p \leq \lfloor\tilde{K}_i\rfloor$ on the first child node and $\sum_{p\in P}\alpha_p^i\lambda_p \geq \lceil\tilde{K}_i\rceil$ on the second child node. This branching requires the dual values associated with the additional constraints to be collected and accounted in the pricing.

3. if there is an arc $(i,j) \in E$ visited a fractional number of times ($\sum_{k\in K}x_{ij}^k = \tilde{x}_{ij}$), branching requires additional constraints in the master problem: $x_{ij} \leq \lfloor\tilde{x}_{ij}\rfloor$ on the first child node and $x_{ij} \geq \lceil\tilde{x}_{ij}\rceil$ on the second child node. This branching requires the dual values associated with the additional constraints to be collected and accounted in the pricing; however, the pricing structure is not affected.

4. if none of the above conditions holds, then there exist two consecutive arcs $(i,j) \in E$ and $(j,l) \in E$ visited consecutively a fractional number of times: $\sum_{p\in P_{ijl}}\lambda_p = \tilde{z}_{ijl}$, where P_{ijl} denotes the set of paths containing arc (j,l) immediately after arc (i,j). In this case, branching requires additional constraints to be added to the master: $z_{ijl} \leq \lfloor\tilde{z}_{ijl}\rfloor$ on the first child node and $z_{ijl} \geq \lceil\tilde{z}_{ijl}\rceil$ on the second child node. This branching requires the dual values associated with the additional constraints to be collected and accounted in the pricing, it also requires modifying the pricing as one additional resource must be added to the state of the dynamic programming algorithm for each additional constraint added to the master problem. However, this last branching is rarely needed (<1% of instances in our tests). See Ryan and Foster (1981) and Desrochers and Soumis (1989) for additional details.

5.6.2 2-Path Cuts

At the root node we try to identify valid 2-path inequalities that are violated by the current linear relaxation solution.

The basic idea of k-path inequalities (Kohl et al., 1999) is to identify a subset of customers that is visited by less than k vehicles in the current fractional solution, although it requires, in the optimal solution, at least k vehicles to be serviced. For any subset of customers $S \subseteq N$, $|S| \geq 1$ we define the flow into S, denoted $x(S)$, as $x(S) = \sum_{i \in \bar{S}} \sum_{j \in S} x_{ij}$ where $\bar{S} = N \setminus S$. Given the smallest number of vehicles needed to service all the customers in S, denoted by $k(S)$, a valid k-path inequality is defined by $x(S) \geq k(S)$.

Since calculating $k(S)$ is very time consuming, we have limited the search to the 2-path inequalities. This reduces to identify some set S such that $x(S) < 2$ and $k(S) > 1$. To determine whether $k(S) > 1$, we solve a Traveling Salesman Problem with Time Windows (TSPTW) for S: if a TSPTW solution cannot be found, then $k(S) > 1$. Since the number of sets S grows exponentially, in our search we limited the size of S to twice the average number of customers per vehicle. Sets are chosen heuristically in a first phase, then, if no valid inequality is found, sets are enumerated. All 2-path inequalities that are violated by more than a predetermined threshold value (0.2 for our tests) are added to the master problem, defining a new linear relaxation to solve.

5.7 Computational results

Algorithms are coded in ANSI C and compiled with gcc 4.1.2. Computational experience is run under a linux operating system on a 2Ghz Intel processor equipped with 2GB of RAM. All restricted master problems are solved using CPLEX version 10.2.

5.7.1 Instances

To the best of our knowledge there is no standard dataset used in the literature for the DSDVRPTW. The most related contribution is that of Nakao and Nagamochi (2007) for which the instances are not available.

We generate our test bed from the well-known Solomon's data set (Solomon, 1983). For all instances of classes R1, C1 and RC1 we consider the first $n = 25, 50$ customers and we discretize the demand of each customer in 12 items ($|R_i| = 12 \ \forall i \in N$). For each customer, we generate 7 orders to represent a meaningful set of possible delivery splittings: 1 full order (containing 12 items); 2 complementary orders 50%-50% (containing 6 items each, partitioned); 2 complementary orders 75%-25% (containing 9 and 3 items respectively, partitioned); 2 complementary 90%-10% orders (containing 11 and 1 items respectively, partitioned). We remark that order complementarity is necessary as we are searching for possible item partitions, thus any order without its complement can be eliminated in a preprocessing phase as it would never be part of a

feasible solution. To build instances of increasing complexity, we consider 3 possible scenarios:

A: full order + 50-50% orders ($|C_i| = 3$);
B: full order + 50-50% orders + 75-25% orders ($|C_i| = 5$);
C: full order + 50-50% orders + 75-25% orders + 90-10% orders ($|C_i| = 7$).

The full order is always included in order to allow the comparison of the DSDVRPTW with the classical VRPTW. The unsplittable case, which is trivially composed of the full order only ($|C_i| = 1$), is denoted as scenario O; detailed results including scenario O are provided in the appendix. We limit the presentation of the results to $|C_i| = 7$ because it is the limit for which our current implementation is able to provide some optimal solutions. In order to enhance splitting, we consider more restrictive capacities than Solomon's, as already done by Gendreau et al. (2006) and Desaulniers (2010). Instances are tested with $Q = 30, 50$ and 100.

From the 29 original Solomon's instances (12 for class R1, 9 for class C1 and 8 for class RC1), we derive 174 instances: 29×2 (25 and 50 customers) $\times$ 3 (30, 50 and 100 for vehicle's capacity). Each instance is tested under the 4 scenarios A, B, C and O.

5.7.2 Branch-and-price results for the DSDVRPTW

The flow-based formulation (5.1)–(5.14) is implemented and solved by Gurobi 3.0. In two hours of computation, it is able to solve to optimality 3 out of 87 instances with 25 customers, scenario A, whereas our algorithm can solve 66 of them. For the 3 instances solved by Gurobi, our algorithm is faster by four orders of magnitude. For the remaining instances, Gurobi solves 34 instances with an average gap greater than 20% and for the 40 instances no gap is provided in two hours of computation because the solver is not able to produce an integral solution. These results motivate the design of a more targeted solution approach. The efficiency of our branch-and-price algorithm is confirmed by the following results.

Table 5.1 presents a summary of the instances solved by the branch-and-price within 1 hour of computational time. Instances are grouped by the number of customers (n) and the capacity (Q). The number of instances of each class is also provided (*nb_inst*). For each group, the table provides the number of instances solved at optimality (*nb_solved*) and the average computational time in seconds (t) for each DSDVRPTW scenario.

We are able to solve 88, 67 and 47 out of 174 instances for scenarios A, B and C, respectively. The difficulty of solving the instances increases with the size of $|C|$: 75, 125 and 175 orders with 25 customers and 150, 250, and 350 orders with 50 customers for scenarios A, B and C, respectively. This difficulty also increases with the number of customers: we were able to solve 76% (A), 60% (B) and 48% (C) of instances with $n = 25$, whereas only 25% (A), 17% (B) and 6% (C) of instances with $n = 50$ were solved at optimality. The average computational time is also affected by the size of

				scenario A		**scenario B**		**scenario C**	
n	class	nb_inst	Q	nb_solved	t	nb_solved	t	nb_solved	t
25	R1	12	30	12	7	12	75	8	466
			50	12	6	12	60	12	430
			100	12	9	12	41	12	113
25	C1	9	30	4	1108	0	x	0	x
			50	9	37	4	2137	0	x
			100	7	706	4	705	2	1876
25	RC1	8	30	2	1988	0	x	0	x
			50	0	x	0	x	0	x
			100	8	3	8	11	8	35
50	R1	12	30	1	1010	0	x	0	x
			50	3	1572	1	385	0	x
			100	3	1035	2	167	2	535
50	RC1	8	30	0	x	0	x	0	x
			50	7	54	6	902	0	x
			100	8	529	6	809	3	2832

Table 5.1: *Summary of the results on delivery-dependent service time instances.*

$|C|$ and the number of customers.

For $n = 25$ customers, instances of class R1 are the easiest to solve. There are 36 (A), 36 (B) and 32 (C) solved instances out of 36 for class R1; 20 (A), 8 (B) and 2 (C) solved instances out of 27 for class C1; 10 (A), 8 (B) and 8 (C) solved instances out of 24 for class RC1. On average, 96% of instances are solved in class R1, 37% in class C1 and 36% in class RC1.

For $n = 50$ customers, class RC1 seems easier to solve than class R1 (on average, 42% versus 11% of solved instances), while no instance in class C1 is solved.

Surprisingly, for class RC, we solve more instances with 50 customers and capacity 50 than with 25 customers and the same capacity. This is due to better upper bounds found earlier in the search tree in the 50 customer case which helped to prune more rapidly the tree. The same feasible vehicles' routes are not present for the 25 customers case.

Detailed results for all instances solved are provided in Appendix A. From these results, we observe that split deliveries are more frequent for instances with small Q values, although they also occur for certain instances with $Q = 100$. In some cases, split deliveries not only decrease the total traveling costs but also allow to save one vehicle. This is relevant for real world logistic problems: indeed, being able to serve the same demand with a reduced size of the fleet implies important savings in terms of fixed investments.

We can also observe that advantageous split deliveries occur with instances of scenarios A and B. The optimal solutions of these scenarios are rarely improved by solutions of scenario C (we recall that the set of feasible solutions of scenario A is a

subset of feasible solutions of scenarios B and C and the same holds for solutions of scenario B w.r.t scenario C). Thus, the improvement that can be obtained by delivery splitting in real logistic problems is to some extent limited and extreme fractional splitting is never convenient. Moreover, discrete splitting with small quantities makes the problem too hard to solve as already noticed by Ceselli et al. (2009a). We conclude that the splitting options considered by scenario B are a good compromise between solution's quality and instance's complexity.

5.7.3 Delivery-dependent service times vs. constant service times

In this section we show that the more realistic assumption of delivery-dependent service times leads to a more difficult problem. In particular, a comparison with the assumption of constant service times is provided.

We test our branch-and-price algorithm on the same set of instances described in section 5.7.1, but assuming constant service time, i.e, the same fixed service time for every order delivered to a customer, disregarding quantity. The service time for each customer equals the nominal service time defined in the original Solomon's instances and every partial order has the same service time of the full order. This is also the assumption made by Gendreau et al. (2006) and Desaulniers (2010).

A summary of the computational results is reported in Table 5.2, where a comparison between delivery-dependent service time ($DDST$) and constant service time (CST) is provided in terms of number of instances solved (nb) and average computational time (t). The figures highlighted in bold denote an increased number of solved instances for CST. Since many more instances are solved under the assumption of constant service times, we also report the average computational time with respect to the subset of instances solved under the assumption of delivery-dependent service time ($\mathtt{t}_{\mathtt{fair}}$), in order to allow a fair comparison in terms of computational time.

We observe that instances with constant service time are easier to solve: 247 out of 522 instances (47%) solved to optimality for CST against 202 out of 522 (38%) for DDST. Furthermore, the algorithm is faster for CST, up to one order of magnitude for several classes of instances.

However, the trade-off is clear: considering constant service times makes the problem easier to solve but may also result in a loss of efficiency of the solution, as shown in the detailed results provided in Appendix A. From these results, we observe that considering delivery-dependent service time may lead to savings in terms of total costs, especially when the number of customers and the vehicle's capacity increase. We infer that savings mainly occur when many customers can be served by the same vehicle with respect to capacity but time windows constraints become more stringent. In this case, splitting under the delivery-dependent service time assumption can be convenient, since the decreased service time can reduce the cost of the solution.

On tested instances, improvements in terms of costs are limited within 1%. Further

				scenario A					**scenario B**					**scenario C**				
				DDST		**CST**			**DDST**		**CST**			**DDST**		**CST**		
n	class	nb_inst	Q	nb	t	nb	t	t_{fair}	nb	t	nb	t	t_{fair}	nb	t	nb	t	t_{fair}
25	R1	12	30	12	7	12	2	*2*	12	75	12	18	*18*	8	466	**11**	570	*87*
			50	12	6	12	1	*1*	12	60	12	11	*11*	12	430	12	34	*34*
			100	12	9	12	3	*3*	12	41	12	12	*12*	12	113	12	35	*35*
25	C1	9	30	4	1108	**5**	521	*164*	0	x	0	x	*x*	0	x	0	x	*x*
			50	9	37	9	12	*12*	4	2137	**6**	765	*223*	0	x	0	x	*x*
			100	7	706	**8**	135	*48*	4	705	**7**	294	*25*	2	1876	**5**	497	*120*
25	RC1	8	30	2	1988	**4**	1175	*507*	0	x	0	x	*x*	0	x	0	x	*x*
			50	0	x	0	x	*x*	0	x	0	x	*x*	0	x	0	x	*x*
			100	8	3	8	<1	*<1*	8	11	8	3	*3*	8	35	8	15	*15*
50	R1	12	30	1	1010	**4**	698	*65*	0	x	0	x	*x*	0	x	0	x	*x*
			50	3	1572	**6**	813	*148*	1	385	**6**	1134	*39*	0	x	**1**	65	*x*
			100	3	1035	**6**	1334	*41*	2	167	**3**	189	*11*	2	535	**3**	695	*29*
50	RC1	8	30	0	x	0	x	*x*	0	x	0	x	*x*	0	x	0	x	*x*
			50	7	54	7	12	*12*	6	902	**8**	95	*92*	0	x	**6**	1754	*x*
			100	8	529	8	43	*43*	6	809	**8**	231	*148*	3	2832	**6**	787	*745*

Table 5.2: *Summary of the comparison of delivery-dependent service time (DDST) vs constant service time (CST) instances.*

research can be done to asses potential savings, especially on instances with a large number of customers.

5.8 Conclusions

In this chapter we have modeled a Split Delivery VRPTW with new additional features such as discrete splits and delivery-dependent service times. A branch-and-price algorithm has been implemented by adapting state-of-the-art techniques to the specific structure and properties of the problem. Computational results have shown that our algorithm is efficient and it largely outperforms commercial solvers.

The new additional modeling features, i.e., discrete splits and delivery-dependent service times, allow to model with much more realism real logistic problems in which continuous demand splitting is not acceptable and the processing time depends on delivered quantities. Our realistic assumptions lead to additional complexity to solve the problem to optimality, as confirmed by the experimental comparison with instances assuming constant service time. However, an efficient adaptation of state of the art techniques allowed us to solve real sized instances in a reasonable amount of time.

Analyzing the computational results, we can conclude that the problem is complex; nevertheless, we managed to solve instances with up to 50 customers and 7 orders per customer, i.e., a total of 350 orders. Although more efficient solution techniques could be explored, we consider these results satisfactory and a good starting point for investigating more sophisticated approaches in the future.

Chapter 6

Two-stage column generation

6.1 Introduction

Column generation is nowadays the most successful tool to solve large-scale integer optimization problems arising in real-world applications. Branch-and-price codes are able to solve problems that commercial MIP solvers could never cope with. However, practical problems of growing size and complexity represent a challenge for the research community and the need of further advances in column generation, both theoretically and algorithmically, is well recognized (Lübbecke and Desrosiers, 2005; Lübbecke, 2010).

In the last decade, different research directions have been explored, aiming to design accelerating techniques for the master and the pricing problem, and to cope with instability issues that affect column generation.

Stabilized column generation, introduced by du Merle et al. (1999) and Ben Amor (2002), has been devised to overcome drawbacks such as slow convergence and generation of irrelevant columns in the first iterations (Vanderbeck, 2005). The main reason is due to the unstable behavior of dual variables, that do not follow a specific pattern throughout the iterations. Stabilization aims therefore to reduce these effects by controlling dual variables. A review of different stabilization techniques for column generation and numerical comparison on five applications can be found in (Briant et al., 2008).

Dynamic constraint aggregation is a method proposed by Elhallaoui et al. (2005; 2008; 2010) to reduce the master problem size by aggregating set partitioning constraints. The main objective is to speed up the solution of the master problem, often slowed down by high primal degeneracy. Optimality is guaranteed by dynamically adjusting the set of aggregated constraints. The crucial point of the method is that aggregated dual master variables need to be disaggregated; remarkably, the proposed disaggregation strategy is specifically designed to satisfy a large number of dual constraints and therefore to speed up convergence. The methodology has been recently extended to cope with general degenerate linear programs and is referred to

as Improved Primal Simplex (Raymond et al., 2010).

Irnich et al. (2010) have proposed an exact method for *arc variable elimination* based on path reduced costs, in the context of column generation with shortest path pricing subproblems. The technique is based on a well known property in integer programming: if the reduced cost of a non-negative integer variable exceeds a given optimality gap, the variable must be zero in any optimal integer solution (Nemhauser and Wolsey, 1988). The authors investigate the relationship between reduced costs of the compact and the extensive formulation, and extend the method proposed by Walker (1969) to compute reduced costs of original formulation variables starting from a dual feasible solution to the master problem. The technique allows to significantly reduce the size of the expanded network underlying the pricing, while keeping optimality.

All the mentioned techniques confirm that good dual information is crucial to enhance column generation schemes. Furthermore, the relationship between compact formulation and column generation is worth investigating, as it may represent an important source of information; in particular, branching rules based on the compact formulation variables have been studied by Villeneuve et al. (2005).

In this chapter we propose a novel framework for complex large-scale optimization problems called *two-stage column generation* where we simultaneously generate columns both for the compact and the extensive formulation. The approach is particularly suited for problems where the large number of variables in the compact formulation directly affects the pricing problem and its efficiency; specifically, we find this structure in the DSDVRPTW, where the expanded network of the pricing subproblem depends not only on arcs and customers, but also on discrete orders. As mentioned, this structure is common to different real-world applications, such as the Tactical Berth Allocation Problem and the Field Technician Scheduling Problem.

The basic idea of two-stage column generation is simple: we propose to start solving the problem on a subset of compact formulation variables, we apply Dantzig-Wolfe decomposition and we solve the resulting master problem via column generation. At this point, either profitable compact formulation variables are dynamically generated and added to the formulation; or we prove that the current solution is optimal, in the same spirit of standard column generation. The key point of our approach is that we evaluate the contribution of compact formulation variables with respect to the extensive formulation, in order to take advantage of the constraints that have been "convexified" in the DW reformulation: indeed, we aim at adding compact formulation variables that are profitable for the master problem, regardless of the optimal solution of the linear relaxation of the compact formulation. At the end of this procedure we possibly consider a smaller subset of original variables and, furthermore, suboptimal variables are detected and eliminated.

We remark that our new framework differs from *nested column generation*, where basically both the master and the pricing problem are solved via column generation and dual information is available for both problems. This approach was proposed by Vanderbeck (2001) for the 3stage 2dimensional cutting stock problem, and recently

applied to the crude oil tanker routing and scheduling in the context of maritime transport (Hennig, 2010).

6.2 Standard column generation

In this section we recall the terminology for standard column generation. Consider the following integer linear program, the *original* or *compact formulation* (CF):

$$
\begin{aligned}
z_{IP} = \min \quad & c^T x && (6.1)\\
\text{s.t.} \quad & Ax \geq b, && (6.2)\\
& Dx \geq d, && (6.3)\\
& x \in \mathbb{Z}^n_+. && (6.4)
\end{aligned}
$$

We assume that constraints $\{Dx \geq d\}$ present a particular structure that can be "convexified". Let $P = conv\{x \in \mathbb{Z}^n_+ : Dx \geq d\} \neq \emptyset$ be a bounded polyhedron. We can represent each $x \in P$ as a convex combination of extreme points $\{p_q\}_{\{q \in Q\}}$ of P:

$$
x = \sum_{q \in Q} p_q \lambda_q, \quad \sum_{q \in Q} \lambda_q = 1, \quad \lambda \in \mathbb{R}^{|Q|}_+. \tag{6.5}
$$

The equivalent *extensive formulation* (EF) of (6.1)–(6.4) is:

$$
\begin{aligned}
z_{IP} = \min \quad & \sum_{q \in Q} c_q \lambda_q && (6.6)\\
\text{s.t.} \quad & \sum_{q \in Q} A_q \lambda_q \geq b, && (6.7)\\
& \sum_{q \in Q} \lambda_q = 1, && (6.8)\\
& \lambda \geq 0, && (6.9)\\
& x = \sum_{q \in Q} p_q \lambda_q, && (6.10)\\
& x \in \mathbb{Z}^n_+. && (6.11)
\end{aligned}
$$

where $c_q = c^T p_q$ and $A_q = A p_q \ \forall q \in Q$. Constraints (6.10) are usually referred to as *coupling constraints*. If we relax the integrality of x in (6.11), coupling constraints

also become redundant and the resulting *master problem* (MP) is:

$$z_{MP} = \min \quad \sum_{q \in Q} c_q \lambda_q \tag{6.12}$$

$$\text{s.t.} \quad \sum_{q \in Q} A_q \lambda_q \geq b, \tag{6.13}$$

$$\sum_{q \in Q} \lambda_q = 1, \tag{6.14}$$

$$\lambda \geq 0. \tag{6.15}$$

Typically, λ variables are in a large number and cannot be enumerated explicitly. Column generation allows for implicit enumeration, and profitable variables are dynamically added to the master problem.

Specifically, we repeatedly solve a *restricted master problem* (RMP) on a subset of variables λ and, at each iteration, we add negative reduce-cost variables not yet in the formulation, if any, by solving the *pricing subproblem*:

$$\min_{q \in Q} \tilde{c}_q := \min_{q \in Q} c_q - \pi A_q - \pi_0 \tag{6.16}$$

where $\pi \geq 0$ is the dual vector associated with constraints (6.13), $\pi_0 \in \mathbb{R}$ is the dual variable associated with the convexity constraint (6.14) and $\tilde{c}_q$ denotes the reduced cost of variable λ_q.

6.3 Two-stage column generation

In this section we formally describe our new framework and we introduce some specific notation and terminology. Furthermore, methods to evaluate the contribution of compact formulation variables to the master problem are discussed.

6.3.1 General framework

Let X be the set of compact formulation variables, $|X| = n$. The basic idea of our approach is to start with a subset $\bar{X} \subset X, |\bar{X}| = \bar{n}$ such that the linear relaxation of (CF) is feasible; the problem is reformulated and the master problem is solved via column generation. At this point, either profitable variables in $\widehat{X} := X \setminus \bar{X}$ are dynamically added to the problem, or we prove that the current solution is optimal, in the same spirit of standard column generation.

The clear benefit of this approach is that the associated pricing problem is solved over a smaller set of variables. Furthermore, not all the variables $x_i \in \widehat{X}$ will eventually need to be added.

Without loss of generality and for simplicity of notation we assume that $x = [\bar{x} \,|\, \hat{x}]$,

$c = [\,\bar{c}\,|\,\hat{c}\,]$, $A = [\,\bar{A}\,|\,\widehat{A}\,]$ and $D = [\,\bar{D}\,|\,\widehat{D}\,]$. The *partial compact formulation* (PCF) is defined as follows:

$$\bar{z}_{IP} = \min \quad \bar{c}^T\bar{x} \tag{6.17}$$
$$\text{s.t.} \quad \bar{A}\bar{x} \geq b, \tag{6.18}$$
$$\bar{D}\bar{x} \geq d, \tag{6.19}$$
$$\bar{x} \in \mathbb{Z}^{\bar{n}}_+. \tag{6.20}$$

We remark that $\bar{z}_{IP} \geq z_{IP}$, since we are solving the problem on a subset $\bar{X} \subset X$.

Let $\bar{P} = conv\{\bar{x} \in \mathbb{Z}^{\bar{n}}_+ \,|\, \bar{D}\bar{x} \geq d\} \neq \emptyset$. Again, we can represent each $\bar{x} \in \bar{P}$ as a convex combination of extreme points $\{p_q\}_{\{q\in\bar{Q}\}}$ of $\bar{P}$:

$$\bar{x} = \sum_{q\in\bar{Q}} p_q\lambda_q, \;\; \sum_{q\in\bar{Q}} \lambda_q = 1, \;\; \lambda \in \mathbb{R}^{|\bar{Q}|}_+ \tag{6.21}$$

By substituting $\bar{c}_q = \bar{c}^T p_q$ and $\bar{A}_q = \bar{A}p_q \;\; \forall q \in \bar{Q}$, we can define the equivalent *partial extensive formulation* (PEF) and its linear relaxation, called the *partial master problem* (PMP):

$$\bar{z}_{MP} = \min \quad \sum_{q\in\bar{Q}} \bar{c}_q\lambda_q \tag{6.22}$$
$$\text{s.t.} \quad \sum_{q\in\bar{Q}} \bar{A}_q\lambda_q \geq b, \tag{6.23}$$
$$\sum_{q\in\bar{Q}} \lambda_q = 1, \tag{6.24}$$
$$\lambda \geq 0. \tag{6.25}$$

The partial master problem is solved via standard column generation, and the resulting pricing subproblem is:

$$\min_{q\in\bar{Q}} \tilde{c}_q := \min_{q\in\bar{Q}} \bar{c}_q - \pi\bar{A}_q - \pi_0. \tag{6.26}$$

As soon as $\tilde{c}_q \geq 0$ for all $q \in \bar{Q}$, the current partial master problem is proved to be optimal. However, we still have to determine whether the current partial compact formulation is optimal or not. Therefore, we "price out" compact formulation variables $x \in \widehat{X}$ in order to identify those that are profitable to be added to the (PCF). Indeed, a correct estimation of the contribution of compact formulation variables to the master problem is necessary to make the overall two-stage column generation methodology consistent; we address this specific issue in the next section.

A sketch of the two-stage column generation procedure is outlined in Figure 6.1. In the inner loop (denoted by CG1) we apply standard column generation to solve

Algorithm 1: Two-stage column generation

```
input set X̄
repeat
    repeat
        CG1: generate extensive variables λ for partial master
        problem (PMP)
    until optimal partial master problem (PMP) ;
    CG2: generate compact variables x for partial compact
    formulation (PCF)
until optimal master problem (MP) ;
```

Figure 6.1: *Pseudo-code for two-stage column generation.*

the partial master problem; in particular, the dual optimal vector π is known at every iteration and thus reduced costs $\tilde{c}_q := \bar{c}_q - \pi\bar{A}_q - \pi_0$ of λ variables can be computed exactly; negative reduced-cost columns are provided by the pricing subproblem, if any. In the outer loop (denoted by CG2), compact formulation variables $x_i \in \widehat{X}$ are dynamically added to the partial compact formulation in the same spirit of standard column generation, until optimality is reached.

6.3.2 Contribution of compact formulation variables

Consider the linear relaxation of the compact formulation (6.1)-(6.4) and denote by α and β the dual vectors associated with constraints (6.2) and (6.3) respectively. The reduced cost of x is defined by:

$$\tilde{c}_{CF}(x) := c^T - b^T\alpha - d^T\beta. \tag{6.27}$$

Within the two-stage column generation framework, we will refer to $\tilde{c}_{CF}(x)$ as *compact-formulation reduced cost*.

In the scientific literature, a few contributions have investigated the computation of reduced cost of the compact formulation variables in the context of Dantzig-Wolfe decomposition and column generation.

Walker (1969) illustrates a method that can be applied only if the pricing problem can be solved as a pure linear program, since it makes use of the final tableau to read the dual variables of the pricing. The author provides a formula to compute the reduced cost of compact formulation variables given an optimal dual solution to the master problem and an optimal dual solution to the pricing subproblem. As observed by Irnich et al. (2010), when the pricing subproblem is a shortest path problem, it

can be formulated as a linear program only if the network is acyclic.

Poggi de Aragão and Uchoa (2003) investigate the possibility of adding cuts expressed in terms of compact formulation variables to the master problem, without modifying the pricing structure. They propose keeping explicitly the coupling constraints in the master problem, introducing an alternative Dantzig-Wolfe reformulation, called Explicit Master. The reduced costs of compact formulation variables are therefore represented by the dual variables associated with the coupling constraints in the master problem. However, Irnich et al. (2010) observe that there exists a feasible solution for such a master problem where the reduced costs associated with the coupling constraints are all zero; furthermore, even adding a perturbation does not guarantee the output control and the proposed approach may result in poor reduced cost information.

In two-stage column generation, we need at some point to determine which compact formulation variables $x \in \widehat{X}$ are worth to be added to (PCF). At first, a decision-making rule based on compact reduced cost $\tilde{c}_{CF}(x)$ would appear the most intuitive choice; however, the information provided by $\tilde{c}_{CF}(x)$ is myopic, since it does not take into account the DW decomposition nor the optimal master problem solution, that is the final goal of our two-stage column generation scheme. Indeed, reduced costs $\tilde{c}_{CF}(x)$ would be negative for all variables x that are part of the optimal solution of the linear relaxation of (CF).

On the contrary, we are interested in evaluating the contribution of compact formulation variables in the extensive formulation. In order to achieve this goal, we introduce the concept of *extensive-formulation reduced cost* for compact formulation variables, denoted by $\tilde{c}_{EF}(x)$, that should guide the column generation process in (CG2) according to the optimal master problem solution.

A recent work by Irnich et al. (2010) investigates the relationship in terms of reduced costs between a compact arc-flow formulation and a path-based DW reformulation. In the context of arc-flow variable elimination, the authors prove that arc-reduced costs can be computed from path-reduced costs and propose an efficient bidirectional search technique to compute path-reduced costs.

In our framework, the method for computing the extensive reduced cost $\tilde{c}_{EF}(x)$ is devised according to the specific structure of the pricing subproblem. In particular, we identify two main classes of subproblems:

- **pricing problem with the integrality property**: when the integrality property holds, we can solve the pricing as a linear program and therefore reduced costs of compact formulation variables can be computed straightforwardly using the method by Walker (1969); in this special case, we have that $\tilde{c}_{CF}(x) \equiv \tilde{c}_{EF}(x)$. An example of how to apply the two-stage column generation framework when the pricing problem satisfies the integrality property is illustrated in Section 6.4.1.

- **pricing problem without the integrality property**: in column generation, one generally aims at having pricing problems without the integrality property,

in order to obtain a better bound than the one provided by the linear relaxation of the compact formulation; although this type of pricing problems can be often formulated as an integer program, it is typically solved by more efficient combinatorial algorithms, such as dynamic programming. In particular, many problems, such as those studied in this dissertation, present a path-based pricing structure: in this case, extensive reduced costs $\tilde{c}_{EF}(x)$ of compact formulation variables can be computed using and/or adapting the method by Irnich et al. (2010). An example of how to apply the two-stage column generation framework when the pricing problem doesn't have the integrality property is illustrated in Section 6.4.2.

6.4 Illustration of two-stage column generation

In this section, we illustrate how to apply two-stage column generation and how the contribution of compact formulation variables to the extensive formulation can be estimated. In particular, we distinguish between pricing subproblems with integrality property, that can be solved as pure linear programs, and pricing subproblems without the integrality property that present a path-based structure. For this purpose, we adapt the well-known primer example on resource constrained shortest paths introduced by Desrosiers and Lübbecke (2005).

6.4.1 Pricing problem with the integrality property

Consider a network $G(N, A, c, t)$, where $s \in N$ denotes the origin and $t \in N$ the destination. Every arc $(i, j) \in A$ has a cost c_{ij} and a resource consumption t_{ij}. The available amount of resource is denoted by T. The Resource Constrained Shortest Path Problem (RCSPP) aims to find the minimum-cost path that satisfies the resource constraint.

The RCSPP can be formulated as an integer program:

$$z_{IP} = \min \sum_{(i,j)\in A} c_{ij}x_{ij} \tag{6.28}$$

$$\sum_{j:(s,j)\in A} x_{sj} = 1, \tag{6.29}$$

$$\sum_{j:(j,i)\in A} x_{ji} - \sum_{j:(i,j)\in A} x_{ij} = 0 \quad \forall i \in A,\ i \neq s, t, \tag{6.30}$$

$$\sum_{i:(i,t)\in A} x_{it} = 1, \tag{6.31}$$

$$\sum_{(i,j)\in A} t_{ij}x_{ij} \leq T, \tag{6.32}$$

$$x_{ij} = \{0, 1\} \quad \forall (i, j) \in A. \tag{6.33}$$

where x_{ij} is a binary decision variable, equal to 1 if arc (i, j) is in the path and 0 otherwise. Formulation (6.28)-(6.33) represents our compact formulation.

The Dantzig-Wolfe reformulation proposed by Desrosiers and Lübbecke (2005) relies on the convexification of constraints (6.29)-(6.31). Consider $X = \{x_{ij}$ binary $: (6.29) - (6.31)\}$. An extreme point x_p of the polytope defined by the convex hull of X corresponds to a path $p \in P$ in the network (Ahuja et al., 1993). Therefore, we can express any arc-flow variable x_{ij} as a convex combination of extreme points of P:

$$x_{ij} = \sum_{p \in P} x_{ijp} \lambda_p \, \forall (i, j) \in A; \qquad \sum_{p \in P} \lambda_p = 1; \qquad \lambda_p \geq 0 \, \forall p \in P$$

where λ_p represents the amount of flow on path $p \in P$ and x_{ijp} is a coefficient equal to 1 if arc $(i, j) \in A$ belongs to path $p \in P$.

Now, if we relax the integrality requirements on x_{ij}, the coupling constraints $x_{ij} = \sum_{p \in P} x_{ijp} \lambda_p$ become redundant and the resulting master problem is:

$$z_{MP} = \min \sum_{p \in P} c_p \lambda_p \tag{6.34}$$

$$\sum_{p \in P} t_p \lambda_p \leq T \tag{6.35}$$

$$\sum_{p \in P} \lambda_p = 1 \tag{6.36}$$

$$\lambda_p \geq 0 \quad \forall p \in P. \tag{6.37}$$

where $c_p = \sum_{(i,j) \in A} c_{ij} x_{ijp}$ is the cost associated with path $p \in P$.

We denote π_T and π_0 the dual variables associated with constraints (6.35) and (6.36) respectively. The pricing subproblem is formulated as a Shortest Path Problem:

$$\min \sum_{(i,j) \in A} (c_{ij} - \pi_T t_{ij}) x_{ij} - \pi_0 \tag{6.38}$$

$$- \sum_{j:(s,j) \in A} x_{sj} = -1 \tag{6.39}$$

$$\sum_{j:(j,i) \in A} x_{ji} - \sum_{j:(i,j) \in A} x_{ij} = 0 \quad \forall i \neq s, t \tag{6.40}$$

$$\sum_{i:(i,t) \in A} x_{it} = 1 \tag{6.41}$$

$$x_{ij} = \{0, 1\} \quad \forall (i, j) \in A. \tag{6.42}$$

We remark that formulation (6.38)-(6.42) has the *integrality property*: it means that even if we solve the linear relaxation of (6.38)-(6.42), we still obtain an integer solution. Indeed, the shortest path problem can be expressed as a transshipment problem with one origin and one destination: by shipping one unit from the origin to the destination, the solution determines the shortest path throughout the network.

Dual variables associated with the pricing constraints (6.39)-(6.41) are denoted by μ_s, μ_i and μ_t respectively.

Reduced costs via Walker's method

In this example, the integrality property holds and the pricing is solved as a pure linear program. Therefore, the (CF) linear relaxation and the master problem provide the same optimal solution: it means that, in this special case, the contribution of compact formulation variables to the master problem corresponds to the compact reduced cost of variable x_{ij}, i.e., $\tilde{c}_{CF}(x_{ij}) \equiv \tilde{c}_{EF}(x_{ij})$.

In this case, the compact reduced cost $\tilde{c}_{CF}(x_{ij})$ can be computed exactly using the method proposed by Walker (1969). This method applies when the pricing problem is linear and it computes exactly the reduced cost of a variable x_{ij} in the compact formulation given an optimal dual solution to the master problem and an optimal dual solution to the pricing subproblem.

We recall the method briefly. We consider the linear relaxation of (6.28)-(6.33) and denote by α_s, α_i, α_t and α_T the dual variables associated with constraints (6.29), (6.30), (6.31) and (6.32) respectively. The dual problem of the linear relaxation of the compact formulation is:

$$z_{DP} = \max \alpha_t - \alpha_s + T\alpha_T \tag{6.43}$$

$$\alpha_j - \alpha_i + t_{ij}\alpha_T \leq c_{ij} \quad \forall (i,j) \in A \tag{6.44}$$

$$\alpha_i \in \mathbb{R} \quad \forall i \in N \tag{6.45}$$

$$\alpha_T \leq 0. \tag{6.46}$$

Walker proves that an optimal solution to (6.43)-(6.46) is given by $(\mu_s^*, \mu_i^*, \mu_t^*, \pi_T^*)$, where $(\mu_s^*, \mu_i^*, \mu_t^*)$ is an optimal dual solution to the pricing problem and π_T^* is the optimal dual solution to the master problem associated with constraint (6.35). Furthermore, the reduced cost $\tilde{c}_{CF}(x_{ij})$ of x_{ij} can be computed as:

$$\tilde{c}_{CF}(x_{ij}) = c_{ij} + \mu_i - \mu_j - t_{ij}\pi_T. \tag{6.47}$$

Two-stage column generation We start solving the problem on a reduced network $G(N, \bar{A}, c, t)$ such that subset $\bar{A} \subset A$ ensures that a feasible solution exists, and we denote by $\widehat{A}$ the set of arcs that are not taken into account in the solution process. The associated formulation is the *partial compact formulation.*

At every iteration, we compute the reduced cost $\tilde{c}_{CF}(x_{ij})$ of variables x_{ij} not in the formulation, associated with arcs $(i,j) \in \widehat{A}$, using Walker's method.

If $\tilde{c}_{CF}(x_{uv}) < 0$ for some arc $(u,v) \in \widehat{A}$, the correspondent "column" x_{uv} is added to the partial compact formulation and $\bar{A} = \bar{A} \cup (u,v)$; otherwise, the current partial compact formulation is proved to be optimal.

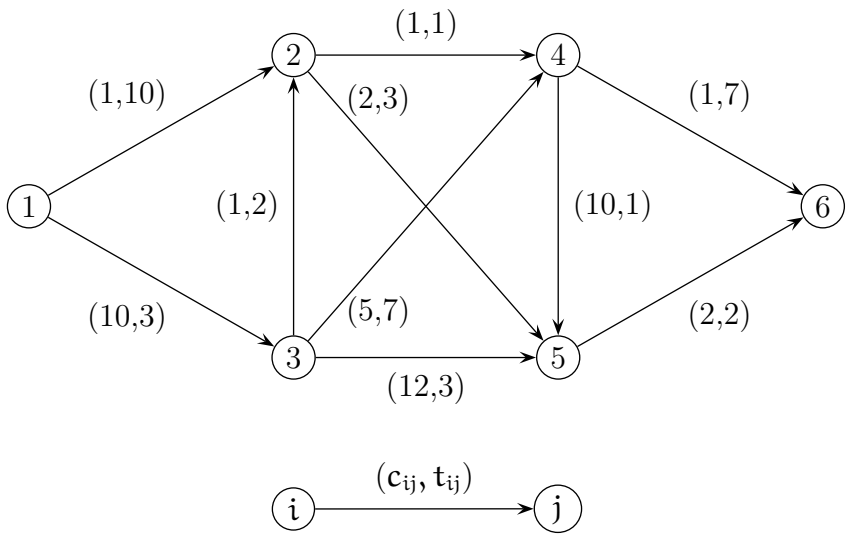

Figure 6.2: *RCSPP with one resource.*

Example

We consider the problem of finding a shortest path from $s = 1$ to $t = 6$ such that the total traversal time of the path does not exceed $T = 14$ time units. The network is illustrated in Figure 6.2.

We can easily enumerate all the paths of this small network:

p	path	cost	time
1	1-2-4-6	3	18
2	1-2-5-6	5	15
3	1-2-4-5-6	14	14
4	1-3-5-6	24	8
5	1-3-4-6	16	17
6	1-3-4-5-6	27	13
7	1-3-2-4-6	13	13
8	1-3-2-4-5-6	24	9
9	1-3-2-5-6	15	10

Some paths are not feasible with respect to the resource constraint ($p = 1, 2, 5$). The minimum cost integer solution for the problem is given by path 13246 with cost $z^*_{IP} = 13$ and resource consumption 13.

The optimal value of the linear relaxation of the compact formulation is $z^*_{LP} = 7$ and the optimal fractional solution is:

$$x^*_{12} = 0.8 \qquad x^*_{13} = 0.2 \qquad x^*_{32} = 0.2 \qquad x^*_{25} = 1 \qquad x^*_{56} = 1.$$

Standard column generation The iterations of standard column generation are reported in the following table:

it	master obj	π_T	π_0	pricing obj	added path
1	100	0	100	-97	1-2-4-6
2	24.555	-5.38889	100	-32.8888	1-3-5-6
3	11.4	-2.1	40.8	-4.8	1-3-2-5-6
4	9	-1.5	30	-2.5	1-2-5-6
5	7	-2	35	0	STOP

The master problem is initialized by an artificial variable y_0 with cost 100. For more details, we refer the reader to Desrosiers and Lübbecke (2005).

Since the integrality property holds, $z^*_{MP} = z^*_{LP} = 7$ and the fractional optimal solutions are equivalent:

$$\lambda^*_{1256} = 0.8 \qquad \lambda^*_{13256} = 0.2$$

Two-stage CG: contribution of non-optimal arcs We start solving the problem on the reduced network $G(N, \bar{A}, c, t)$ where $\bar{A} = \{(1,2), (1,3), (2,5), (3,2), (4,5), (4,6), (5,6)\}$.

In this example, the set $\widehat{A} = \{(2,4), (3,4), (3,5)\}$ is composed of arcs that are all *non-optimal* for the linear relaxation of the compact formulation and for the master problem.

We apply column generation to the reduced problem and we solve a (restricted) partial master problem via standard column generation.

The first iteration of CG2 is reported in the following table:

it	master obj	π_T	π_0	pricing obj	added path
1.1	100	0	100	-95	1-2-5-6
1.2	11.3333	-6.333	100	-21.6667	1-3-2-5-6
1.3	7	-2	35	0	STOP

At this point, we have a partial master problem that is optimal. Now, we want to determine whether the current partial compact formulation is optimal. We therefore compute the reduced cost $\tilde{c}_{CF}(x_{ij})$ of compact formulation variables x_{ij} associated with not-yet-considered arcs $(i,j) \in \widehat{A}$ using Walker's method.

The optimal dual solutions associated with the master and pricing problems are:

$$\mu^*_1 = -16 \qquad \mu^*_2 = 5 \qquad \mu^*_3 = 0 \qquad \mu^*_4 = 4 \qquad \mu^*_5 = 13 \qquad \mu^*_6 = 19 \qquad \pi^*_T = -2$$

Using Walker's procedure we obtain:

$$\begin{aligned}
\tilde{c}_{24} &= c_{24} + \mu_2 - \mu_4 - t_{24}\pi_T = 1 + 5 - 4 - 1 \cdot (-2) = 4 \\
\tilde{c}_{34} &= c_{34} + \mu_3 - \mu_4 - t_{34}\pi_T = 5 + 0 - 4 - 7 \cdot (-2) = 15 \\
\tilde{c}_{35} &= c_{35} + \mu_3 - \mu_5 - t_{35}\pi_T = 12 + 0 - 13 - 3 \cdot (-2) = 5
\end{aligned}$$

Indeed, $\tilde{c}_{CF}(x_{ij}) \geq 0 \ \forall (i,j) \in \widehat{A}$: it means that the current partial compact formulation is optimal and the two-stage column generation scheme terminates with only one iteration of CG2.

Comparing the inner column generation scheme CG1 to standard column generation, we remark a smaller number of iterations (3 vs 5) and a smaller number of generated columns (2 vs 4). Furthermore, the example shows that non-optimal arcs are detected and not added to the problem.

Two-stage CG: contribution of optimal arcs We start solving the problem on the reduced network $G(N, \bar{A}, c, t)$ where $\bar{A} = \{A \setminus \{(1,3),(2,5)\}$.

In this example, the set $\widehat{A} = \{(1,3),(2,5)\}$ is composed of arcs that are all *optimal* for the linear relaxation of the compact formulation and for the master problem.

The first iteration of CG2 is reported in the following table:

it	master obj	π_T	π_0	pricing obj	added path
1.1	100	0	100	-97	1-2-4-6
1.2	24.5555	-5.389	100	-10.5555	1-2-4-5-6
1.3	14	-6.143	100	0.00004	STOP

The optimal dual solutions associated with the master and pricing problems are:

$$\mu_1^* = -100 \qquad \mu_2^* = -37.57 \qquad \mu_3^* = 0 \qquad \mu_4^* = -30.43 \qquad \mu_5^* = -14.28 \qquad \mu_6^* = 0$$

Using Walker's procedure we obtain:

$$\tilde{c}_{CF}(x_{13}) = c_{13} + \mu_1 - \mu_3 - t_{13}\pi_T = 10 - 100 - (0) - 3 \cdot (-6.143) = -71.57$$
$$\tilde{c}_{CF}(X_{25}) = c_{25} + \mu_2 - \mu_5 - t_{25}\pi_T = 2 - 37.57 - (-14.28) - 3 \cdot (-6.143) = -2.857$$

We add to the partial compact formulation the variable with the most negative reduced cost, i.e., x_{13} and we iterate. In particular, $\bar{A} = \{\bar{A} \cup (1,3)\}$.

The second iteration of CG2 is reported in the following table:

it	master obj	π_T	π_0	pricing obj	added path
2.1	14	-6.14286	100	-26.857	1-3-5-6
2.2	11.4	-2.1	40.8	-0.5	1-3-2-4-6
2.3	11	-2	39	0	STOP

The optimal dual solutions associated with the master and pricing problems are:

$$\mu_1^* = -16 \qquad \mu_2^* = 5 \qquad \mu_3^* = 0 \qquad \mu_4^* = 8 \qquad \mu_5^* = 18 \qquad \mu_6^* = 23 \qquad \pi_T^* = -2$$

Using Walker's procedure we obtain:

$$\tilde{c}_{CF}(x_{25}) = c_{25} + \mu_2 - \mu_5 - t_{25}\pi_T = 2 + 5 - 18 - 3 \cdot (-2) = -5$$

As expected, $\tilde{c}_{CF}(x_{25}) < 0$, therefore variable x_{25} is added to the partial compact formulation and $\bar{A} = \{\bar{A} \cup (2,5)\}$. We remark that, since $\bar{A} \equiv A$, the third CG2 iteration is the final one, since it corresponds to a run of standard column generation;

therefore, two-stage column generation has required an additional computational effort in this special case. However, the example shows that optimal arcs are detected correctly. Furthermore, the number of CG2 iterations may be reduced by implementing smarter strategies for adding columns. This analysis is out of the scope of this example, but will be treated later in the chapter.

Two-stage CG: simultaneous contribution of optimal and non-optimal arcs We start solving the problem on the reduced network $G(N, \bar{A}, c, t)$ where $\bar{A} = \{(1,2),(1,3),(3,2),(4,5),(4,6),(5,6)\}\}$.

In this example, the set $\widehat{A} = \{(2,4),(2,5),(3,4),(3,5)\}$ is composed of arc (2,5) that is *optimal* for the linear relaxation of the compact formulation and the master problem, plus three arcs that are *non-optimal*.

Furthermore, the initial network is disconnected. Therefore, the first iteration of CG2 stops in one iteration, with the artificial variable y_0 equal to one in the optimal solution.

it	master obj	π_T	π_0	pricing obj	added path
1.1	100	0	100	0	STOP

The optimal dual solutions associated with the master and pricing problems are:

$$\mu_1^* = 0 \qquad \mu_2^* = 0 \qquad \mu_3^* = 10 \qquad \mu_4^* = 99 \qquad \mu_5^* = 98 \qquad \mu_6^* = 100 \qquad \pi_T = 0$$

Using Walker's procedure we obtain:

$$\begin{aligned}
\tilde{c}_{CF}(x_{24}) &= c_{24} + \mu_2 - \mu_4 - t_{24}\pi_T = 1 + 0 - 99 - 1 \cdot (0) = -98 \\
\tilde{c}_{CF}(x_{25}) &= c_{25} + \mu_2 - \mu_5 - t_{25}\pi_T = 2 + 0 - 98 - 3 \cdot (0) = -96 \\
\tilde{c}_{CF}(x_{34}) &= c_{34} + \mu_3 - \mu_4 - t_{34}\pi_T = 5 + 10 - 99 - 7 \cdot (0) = -84 \\
\tilde{c}_{CF}(x_{35}) &= c_{35} + \mu_3 - \mu_5 - t_{35}\pi_T = 12 + 10 - 98 - 3 \cdot (0) = -76
\end{aligned}$$

Therefore, variable x_{24} is added to the partial compact formulation and $\bar{A} = \{\bar{A} \cup (2,4)\}$.

The second iteration of CG2 is reported in the following table:

it	master obj	π_T	π_0	pricing obj	added path
2.1	100	0	100	-97	1-2-4-6
2.2	24.55	-5.38	100	-27.5	1-3-2-4-5-6
2.3	12.33	-2.33	45	-1.67	1-3-2-4-6
2.4	11	-2	39	0	STOP

The optimal dual solutions associated with the master and pricing problems are:

$$\mu_1^* = -16 \qquad \mu_2^* = 5 \qquad \mu_3^* = 0 \qquad \mu_4^* = 8 \qquad \mu_5^* = 20 \qquad \mu_6^* = 23 \qquad \pi_T^* = -2$$

Using Walker's procedure we obtain:

$$\tilde{c}_{CF}(x_{25}) = c_{25} + \mu_2 - \mu_5 - t_{25}\pi_T = 2 + 5 - 20 - 3 \cdot (-2) = -7$$
$$\tilde{c}_{CF}(x_{34}) = c_{34} + \mu_3 - \mu_4 - t_{34}\pi_T = 5 + 0 - 8 - 7 \cdot (-2) = 11$$
$$\tilde{c}_{CF}(x_{35}) = c_{35} + \mu_3 - \mu_5 - t_{35}\pi_T = 12 + 0 - 20 - 3 \cdot (-2) = -2$$

Therefore, variable x_{25} is added to the partial compact formulation and $\bar{A} = \{\bar{A} \cup (2,5)\}$.

The third iteration of CG2 is reported in the following table:

it	**master obj**	π_T	π_0	**pricing obj**	**added path**
3.1	11	-2	39	-4	1-2-5-6
3.2	8.16	-3.16	52.5	-5.9	1-3-2-5-6
3.3	7	-2	35	0	STOP

The optimal dual solutions associated with the master and pricing problems are:

$$\mu_1^* = -16 \qquad \mu_2^* = 5 \qquad \mu_3^* = 0 \qquad \mu_4^* = 4 \qquad \mu_5^* = 13 \qquad \mu_6^* = 19 \qquad \pi_T^* = -2$$

Using Walker's procedure we obtain:

$$\tilde{c}_{CF}(x_{34}) = c_{34} + \mu_3 - \mu_4 - t_{34}\pi_T = 5 + 0 - 4 - 7 \cdot (-2) = 15$$
$$\tilde{c}_{CF}(x_{35}) = c_{35} + \mu_3 - \mu_5 - t_{35}\pi_T = 12 + 0 - 13 - 3 \cdot (-2) = 5$$

We see that $\tilde{c}_{CF}(x_{ij}) \geq 0 \ \forall (i,j) \in \widehat{A}$: it means that the current partial compact formulation is optimal and the two-stage column generation scheme terminates. We remark that 3 iterations of CG2 have been performed: 2 out of 4 arcs have been added, one optimal and one non-optimal.

6.4.2 Pricing problem without the integrality property

Consider the Resource Constrained Shortest Path Problem introduced in the previous section, with the addition of one resource, that is capacity Q. The compact

formulation therefore presents an additional resource constraint and becomes:

$$z_{IP} = \min \sum_{(i,j)\in A} c_{ij}x_{ij} \tag{6.48}$$

$$\sum_{j:(s,j)\in A} x_{sj} = 1, \tag{6.49}$$

$$\sum_{j:(j,i)\in A} x_{ji} - \sum_{j:(i,j)\in A} x_{ij} = 0 \quad \forall i \in A,\ i \neq s,t, \tag{6.50}$$

$$\sum_{i:(i,t)\in A} x_{it} = 1, \tag{6.51}$$

$$\sum_{(i,j)\in A} q_{ij}x_{ij} \leq Q, \tag{6.52}$$

$$\sum_{(i,j)\in A} t_{ij}x_{ij} \leq T, \tag{6.53}$$

$$x_{ij} = \{0,1\} \quad \forall (i,j) \in A. \tag{6.54}$$

Since we are interested in obtaining a pricing problem without the integrality property, we propose to "convexify" the set of constraints (6.49)-(6.52), i.e., we handle resource T in the master problem and resource Q in the pricing subproblem. We remark that this is not a "smart" way to convexify the resource constraints, since they would typically be handled together in the pricing subproblem (combinatorial algorithms such as dynamic programming are particularly suited to take into account resources); however, we introduce this decomposition for illustration purposes.

The formulation of the master problem is unchanged with respect to the previous example:

$$z_{MP} = \min \sum_{p\in P} c_p\lambda_p \tag{6.55}$$

$$\sum_{p\in P} t_p\lambda_p \leq T \tag{6.56}$$

$$\sum_{p\in P} \lambda_p = 1 \tag{6.57}$$

$$\lambda_p \geq 0 \quad \forall p \in P. \tag{6.58}$$

On the contrary, the pricing subproblem now takes into account resource Q and

is therefore formulated as a Resource Constrained Shortest Path Problem:

$$\min \sum_{(i,j)\in A} (c_{ij} - \pi_T t_{ij}) x_{ij} - \pi_0 \tag{6.59}$$

$$- \sum_{j:(s,j)\in A} x_{sj} = -1 \tag{6.60}$$

$$\sum_{j:(j,i)\in A} x_{ji} - \sum_{j:(i,j)\in A} x_{ij} = 0 \quad \forall i \neq s,t \tag{6.61}$$

$$\sum_{i:(i,t)\in A} x_{it} = 1 \tag{6.62}$$

$$\sum_{(i,j)\in A} q_{ij} x_{ij} \leq Q \tag{6.63}$$

$$x_{ij} = \{0,1\} \quad \forall (i,j) \in A. \tag{6.64}$$

Reduced costs via Irnich's method

The pricing subproblem is based on shortest paths, therefore we can use the method by Irnich et al. (2010) to estimate the contribution $\tilde{c}_{EF}(x_{ij})$ of compact formulation variables x_{ij} to master problem.

We denote by $\tilde{c}_p$ the reduced cost of path $p \in P$ and by $\mathcal{F}_{ij}$ the set of all paths that use arc $(i,j) \in A$. In other words $\mathcal{F}_{ij} = \{p \in P : x_{ijp} = 1\}$.

According to Irnich et al. (2010), we can estimate the extensive reduced cost $\tilde{c}_{EF}(x_{ij})$ of arc-flow variable x_{ij} as follows:

$$\tilde{c}_{EF}(x_{ij}) = \min_{p\in\mathcal{F}_{ij}} \tilde{c}_p. \tag{6.65}$$

In other words, $\tilde{c}_{EF}(x_{ij})$ is given by the minimum reduced cost of any path passing by arc (i,j). If $\mathcal{F}_{ij} = \{\emptyset\}$, i.e., if there is no path using arc (i,j), then the contribution of arc (i,j) in the master problem solution is null and therefore we set $\tilde{c}_{EF}(x_{ij}) = 0$.

Two-stage column generation As in the previous example, the initialization is given by a subset of arcs $\bar{A} \subset A$ such that the problem is feasible. We start solving the problem on the reduced network $G(N, \bar{A}, c, t)$, and we denote by $\widehat{A}$ the set of arcs that are not taken into account in the solution process. The associated formulation is the *partial compact formulation.*

At every iteration, the extensive reduced cost $\tilde{c}_{EF}(x_{ij})$ of variables x_{ij} associated with arcs $(i,j) \in \widehat{A}$ not in the formulation is estimated according to (6.65).

If $\tilde{c}_{EF}(x_{uv}) < 0$ for some arc $(u,v) \in \widehat{A}$, the correspondent "column" x_{uv} is added to the partial compact formulation and $\bar{A} = \bar{A} \cup (u,v)$; otherwise, the current partial compact formulation is proved to be optimal.

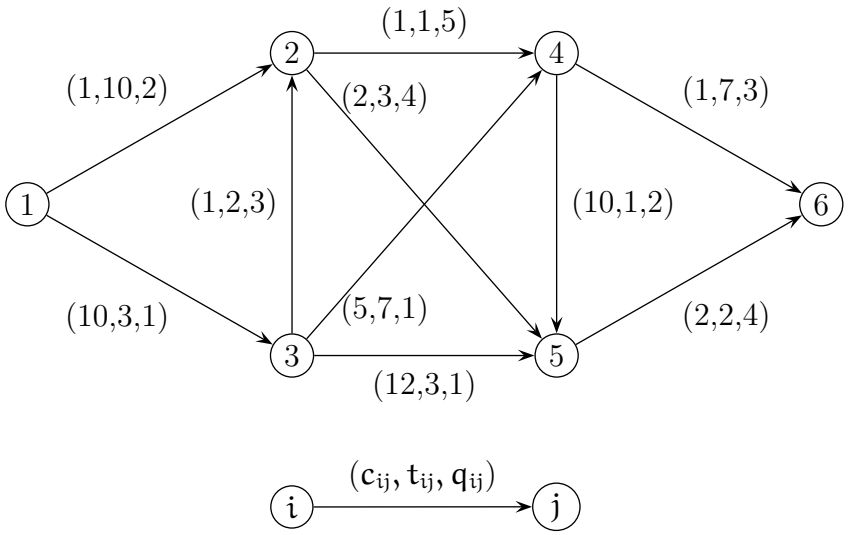

Figure 6.3: *RCSPP with two resources.*

Example

We consider the problem of finding a shortest path from $s = 1$ to $t = 6$ such that the total traversal time of the path does not exceed $T = 14$ time units and the total capacity of the path does not exceed $Q = 10$. The network is illustrated in Figure 6.3.

We can easily enumerate all the paths of this small network:

p	path	cost	time	capacity
1	1-2-4-6	3	18	10
2	1-2-5-6	5	15	10
3	1-2-4-5-6	14	14	13
4	1-3-5-6	24	8	6
5	1-3-4-6	16	17	5
6	1-3-4-5-6	27	13	6
7	1-3-2-4-6	13	13	12
8	1-3-2-4-5-6	24	9	15
9	1-3-2-5-6	15	10	12

Some paths are not feasible with respect to the time and/or capacity constraint. The min-cost integer solution for the problem is given by path 1356 with cost $z^*_{IP} = 24$, time consumption 8 and capacity consumption 6.

The optimal value of the linear relaxation of the compact formulation is $z^*_{LP} = 7.2941$ and the optimal fractional solution is:

$$x^*_{12} = 0.82 \qquad x^*_{13} = 0.18 \qquad x^*_{25} = 0.94 \qquad x^*_{32} = 0.12 \qquad x^*_{35} = 0.06 \qquad x^*_{56} = 1$$

corresponding to paths 1-2-5-6 (flow 0.82), 1-3-2-5-6 (flow 0.12) and 1-3-5-6 (flow 0.06).

Standard column generation The iterations of standard column generation are reported in the following table:

it	master obj	π_0	π_T	pricing obj	added path
1	100	100	0	-97.00	1-2-4-6
2	24.5	100	-5.39	-32.88	1-3-5-6
3	11.4	40.8	-2.10	-4.3	1-2-5-6
4	7.71	45.7	-2.71	0.0	STOP

The master problem is initialized by an artificial variable y_0 with cost 100. For more details, we refer the reader to Desrosiers and Lübbecke (2005).

The optimal value of the master problem is $z^*_{MP} = 7.71$ and the optimal fractional solution consists of two paths:

$$\lambda^*_{1256} = 0.86 \qquad \lambda^*_{1356} = 0.14.$$

We remark that, since the integrality property does not hold, $z^*_{MP} > z^*_{LP}$.

Two-stage CG: contribution of non-optimal arcs We start solving the problem on the reduced network $G(N, \bar{A}, c, t)$ where $\bar{A} = \{A \setminus (4,6)\}$.

In this example, arc $(4,6) \in \widehat{A}$ is *non-optimal* both for the linear relaxation of the compact formulation and for the master problem.

We apply column generation to the reduced problem and we solve a (restricted) partial master problem via standard column generation.

The first iteration of CG2 is reported in the following table:

it	master obj	π_0	π_T	pricing obj	added path
1.1	100.00	100.00	0.00	-95.00	1-2-5-6
1.2	11.33	100.00	-6.33	-25.33	1-3-5-6
1.3	7.71	45.71	-2.71	0.00	STOP

The pricing subproblem is solved by enumeration, due to the small size of the network.

At this point, we have a partial master problem that is optimal. Now, we want to determine whether the current partial compact formulation is optimal. We therefore estimate the contribution of arc $(4,6)$ to the master problem using Irnich et al.'s (2010) method.

All the paths and the associated contributions are reported in the following table:

path	$\tilde{c}_p$	q_p	*note*
1-2-4-6	6.14	10	
1-2-5-6	0.00	10	
1-2-4-5-6	6.29	13	$q_p > Q$
1-3-5-6	0.00	6	
1-3-4-6	16.43	5	
1-3-4-5-6	16.57	6	
1-3-2-4-6	2.57	12	$q_p > Q$
1-3-2-4-5-6	2.71	15	$q_p > Q$
1-3-2-5-6	-3.57	12	$q_p > Q$

We remark that only feasible paths with respect to capacity are considered. According to (6.65), the extensive reduced cost of x_{46} is given by:

$$\tilde{c}_{EF}(x_{46}) = \min\{\tilde{c}_{12462}, \tilde{c}_{1346}\} = \tilde{c}_{1246} = 6.14. \tag{6.66}$$

Indeed, $\tilde{c}_{EF}(x_{46}) \geq 0$: it means that the current partial compact formulation is optimal and the two-stage column generation scheme terminates with only one iteration of CG2.

Comparing the inner column generation scheme CG1 to standard column generation, we remark a smaller number of iterations (3 vs 4) and a smaller number of generated columns (2 vs 3). Furthermore, the example shows that non-optimal arcs are detected and not added to the problem.

Two-stage CG: contribution of optimal arcs We start solving the problem on the reduced network $G(N, \bar{A}, c, t)$ where $\bar{A} = \{A \setminus (5, 6)\}$.

In this example, arc $(5, 6) \in \widehat{A}$ is *optimal* both for the linear relaxation of the compact formulation and for the master problem.

We apply column generation to the reduced problem and we solve a (restricted) partial master problem via standard column generation.

The first iteration of CG2 is reported in the following table:

it	**master obj**	π_0	π_T	**pricing obj**	**added path**
1.1	100.00	100	0.00	-97.00	1-2-4-6
1.2	24.55	100	-5.39	0.00	STOP

At this point, we have a partial master problem that is optimal. Now, we want to determine whether the current partial compact formulation is optimal. We therefore compute the contribution of arc $(5, 6) \in \widehat{A}$ to the master problem using Irnich et al.'s (2010) method.

All the paths and the associated contributions are reported in the following table:

path	$\tilde{c}_p$	q_p	*note*
1-2-4-6	0.00	10	
1-2-5-6	-14.17	10	
1-2-4-5-6	-10.56	13	$q_p > Q$
1-3-5-6	-32.89	6	
1-3-4-6	7.61	5	
1-3-4-5-6	-2.94	6	
1-3-2-4-6	-16.94	12	$q_p > Q$
1-3-2-4-5-6	-27.50	15	$q_p > Q$
1-3-2-5-6	-31.11	12	$q_p > Q$

We remark that only feasible paths with respect to capacity are considered. According to (6.65), the extensive reduced cost of x_{56} is given by:

$$\tilde{c}_{EF}(x_{56}) = \min\{\tilde{c}_{1256}, \tilde{c}_{1356}, \tilde{c}_{13456}\} = \tilde{c}_{1356} = -32.89. \tag{6.67}$$

As expected, $\tilde{c}_{EF}(x_{56}) < 0$, therefore variable x_{56} is added to the partial compact formulation and $\bar{A} = \{\bar{A} \cup (5,6)\}$. We remark that, since $\bar{A} \equiv A$, the second CG2 iteration is the final one, since it corresponds to a run of standard column generation; therefore, two-stage column generation has required an additional computational effort in this special case. However, the example shows that optimal arcs are detected correctly.

Two-stage CG: contribution of an arc that is optimal for the relaxed LP and non-optimal for the master problem We start solving the problem on the reduced network $G(N, \bar{A}, c, t)$ where $\bar{A} = \{A \setminus (3,2)\}$.

In this example, arc $(3,2) \in \widehat{A}$ is *optimal* for the linear relaxation of the compact formulation but is *non-optimal* for the master problem.

We apply column generation to the reduced problem and we solve a (restricted) partial master problem via standard column generation.

The first iteration of CG2 is reported in the following table:

it	**master obj**	π_0	π_T	**pricing obj**	**added path**
1.1	100.00	100.00	0.00	-97.00	1-2-4-6
1.2	24.55	100.00	-5.39	-32.89	1-3-5-6
1.3	11.40	40.80	-2.10	-4.30	1-2-5-6
1.4	7.71	45.71	-2.71	0.00	STOP

At this point, we have a partial master problem that is optimal. Now, we want to determine whether the current partial compact formulation is optimal. We therefore compute the contribution of arc $(3,2) \in \widehat{A}$ to the master problem using Irnich et al.'s (2010) method.

All the paths and the associated contributions are reported in the following table:

path	$\tilde{c}_p$	q_p	*note*
1-2-4-6	51.86	10	
1-2-5-6	45.71	10	
1-2-4-5-6	52.00	13	$q_p > Q$
1-3-5-6	45.71	6	
1-3-4-6	62.14	5	
1-3-4-5-6	62.28	6	
1-3-2-4-6	48.28	12	$q_p > Q$
1-3-2-4-5-6	48.43	15	$q_p > Q$
1-3-2-5-6	42.14	12	$q_p > Q$

No path that make use of arc (3,2) is feasible with respect to the capacity constraint. Therefore, $\mathcal{F}_{32} = \{\emptyset\}$ and the extensive reduced cost is $\tilde{c}_{EF}(x_{32}) = 0.0$.

Since $\tilde{c}_{EF}(x_{32}) \geq 0$, the current partial compact formulation is optimal and the two-stage column generation scheme terminates with only one iteration of CG2.

Remarks The last example is very interesting, since it clearly shows how extensive reduced costs for compact formulation variables are able to guide the optimization towards the optimal solution of the master problem, and not towards the optimal solution of the linear relaxation of (CF).

Remarkably, arc $(3, 2)$ is not added to the partial compact formulation, in spite of the fact that it is optimal in the (CF) linear relaxation. This result is not trivial.

As observed by Desrosiers and Lübbecke (2005), an alternative procedure to obtain reduced costs of compact formulation variables is to directly keep coupling constraints in the formulation (Poggi de Aragão and Uchoa, 2003) and impose $x \geq \epsilon$ for a small $\epsilon > 0$. The shadow prices of these constraints are the reduced costs of x.

Indeed, using this technique for variable x_{32} we obtain a reduced cost of -3.57143: therefore, using this estimation, arc $(3, 2)$ would have been added.

6.5 Application to DSDVRPTW

In this section, we illustrate how the two-stage column generation framework can be applied to the Discrete Split Delivery Vehicle Routing Problem with Time Windows introduced in Chapter 5.

The arc-flow model presented in Section 5.4 for the DSDVRPTW represents our compact formulation. We apply Dantzig-Wolfe decomposition and we obtain the master problem and the pricing subproblem introduced in Section 5.5. In particular, the pricing subproblem is a Resource Constrained Shortest Path Problem that is solved by bidirectional bounded dynamic programming (Righini and Salani, 2006).

Computational results have confirmed that the problem is complex and the branch-and-price algorithm is not able to converge for large-size instances. In particular, as soon as the number of orders increase, the problem becomes much more difficult to

solve, despite the advanced and sophisticated techniques that we implemented for accelerating the master and the pricing problem.

The problem structure is particularly suited to apply two-stage column generation and we attempt to overcome the intrinsic complexity of the problem by exploiting our new framework.

The main issue is typically represented by the estimation of extensive reduced costs in the CG2 step of the two-stage scheme. Given the structure of the pricing problem for the DSDVRPTW, we can adapt Irnich et al.'s (2010) method to estimate the contribution of compact formulation variables to the extensive formulation. Their technique is based on bidirectional dynamic programming *without* bounding and has been proposed and successfully applied to arc-flow variables. For the specific implementation details, we refer the reader to (Irnich et al., 2010; Irnich, 2010).

In order to avoid confusion, we refer to the dynamic programming algorithm developed for computing extensive reduced costs as *CG2 dynamic programming*, according to the fact that this computation occurs at the CG2 step of two-stage column generation. Although the implementation proposed by Irnich et al. (2010) relies on exact dynamic programming, we also propose a relaxed version of the CG2 DP, mainly motivated by efficiency reasons and reduction of the computational effort. Further details are provided in section 6.6.1.

The contribution of variables x_{ij} associated with arcs $(i,j) \in E$ to the master problem solution is computed as follows. Let $\tilde{c}_p$ be the reduced cost of route $p \in P$ as defined in section 5.5.2, and let $\mathcal{F}_{ij} \subseteq P$ be the set of all routes that make use of arc $(i,j) \in E$. The extensive reduced cost of variable x_{ij} is estimated by:

$$\tilde{c}_{EF}(x_{ij}) = \min_{p \in \mathcal{F}_{ij}} \tilde{c}_p, \tag{6.68}$$

that is the minimum reduced cost among any route passing by arc (i,j).

In two-stage column generation, we start considering a subset of arcs $\bar{E} \subset E$ such that a feasible solution exists, and we define $\widehat{E} = E \setminus \bar{E}$ the set of arcs not yet taken into account in the solution process. During two-stage column generation, variables x_{ij} associated with arcs $(i,j) \in \widehat{E}$ are dynamically added to the problem according to their extensive reduced cost, until no variable with negative reduced cost exists.

At every step of the algorithm, we assume to have an upper bound ub available (typically obtained with primal heuristics) and a valid lb. The quantity $ub - lb$ is referred to as *optimality gap*. According to their status, we can always classify compact formulation variables in three groups:

- **active variables**: with this term, we refer to variables x_{ij} that are in the formulation, i.e., variables associated with the subset of arcs $(i,j) \in \bar{E}$; active variables are either in the formulation since the initialization, or they have been added during the two-stage column generation process;

- **inactive variables**: with this term, we refer to variables x_{ij} that are not in

the formulation, i.e., variables associated with the subset of arcs $(i,j) \in \widehat{E}$; inactive variables present a positive reduced cost, therefore they are not taken into account in the formulation and the correspondent arcs are removed from the underlying network; the more active variables, the more the problem size is reduced and we gain in speed-up;

- **suboptimal variables**: with this term, we refer to variables x_{ij} that are not in the formulation, i.e., variables associated with the subset of arcs $(i,j) \in \widehat{E}$, and that are proved to be suboptimal, according to their reduced cost: indeed, if a variable has a reduced cost greater than the optimality gap, this variable is suboptimal and can be definitely eliminated from the formulation. The distinction between inactive and suboptimal variables make sense in the context of a branch-and-price algorithm, where two-stage column generation is applied at every node of the search tree: in this case, all the variables that are suboptimal at the end of the root node can be eliminated from the problem and not taken into account in the child nodes; on the contrary, variables that are inactive at the end of the root node, are still candidate to be added to the formulation in the child nodes.

We also apply two-stage column generation to another type of compact formulation variables, since our framework is defined for a general variable belonging to the compact formulation, and not only to arc-flow variables. In particular, we analyze order-selection variables y_c associated with orders $c \in C$.

We extend the method proposed by Irnich et al. (2010) for arc-flow variables to handle also order-selection variables: the extension requires to modify the domination rule in the CG2 dynamic programming, in order to obtain a least reduced cost path for every arc $(i,j) \in E$ and for every order $c \in C$.

The extensive reduced cost of y_c is computed as follows. Let $\mathcal{F}_c \subseteq P$ be the set of all routes that deliver order $c \in C$. The reduced cost of variable y_c is estimated as:

$$\tilde{c}_{EF}(y_c) = \min_{p \in \mathcal{F}_c} \tilde{c}_p, \tag{6.69}$$

that is the minimum reduced cost among any route delivering order c.

As for arc-flow variables, we start considering a subset of orders $\bar{C} \subset C$ such that a feasible solution exists, and we define $\widehat{C} = C \setminus \bar{C}$ the set of orders not yet taken into account in the solution process. During two-stage column generation, variables y_c associated with orders $c \in \widehat{C}$ are dynamically added to the problem according to their extensive reduced cost. Furthermore, throughout the solution process we can distinguish among active, inactive and suboptimal variables.

6.6 Computational experiments

In this section, we present the computational experiments that we have carried out on the Discrete Split Delivery Vehicle Routing Problem, with the primary purpose of validating our two-stage column generation framework. Problems belonging to the class of vehicle routing, such as the DSDVRPTW, represent indeed a well accepted benchmark for testing new approaches in column generation and branch-and-price schemes.

In this analysis, we mainly focus on the root node and on the optimal master problem solution, and we compare standard column generation to our two-stage scheme. The validity of our framework is confirmed by the fact that we obtain the same lower bound at the end of the root node; furthermore, by applying two-stage column generation we expect to generate a lower number of columns (namely, only "good" columns) and to significantly speed-up the solution process.

Two-stage column generation for the DSDVRPTW is implemented in ANSI C and compiled with gcc 4.1.2. All restricted master problems are solved using ILOG CPLEX version 10.2. Computational experience is run under a linux operating system on a 2Ghz Intel processor equipped with 2GB of RAM.

Computational tests are performed on a subset of instances described in Chapter 5, section 5.7.1. We start the analysis of 25 customers by selecting instances of class R1_25_C_100, in order to deal with the maximum number of orders (scenario C); for the analysis of 50 customers, we start with instances of class R1_50_A_50, since the increased number of customers makes the instances complex already with scenario A. The time limit is set to one hour for all tests.

We test three possible initialization strategies for the two-stage framework:

- **opt_basis**: we initialize $\bar{\mathsf{E}}$ and $\bar{\mathsf{C}}$ according to the arcs and orders that compose the optimal basis for the master problem; this corresponds to the "best case analysis", since in principle it shouldn't be profitable to add more arcs and orders to the compact formulation.

- **opt_master**: we initialize $\bar{\mathsf{E}}$ and $\bar{\mathsf{C}}$ according to the arcs and orders that belong to columns with positive flow in the optimal master solution; this corresponds to the "best guess" that we may have of the optimal solution; it is reasonable to assume that such a good solution may be provided by a fast primal heuristic, that is nowadays a standard component of state-of-the-art solvers based on column generation and branch-and-price.

- **opt_lp**: we initialize $\bar{\mathsf{E}}$ and $\bar{\mathsf{C}}$ according to the arcs and orders taken with positive value in the optimal solution of the linear relaxation of the compact formulation; this corresponds to a "bad initialization" but that we assume to be easily available.

The computation of 2-path inequalities is deactivated, in order to ensure a fair comparison between standard and two-stage column generation.

In this section we present a summary of the computational experiments. Detailed computational results are provided in Appendix B.

6.6.1 Exact CG2 dynamic programming

Table 6.1 provides a summary of the comparison between standard column generation and two-stage column generation at the root node. For each tested class we report the total number of instances (nr) and, for each method, the number of instances solved (sol), the average number of generated columns (cols) and the average computational time in seconds (t). The two-stage framework has been tested using three initializations (opt_basis, opt_master, opt_lp) and exact CG2 dynamic programming for the computation of extensive reduced costs.

The reduction of generated columns is huge and amounts to 42% on average: opt_master is the initialization that reaches the higher average reduction (62%), while surprisingly opt_basis only allows for 46% of reduced columns: although the result is still significant, opt_basis was supposed to be the best initialization. On the contrary, we remark that in general opt_master outperforms opt_basis in terms of columns and computational time, although opt_basis requires a lower number of CG2 iterations to prove optimality. The behavior of the LP initialization is unstable and the number of generated columns is even increased for instance r101_50_A_50; as expected, it proves to be the worst initialization for our method. Class R1_25_C_100 obtains a higher reduction on average than class R1_50_A_50 (58% vs 26%): we explain this result by the fact that scenario C presents a higher number of order partitions per customer than scenario A. Two-stage CG has therefore the possibility to keep out of the formulation a higher number of order partitions and only "good" columns are generated.

Unfortunately, the proposed method requires a huge computational effort with respect to standard column generation and not all instances can be solved within the time limit of one hour. Therefore, although efficient in terms of reduced number of columns, this implementation of two-stage column generation is not competitive in terms of computational time.

In particular, we remark that almost 100% of time is spent for computing the

		Stand.CG			**Two-stage column generation (exact CG2 DP)**								
					opt_basis			**opt_master**			**opt_lp**		
Class	**nr**	**sol**	**cols**	**t**	**sol**	**cols**	**t**	**sol**	**cols**	**t**	**sol**	**cols**	**t**
R1_25_C_100	12	12	3165	18	8	943	1386	8	778	713	2	689	135
R1_50_A_50	12	12	1789	37	3	887	834	3	525	593	1	1143	185

Table 6.1: *Standard vs Two-stage column generation: summary of results for exact CG2 dynamic programming.*

reduced cost of compact formulation variables, that corresponds to the dynamic programming algorithm adapted from Irnich et al. (2010). We explain this behavior by the modified domination rule: indeed, a much higher number of states survive in the label extension step, and this usually slows down significantly the overall dynamic programming algorithm. Therefore, we target our attempts to speed up the CG2 step of our two-stage column generation scheme.

6.6.2 Relaxed CG2 dynamic programming

In order to speed up the CG2 step and the whole two-stage column generation, we implemented a relaxed CG2 dynamic programming algorithm to compute the extensive reduced cost of compact formulation variables. Specifically, we remove the elementarity requirement on paths and we solve a RCSPP instead of a RCESPP. As a consequence, we obtain a lower bound to $\mathsf{LB}(\tilde{c}_{\mathsf{EF}}(\cdot))$, that is still valid and consistent with our method.

Table 6.1 provides a summary of the results using the relaxed CG2 dynamic programming implementation. We remark at a glance the huge speed up yielded by the proposed relaxation with respect to the exact approach. All tested instances are solved within the time limit, opposite to roughly 35% instances solved by the exact CG2 DP.

Table 6.3 allows for a better comparison between exact and relaxed CG2 dynamic programming for two-stage column generation. The average number of CG2 iterations (`it`), the average reduction of columns (%`col`) with respect to standard CG, the average percentage of active orders (%`ord`) and active arcs (%`arc`) at the end of the root node are reported. Expectedly, a higher number of compact formulation variables is active in the final formulation, and a higher number of CG2 iterations is required to prove optimality. However, we don't lose so much in terms of quality: surprisingly, the reduction of generated columns is comparable to the exact case, except for the `opt_lp` initialization, where we observe a poorer performance.

Concerning the `opt_basis` initialization, the percentage of active variables increases from 62% to 88% for orders and from 13% to 24% for arcs on average for class `R1_25_C_100`. Generated columns are reduced steadily both for exact and relaxed

		Stand.CG			**Two-stage column generation (relaxed CG2 DP)**								
					opt_basis			**opt_master**			**opt_lp**		
Class	**nr**	**sol**	**cols**	**t**	**sol**	**cols**	**t**	**sol**	**cols**	**t**	**sol**	**cols**	**t**
R1_25_C_100	12	12	3165	18	12	1061	72	12	971	88	12	1619	140
R1_50_A_50	12	12	1789	37	12	931	24	12	667	28	12	1690	192

Table 6.2: *Standard vs Two-stage column generation: summary of results for relaxed CG2 dynamic programming.*

	exact CG2 DP				relaxed CG2 DP			
Class	**%col**	**it**	**%ord**	**%arc**	**%col**	**it**	**%ord**	**%arc**
opt_basis								
R1_25_C_100	-70%	6.75	62%	13%	-66%	11.25	88%	24%
R1_50_A_50	-50%	6.67	96%	7%	-48%	13.50	98%	9%
opt_master								
R1_25_C_100	-75%	9.75	59%	13%	-69%	14.83	85%	23%
R1_50_A_50	-71%	6.33	86%	6%	-63%	14.92	92%	9%
opt_lp								
R1_25_C_100	-78%	10.00	45%	7%	-49%	15.25	88%	24%
R1_50_A_50	-36%	16.00	100%	7%	-6%	26.50	100%	11%

Table 6.3: *Comparison between exact and relaxed CG2 dynamic programming.*

DP by more than 50%. The computational time of two-stage CG with relaxed DP is often higher than standard column generation, although within the same order of magnitude. For class R1_50_A_100, relaxed DP appears to be competitive in terms of quality with exact DP: indeed, the number of active variables as well as the number of generated columns is almost unchanged. Furthermore, the two-stage approach allows for time savings (36% on average) and 48% of column reduction with respect to standard column generation. These results are promising, since in addition to the validation of our framework, we are also able to reduce the computational time.

The opt_master initialization confirms the same behavior and slightly improves the results obtained by opt_basis. In particular, the column reduction increases from 66% to 69% for class R1_25_C_100 and from 48% to 63% for class R1_50_A_100; also, the number of active variables decreases.

The opt_lp initialization is less efficient than the previous ones with respect to column reduction (49% for class R1_25_C_100 and only 6% for class R1_50_A_100) and computational time, that is much higher than opt_basis and opt_master; furthermore, this initialization is not competitive with standard column generation.

To sum up, the computational time is reduced significantly when using relaxed DP; still, two-stage column generation is often slower than standard CG. However, a stable pattern is not recognized yet, since for some instances we remark time savings; furthermore, we are able to reduce substantially and steadily the number of generated columns. Therefore, we proceed with a deeper computational study of our method, enlarging the test case and performing different sensitivity analysis.

In particular, we remark a high number of CG2 iterations in all tests and this is computationally expensive. Our attempts go therefore in the direction of reducing the CG2 iterations by analyzing the best strategy for adding columns.

6.6.3 Sensitivity analysis: strategies for adding CG2 columns

In standard column generation, it is well known that adding one column per iteration is not efficient. When using a combinatorial algorithm for solving the pricing subproblem, such as dynamic programming, it is typically a *set* of negative reduced cost columns that is added to the master problem, in order to reduce the callings to the pricing subproblem, that is usually very expensive to solve in terms of computational effort.

Similar strategies must be investigated for the CG2 step of two-stage column generation, that appears to be the more expensive; the primary objective is the reduction of CG2 iterations, and thus of computational time.

For this purpose, we analyze the following strategies for adding columns in the CG2 step:

- **10ord-10arc**: it consists of adding at most 10 new orders and 10 new arcs per iteration to the partial compact formulation, among those with negative reduced cost; this was the default strategy in our implementation, and all the computational results shown so far have been obtained using this rule.
- **50ord-50arc**: it consists of adding at most 50 new orders and 50 new arcs per iteration to the partial compact formulation, among those with negative reduced cost;
- **10ord-100arc**: it consists of adding at most 10 new orders and 100 new arcs per iteration to the partial compact formulation, among those with negative reduced cost;
- **10ord-150arc**: it consists of adding at most 10 new orders and 150 new arcs per iteration to the partial compact formulation, among those with negative reduced cost;
- **50ord-150arc**: it consists of adding at most 50 new orders and 150 new arcs per iteration to the partial compact formulation, among those with negative reduced cost.

Computational analysis is now extended to more difficult instances: we include class C1_25_C_100 for 25 customers and class C1_50_A_100 for 50 customers.

Since the opt_basis initialization has shown similar performance to opt_master, from now on we will only compare the opt_master and the opt_lp initializations.

A summary of the computational results is provided by Table 6.4.

With respect to standard column generation, the complexity of classes C1_25_C_100 and C1_50_A_100 is confirmed by the increased average computational time. Furthermore, two instances cannot be solved at the root node within the time limit of one hour.

Concerning 25 customers, the strategy "$50\mathtt{ord} - 150\mathtt{arc}$" yields the highest reduction in terms of computational time and CG2 iterations; although it generates

		Stand.CG			Two-stage column generation													
					10ord-10arc			50ord-50arc			10ord-100arc			10ord-150arc			50ord-150arc		
Class	**nr**	**sol**	**col**	**t**	**sol**	**col**	**t**	**sol**	**col**	**t**	**sol**	**col**	**t**	**sol**	**col**	**t**	**sol**	**col**	**t**
opt_master																			
R1_25_C_100	12	12	3165	18	12	971	88	12	1402	25	12	648	51	12	689	58	12	1129	24
C1_25_C_100	9	8	3953	596	7	881	326	8	1223	872	7	906	823	7	805	720	7	997	610
R1_50_A_50	12	12	1789	37	12	667	28	12	681	9	12	704	16	12	703	19	12	721	12
C1_50_A_100	9	8	2910	501	8	852	30	8	1034	17	8	892	79	8	872	123	8	1082	18
opt_lp																			
R1_25_C_100	12	12	3165	18	12	1619	140	12	3074	72	12	922	84	12	898	78	12	1855	52
C1_25_C_100	9	8	3953	596	3	1683	215	2	3817	146	3	1226	146	3	1180	142	4	2646	939
R1_50_A_50	12	12	1789	37	12	1690	192	12	1989	91	12	1578	97	12	1525	123	12	1970	89
C1_50_A_100	9	8	2910	501	8	2894	268	8	3610	179	8	2438	267	8	2281	213	8	3991	102

Table 6.4: *Comparison of different strategies for adding columns in the CG2 step.*

a little more columns than strategies that add only 10 orders per iteration, the reduction with respect to standard column generation is still significant (about 65% of reduced columns when using the opt_master initialization, and about 30% with opt_lp). As a general remark, we clearly see that the number of CG2 iterations is strongly affected by the chosen strategy, varying from an average of 4 iterations for the "50ord − 150arc" strategy up to an average of 15 iterations for the "10ord − 10arc" strategy (detailed results are provided in Appendix B). As expected, the opt_master initialization performs better than the LP initialization: in addition to a longer computational time, opt_lp also generates a higher number of columns.

At first glance, our two-stage approach does not yield time savings for easy instances of class R1_25_C_100: indeed, the root node is always efficiently solved by standard CG in less than one minute for this class. On the contrary, applying two-stage column generation to the more difficult instances of class C1_25_C_100 is beneficial: we reduce the computational time for some instances, and the result is encouraging, in addition to the substantial reduction of columns that we constantly obtain with our approach.

The overall behavior for 50 customers confirms the results obtained for 25 customers: strategy "50ord − 150arc" is the best option in terms of computational time and CG2 iterations, and the number of CG2 iterations is strongly affected by the chosen strategy. Opposite to 25 customers, the time reduction for 50 customers is significant, especially when using the opt_master initialization: the average computational time is decreased already for the easier class R1_50_A_50, from 37 seconds of standard CG to 12 seconds of two-stage CG with the "50ord − 150arc" strategy. Remarkably, for class C1_50_A_100, the average computational time is reduced from 501 seconds to about 18 seconds.

Surprisingly, the "bad" LP initialization (Table B.8) also yields to substantial savings for class C1_50_A_100: the average computational time for these difficult instances decreases from 501 seconds to 102 seconds. Unfortunately, for class R1_50_A_50 we do not observe a time reduction but a slight increase.

To sum up, strategy "50ord − 150arc" results to be the best choice for two-stage column generation; in particular, it allows to significant reduce the computational time with respect to standard column generation, especially with difficult instances. However, a good guess of the optimal master solution is required for the initialization, in order to obtain the best performance in terms of time and number of columns.

According to the results of this section, we continue our computational analysis by testing only strategy "50ord − 150arc" for adding columns in the CG2 step.

6.6.4 Sensitivity analysis: increasing number of orders

In this test, we are interested in analyzing how two-stage column generation performs when the number of orders increases. We focus on the instances with 50 customers seen so far, and we compare our approach over scenarios A (150 orders), B (250 orders) and C (350 orders). A summary of the computational results is provided in

		Stand.CG			Two-stage column generation					
					opt_master			opt_lp		
Class	**nr**	**sol**	**cols**	**t**	**sol**	**cols**	**t**	**sol**	**cols**	**t**
R1_50_A_50	12	12	1789	37	12	721	12	12	1970	89
R1_50_B_50	12	11	3331	292	11	1247	269	10	2897	541
R1_50_C_50	12	10	4612	980	8	1624	949	6	3243	997
C1_50_A_50	9	8	2910	501	8	1082	18	8	3991	102
C1_50_B_50	9	6	4882	1099	6	1804	739	4	5054	888
C1_50_C_50	9	4	6443	1937	3	2320	545	1	7063	2479

Table 6.5: *Summary of results for increasing number of orders.*

Table 6.5.

Concerning class R1, expectedly, the computational time increases with the problem size, and so the number of generated columns, both for standard and two-stage CG. Time savings are constant when using the opt_master initialization for scenarios A and B, while we observe an increase of computational time for some instances of scenario C, that results in 8 instances solved within one hour instead of 10. Remarkably, two-stage column generation is able to solve the root node within the time limit for instances r108_50_B_50 and r111_50_C_50, whereas standard CG fails (cf. Appendix B). The column reduction is stable for opt_master, that generates on average 60% less columns than standard CG. The LP initialization performs differently: under scenario A, a higher number of columns is generated, whereas for scenarios B and C we notice a reduction that increases with the number of orders. As mentioned, this behavior is explained by the fact that, as soon as a higher number of order partitions is available, two-stage CG has the possibility to keep out of the formulation a higher number of orders and therefore only "good" columns are generated. As remarked in the previous section for scenario A, the opt_lp initialization does not allow for time savings on these instances, and this behavior is confirmed also for scenarios B and C; in particular, a lower number of instances can be solved within one hour with this initialization.

Computational experiments for class C1 confirm the behavior already observed for class R1: remarkably, two-stage column generation appears to be even more beneficial for the difficult instances of class C1. The time and column reduction yielded by the opt_master initialization is remarkable: even for scenarios B and C, the computational time is more than halved and the number of generated columns is decreased on average by 60%. Only for scenario C, one instance solved by standard column generation cannot be closed by opt_master within one hour. On the contrary, the opt_lp initialization presents an unstable behavior: while a time reduction is obtained for scenario A, the computational time suddenly increases for scenarios B and C, resulting in a much lower number of solved instances; furthermore, the number of generated columns is always greater than in standard column generation. We attribute this fact to the bad quality of the initialization provided by the LP relaxation.

	Variable Elimination			Two-stage column generation				
Class	$\%\lvert C_{sub}\rvert$	$\%\lvert E_{sub}\rvert$	**t**	$\%\lvert C_{sub}\rvert$	$\%\lvert C_{ina}\rvert$	$\%\lvert E_{sub}\rvert$	$\%\lvert E_{ina}\rvert$	**t**
Exact CG2 DP								
R1_25_C_100	18%	47%	364	17%	23%	2%	84%	713
R1_50_A_50	0%	33%	463	0%	14%	0%	94%	593
Relaxed CG2 DP								
R1_25_C_100	0%	21%	22	4%	11%	1%	77%	88
R1_50_A_50	0%	26%	38	0%	8%	0%	91%	28

Table 6.6: *Comparison between variable elimination (Irnich et al., 2010) and two-stage column generation with the* opt_master *initialization.*

6.6.5 Variable elimination

Our two-stage column generation framework allows for an additional feature: compact formulation variables, such as orders and arcs, can be eliminated as soon as their reduced cost exceeds the current optimality gap, that is the difference between a valid upper bound and a valid lower bound.

We adapt the method proposed by Irnich et al. (2010) for arc-flow variable elimination to order-selection variables too and we compared the performance of two-stage column generation with respect to the variable elimination method.

We remark that Irnich et al.'s (2010) approach differs from our two-stage column generation approach: while variable elimination is applied by Irnich et al. (2010) at the end of the root node in order to speed-up the overall branch-and-price algorithm in the child nodes, in two-stage CG we dynamically add compact formulation variables; however, we take advantage of the computational effort required for the computation of reduced costs in the CG2 step by including variable elimination as a "bonus" in our framework. In addition to the set of *suboptimal* variables, in two-stage column generation we also have a set of *inactive* variables that are not in the compact formulation, but that cannot be proved suboptimal due to a positive reduced cost that does not exceed the optimality gap. While the speed-up provided by Irnich et al. (2010) only concerns detected suboptimal variables, the speed-up provided by two-stage column generation originates from suboptimal and inactive variables, since both types of variables are not taken into account in the compact formulation and, more in general, in the problem resolution.

Table 6.6 provides a summary of the comparison between standard variable elimination and two-stage column generation in terms of detected suboptimal variables. For every class, the average percentage of suboptimal orders ($\%\lvert C_{sub}\rvert$), suboptimal arcs ($\%\lvert E_{sub}\rvert$), inactive orders ($\%\lvert C_{ina}\rvert$) and inactive arcs ($\%\lvert E_{ina}\rvert$), as well as the

average computational time (t) is reported.

Concerning Irnich et al.'s (2010) variable elimination, the exact DP provides better results than relaxed DP, as expected: on average, 18% of orders and 47% of arcs are proved to be suboptimal for class R1_25_C_100, while relaxed DP only detects 21% of suboptimal arcs and no orders at all. For class R1_50_A_50, the performance is almost unchanged: from 33% to 26% of suboptimal arcs detected on average, and zero orders. We notice a different behavior with different type of variables: while suboptimal arcs are easily detected, only a few orders are proved to be suboptimal. This may be due to a different behavior of the reduced costs: in fact, we remark that arc-flow variables appear in the objective function and order-selection variables do not.

Two-stage column generation cannot prove suboptimality as much as Irnich et al.'s (2010) variable elimination; however, if we consider also the inactive variables, the number of variables not taken into account throughout the solution process is higher (41% of orders and 86% of arcs for class R1_25_C_100, 14% of orders and 94% of arcs for class R1_50_A_50). The percentages are lower when using relaxed DP and/or the opt_lp initialization (cf. Appendix B).

Finally, the computational time of two-stage column generation is higher: this is an expected result, since in variable elimination method the reduced costs of compact formulation variables are computed only once, at the end of the root node, and this corresponds to only one CG2 iteration in the two-stage approach.

6.7 Application to TBAP

In this section, we discuss how the two-stage column generation framework can be applied to the Tactical Berth Allocation Problem studied in Chapters 3 and 4.

The MILP model presented in Section 3.3.4 represents our compact formulation. We apply Dantzig-Wolfe decomposition and we obtain the master problem and the pricing subproblem introduced in Section 4.1.3.

Profile-assignment variables λ_i^p associated with quay-crane profiles $p \in P_i$ for vessels $i \in N$ are particularly suited to be treated by two-stage column generation.

The basic idea is to start with a subset of feasible profiles $p \in \bar{P}_i \subset P_i$ for every vessel $i \in N$, such that a feasible solution exists. Profitable profiles $p \in \widehat{P}_i = P_i \setminus \bar{P}_i$ are dynamically added to the problem according to the extensive reduced cost of compact formulation variables λ_i^p.

The contribution of compact formulation variables λ_i^p to the master problem is computed similarly to that of order-selection variables y_c in the DSDVRPTW. Let $\tilde{v}_{r_k}$ be the reduced cost of sequence $r_k \in \Omega^k$ for berth $k \in M$ as defined in section 4.1.3. We define $\mathcal{F}_{ip}^k \subseteq \Omega^k$ as the set of all sequences that operate vessel i with quay crane profile p for berth k. The extensive reduced cost of variable λ_i^p, is estimated

as:

$$\tilde{v}_{EF}(\lambda_i^p) = \min_{k \in M} \tilde{v}^*_{r_k} = \min_{k \in M} \min_{r_k \in \mathcal{F}^k_{ip}} \tilde{v}_{r_k}, \tag{6.70}$$

that is the minimum reduced cost among any sequence that operates vessel i with quay crane profile p, over all berths $k \in M$.

The implementation of CG2 dynamic programming for the computation of extensive reduced costs is more complicated than for the DSDVRPTW, since we must take into account the specific features of the TBAP problem in terms of reformulation and pricing structure.

In particular, the following issues need to be addressed:

- the reformulation for TBAP relies on *multiple pricing*, one for every berth $k \in M$, that are solved at the CG1 step of the two-stage column generation framework; analogously, the CG2 step could be implemented by computing the extensive reduced cost for every berth ($\tilde{v}^*_{r_k}$) and take the minimum over all berths. As a consequence, we would have multiple pricing at the CG2 step too, and this would be highly inefficient, since we saw that CG2 dynamic programming is very expensive. A smarter way to compute the extensive reduced costs should be investigated, in order to minimize the additional computational effort required by two-stage column generation.

- the dynamic programming algorithm implemented for TBAP includes many acceleration techniques that have been specifically conceived for the problem (cf. section 4.1.4). This corresponds to the CG1 step in the two-stage column generation framework. Adapting this algorithm for the CG2 step would require to remove some of these techniques, such as the domination of (h, p) pairs, and as a consequence the algorithm would be further slowed down. Specific accelerating techniques and domination criteria should be studied in order to obtain an efficient implementation of CG2 dynamic programming.

The study of these issues and the actual implementation of two-stage column generation for TBAP is the objective of future research.

6.8 Conclusions

In this chapter we introduce a novel framework called two-stage column generation, specifically conceived to tackle complex problems that cannot be efficiently solved by standard column generation. We focus our attention on the relationship between compact and extensive formulation with the objective of exploiting the information provided by the DW reformulation when dealing with compact formulation variables.

In two-stage column generation, we start solving the problem on a subset of compact formulation variables, we apply Dantzig-Wolfe decomposition and we solve the

resulting master problem via standard column generation. Furthermore, compact formulation variables are also dynamically added to the formulation according the their reduced cost. In particular, we introduce the concept of extensive reduced cost, that allows to estimate the contribution of compact formulation variables to the master problem.

A formal description of the new framework is provided and an example based on the Resource Constrained Shortest Path Problem illustrates how two-stage column generation basically works when the pricing subproblem satisfies or not the integrality property.

We apply the proposed methodology to the Discrete Split Delivery Vehicle Routing Problem with Time Windows and we perform intensive computational experiments in order to validate our new framework and analyze the effects especially on very complex instances.

Computational results show that two-stage column generation significantly reduces the number of generated columns to prove optimality of the root node. Furthermore, suboptimal compact formulation variables are detected correctly and a large percentage of variables is inactive and therefore not taken into account during the solution process. However, the method is sensitive to initialization and a significant reduction of computational time is obtained only for the most complex instances that we tested. Indeed, the result is promising, since the primary objective of two-stage column generation is to deal with complex instances, and it proves to be successful within this context. The additional effort required by our sophisticated approach makes the method competitive in terms of computational time only for instances of a certain difficulty.

To conclude, the two-stage column generation framework is a promising new approach to investigate more and more complex problems. Clearly, there are many aspects that need to be further investigated by future research. The computation of extensive reduced costs for compact formulation variables still represents a major issue in terms of computational time, that could be overcome by more efficient implementations for CG2 dynamic programming or new approaches for estimating the contribution of original variables to the extensive formulation. Also, results show that the strategy chosen for adding compact formulation variables at every CG2 step and the number of CG2 iterations are strictly correlated: smart techniques are worth being investigated in order to reduce the number of CG2 iterations and thus the computational effort. We may think of heuristic strategies based on the current optimality gap, as well as problem-specific strategies that take into account the actual objects modeled by compact formulation variables. Finally, we remark that the initialization of compact formulation variables represents a crucial point for the overall efficiency of the two-stage framework, therefore good initialization strategies and their effect on the overall performance of the algorithm need to be further analyzed.

Chapter 7

Conclusion

In this chapter we summarize the main results of this dissertation and we discuss some directions and perspectives for future research.

Part I of the dissertation is devoted to container terminal management. We propose models and algorithms that are specifically conceived for this application and we discuss the advantages of integrated planning of operations. The broader objective is to provide methods that can be further generalized to solve large scale optimization problems in other domains.

In Chapter 2 we introduce the context of maritime transportation and logistics, and we provide an overview of container terminal operations. The main processes associated with the flow of containers within the terminal are described and we discuss what are the critical bottlenecks in the system. We review the state-of-the-art in container terminal management, with a particular focus on operations research and optimization techniques. We discuss the current research challenges and promising research directions are suggested.

In Chapter 3 we propose a new model for the integrated planning of berth allocation and quay crane assignment, addressing the problem at the tactical decision level. From a modeling point of view, we overcome the limiting assumptions of existing approaches and we provide a more realistic problem definition that includes operational constraints, unwritten rules and best practices. We propose a heuristic approach that is able to produce good-quality solutions with the minimum computational effort: computational results confirm that our method clearly outperforms commercial solvers.

In Chapter 4 the contribution is twofold. The first part of the chapter is mainly algorithmic: we propose an exact branch-and-price algorithm for Tactical Berth Allocation Problem and we present several accelerating techniques for the pricing and the master problem devised for this specific problem. In particular, some of these techniques can be easily generalized for other branch-and-price schemes. The proposed method outperforms commercial solvers and computational results are promising. Clearly, the problem is complex and difficult to solve: this motivates the design of

new solution approaches for difficult large scale optimization problems, such as TBAP, addressed in the second part of the dissertation. The chapter also contributes from the modeling point of view: we provide an experimental comparison between the hierarchical approach, that solves the berth allocation and the quay crane assignment sequentially, and our integrated modeling approach, and we show the added value of integrated planning in terms of cost reduction and efficient use of resources.

Part II of the dissertation is devoted to the design of new concepts and methods for difficult large scale optimization problem mainly based on Dantzig-Wolfe decomposition and column generation. We propose a novel framework, called two-stage column generation, that represents a major contribution of our work. In particular, we study a class of split delivery vehicle routing problems that generalizes some interesting features of TBAP that are relevant also to other applications such as transportation, logistics and telecommunication.

In Chapter 5 we introduce the Discrete Split Delivery Vehicle Routing Problem with Time Windows and we study some properties of the new problem. The proposed arc-flow model is reformulated via Dantzig-Wolfe decomposition and solved via column generation. We implement an exact branch-and-price algorithm that clearly outperforms commercial solvers and we provide an interesting comparison between constant service time vs quantity-dependent service time, in order to support the importance of the new modeling feature; in particular, we show that additional complexity comes with potential savings.

In Chapter 6 we propose the Two-stage column generation framework, a new approach that extends the well known concepts in column generation by exploring the relationship between compact and extensive formulation. We aim to exploit the information provided by the DW reformulation when dealing with compact formulation variables. The broader objective is to design a framework able to tackle complex problems that cannot be efficiently solved by standard column generation. We provide a formal description of the new approach and we apply the proposed methodology to the Discrete Split Delivery Vehicle Routing Problem with Time Windows. The results are promising: we can significantly reduce the number of generated columns and, for the most complex instances that we tested, also the computational time.

Research perspectives. The presented models and methodologies show promising results for container terminal management and, more in general, for large scale optimization problems.

In particular, the proposed Two-stage column generation framework is a general approach that is particularly suited for those problems where the large number of variables in the compact formulation directly affects the pricing problem and its efficiency; we remark that this structure is common to different real-world applications in transportation, telecommunication and logistics.

We believe that Two-stage column generation is a promising new approach to investigate more and more complex problems. However, many aspects are worth

being investigated in future research. The computation of extensive reduced costs for compact formulation variables still represents a major issue in terms of computational time, that could be overcome by more efficient implementations or new approaches for estimating the contribution of original variables to the extensive formulation. Also, as a next step, we may improve the overall performance of the framework by designing smart strategies for adding compact formulation variables and by defining good initializations. We expect future contributions to this research stream to have a relevant impact on the optimization of difficult large scale optimization problems.

With respect to the specific application of container terminals, the integrated planning of operations certainly represents the current research challenge in the field. The resulting problems are typically complex, but also yield to significant improvements in terms of efficiency, productivity and cost reduction, as they allow for a better control of terminal resources and operations. Therefore, further research in this direction, both in terms of models and specialized solution algorithms, may have a relevant impact in the practice of container terminal management.

Finally, a challenging research stream is represented by the analysis of congestion in container terminal management. Indeed, traffic and congestion issues are becoming more and more relevant, especially because of the volume increase in container traffic. Although congestion is often disregarded in the planning, operations are usually slowed down because of overloaded areas, such as the yard, and congestion rapidly spreads to the whole system. We believe that an analytical description of congestion may highly contribute to design specific solutions and therefore improve terminal efficiency, and we expect this to become a major research topic in container terminal management in the near future.

Appendix A

Discrete Split Delivery VRPTW: computational results

Branch-and-price for the DSDVRPTW In Section 5.7.2, we have reported a summary of the computational results that were obtained by the branch-and-price algorithm for the DSDVRPTW. In this appendix, we present the corresponding detailed results for each instance that was solved to optimality within one hour of computational time. The results are given in Tables A.1 to A.5. For each instance, we provide the capacity (Q), the name (id), the value of the optimal integer solution (z_{IP}), the number of vehicles (veh) and the computational time in seconds (t). The three DSDVRPTW scenarios A, B, C and compared to the unsplittable VRPTW scenario O: figures highlighted in bold denote savings due to split deliveries. Instances that are not feasible for the unsplittable case because of insufficient capacity are denoted by *"*$Q < d$*"*. Instances not solved at optimality within one hour of computational time are denoted by *"*x*"*. A global discussion about these results can be found in Section 5.7.2.

Delivery-dependent service times vs. constant service times In Section 5.7.3, we have reported a summary of the computational results that were obtained by the branch-and-price algorithm when the assumption of constant service time is made for the DSDVRPTW. In this appendix, we present the corresponding detailed results for each instance that was solved to optimality within one hour of computational time. Only instances for which considering delivery dependent service times has produced an improvement of the solution are reported. The results are given in Table A.6. For each instance, we provide the name (id), the number of customers (n), the vehicle's capacity (Q) and the considered scenario ($scen$). Delivery-dependent service time instances are denoted by DDST while constant service time is denoted by CST. For both DDST and CST, we provide the value of the optimal integer solution (z_{IP}), the number of vehicles (veh) and the computational time in seconds (t). A global discussion about these results can be found in Section 5.7.3.

		scenario O			**scenario A**			**scenario B**			**scenario C**		
Q	id	z_{IP}	veh	t	z_{IP}	veh	t	z_{IP}	veh	t	z_{IP}	veh	t
30	r101	795.6	13	0	**795.1**	13	1	**782.5**	13	3	**782.5**	13	11
	r102	789.1	13	0	**772.3**	13	4	**765.9**	**12**	161	**761.2**	**12**	291
	r103	759.6	12	0	759.6	12	19	**751.7**	12	176	**745.3**	12	70
	r104	759.6	12	0	759.6	12	33	**747.0**	12	32	**745.3**	12	140
	r105	775.7	12	0	**775.3**	12	3	**773.2**	12	47	**773.2**	12	558
	r106	772.6	13	0	**763.7**	**12**	4	**756.6**	**12**	50	**753.4**	**12**	115
	r107	748.5	12	0	748.5	12	3	**744.1**	12	57	x		
	r108	748.5	12	0	748.5	12	4	**744.1**	12	100	x		
	r109	754.6	12	0	754.6	12	1	**750.2**	12	20	**750.2**	12	1041
	r110	748.5	12	0	748.5	12	4	**744.1**	12	37	**744.1**	12	1498
	r111	754.6	12	0	754.6	12	2	**750.2**	12	102	x		
	r112	748.5	12	0	748.5	12	5	**744.1**	12	118	x		
50	r101	635.0	9	0	**631.5**	**8**	0	**631.5**	**8**	1	**631.5**	**8**	1
	r102	580.7	8	0	580.7	8	7	580.7	8	35	580.7	8	221
	r103	534.3	7	0	534.3	7	3	534.3	7	65	534.3	7	333
	r104	527.3	7	0	527.3	7	7	527.3	7	76	527.3	7	437
	r105	596.1	8	0	**588.9**	8	1	**585.4**	8	4	**585.4**	8	13
	r106	543.3	7	0	**542.5**	7	4	**542.3**	7	52	**542.3**	7	233
	r107	527.7	7	0	527.7	7	14	527.7	7	187	527.7	7	1309
	r108	521.6	7	0	521.6	7	16	521.6	7	185	521.6	7	2175
	r109	524.6	7	0	524.6	7	1	524.6	7	5	524.6	7	11
	r110	536.7	7	0	**529.1**	7	3	**526.0**	7	17	**526.0**	7	119
	r111	521.6	7	0	521.6	7	7	521.6	7	45	521.6	7	178
	r112	515.8	7	0	515.8	7	8	515.8	7	46	515.8	7	135
100	r101	617.1	8	0	617.1	8	0	617.1	8	1	617.1	8	1
	r102	547.1	7	0	547.1	7	1	547.1	7	7	547.1	7	15
	r103	454.6	5	0	454.6	5	2	454.6	5	8	454.6	5	14
	r104	416.9	4	0	416.9	4	5	416.9	4	14	416.9	4	58
	r105	530.5	6	0	530.5	6	1	530.5	6	3	530.5	6	5
	r106	465.4	5	0	465.4	5	7	465.4	5	55	465.4	5	201
	r107	428.4	4	0	428.4	4	7	428.4	4	32	428.4	4	87
	r108	403.2	4	0	403.2	4	10	403.2	4	28	403.2	4	111
	r109	441.3	5	0	441.3	5	2	441.3	5	7	441.3	5	12
	r110	444.1	5	0	444.1	5	13	444.1	5	96	444.1	5	229
	r111	428.8	4	0	428.8	4	6	428.8	4	30	428.8	4	101
	r112	401.7	4	1	**401.3**	4	59	**401.3**	4	209	**401.3**	4	519

Table A.1: *Optimal solutions for class R1,* $\mathfrak{n} = 25$ *customers.*

Q	id	scenario O z_{IP}	veh	t	scenario A z_{IP}	veh	t	scenario B z_{IP}	veh	t	scenario C z_{IP}	veh	t
30	c101	Q < d			825.3	16	532	x			x		
	c105	Q < d			825.7	16	985	x			x		
	c106	Q < d			826.4	16	630	x			x		
	c107	Q < d			825.7	16	2285	x			x		
50	c101	516.9	10	0	**516.8**	10	5	**516.8**	10	1242	x		
	c102	516.6	10	0	**516.5**	10	29	x			x		
	c103	516.6	10	0	**516.5**	10	56	x			x		
	c104	516.6	10	0	**516.4**	10	142	x			x		
	c105	516.9	10	0	**516.8**	10	9	**516.8**	10	2030	x		
	c106	516.9	10	0	**516.8**	10	7	**516.8**	10	1721	x		
	c107	516.9	10	0	**516.8**	10	17	**516.8**	10	3555	x		
	c108	516.8	10	0	**516.7**	10	29	x			x		
	c109	516.8	10	0	**515.9**	10	42	x			x		
100	c101	291.9	5	0	291.9	5	17	291.9	5	175	291.9	5	1858
	c102	291.9	5	10	291.9	5	1010	x			x		
	c105	291.9	5	1	291.9	5	47	291.9	5	687	x		
	c106	291.9	5	1	291.9	5	24	291.9	5	231	291.9	5	1894
	c107	291.9	5	1	291.9	5	86	291.9	5	1726	x		
	c108	291.9	5	2	291.9	5	530	x			x		
	c109	289.5	5	15	289.5	5	3226	x			x		

Table A.2: *Optimal solutions for class C1,* $\mathfrak{n} = 25$ *customers.*

Q	id	scenario O z_{IP}	veh	t	scenario A z_{IP}	veh	t	scenario B z_{IP}	veh	t	scenario C z_{IP}	veh	t
30	rc101	Q < d			1438.0	18	453	x			x		
	rc106	Q < d			1438.0	18	3523	x			x		
100	rc101	534.3	6	0	534.3	6	1	534.3	6	6	534.3	6	19
	rc102	523.7	6	0	523.7	6	2	523.7	6	11	523.7	6	34
	rc103	514.7	6	0	**513.7**	6	3	**513.7**	6	11	**513.7**	6	54
	rc104	506.7	6	0	506.7	6	3	506.7	6	18	506.7	6	34
	rc105	527.5	6	0	527.5	6	3	527.5	6	6	527.5	6	32
	rc106	515.6	6	0	515.6	6	1	515.6	6	4	515.6	6	12
	rc107	505.7	6	0	505.7	6	3	505.7	6	13	505.7	6	39
	rc108	505.7	6	0	505.7	6	4	505.7	6	16	505.7	6	56

Table A.3: *Optimal solutions for class RC1,* $\mathfrak{n} = 25$ *customers.*

		scenario O			**scenario A**			**scenario B**			**scenario C**		
Q	id	z_{IP}	veh	t	z_{IP}	veh	t	z_{IP}	veh	t	z_{IP}	veh	t
30	r101	Q < d			1664.6	26	1010	x			x		
50	r101	1222.0	16	1	**1211.1**	16	127	**1198.7**	**15**	385	x		
	r102	1134.9	16	2	**1125.1**	16	3404	x			x		
	r105	1166.3	16	17	**1148.5**	16	1185	x			x		
100	r101	1044.0	12	0	1044.0	12	9	**1040.6**	12	22	**1040.6**	12	54
	r102	913.2	11	1	913.2	11	58	**911.9**	11	311	**911.9**	11	1016
	r105	918.2	9	7	918.2	9	3038	x			x		

Table A.4: *Optimal solutions for class R1,* $\mathfrak{n} = 50$ *customers.*

		scenario O			**scenario A**			**scenario B**			**scenario C**		
Q	id	z_{IP}	veh	t	z_{IP}	veh	t	z_{IP}	veh	t	z_{IP}	veh	t
50	rc101	1713.2	20	0	**1708.9**	20	13	**1708.3**	20	594	x		
	rc102	1704.3	20	0	**1700.5**	20	62	**1700.5**	20	1938	x		
	rc103	1703.4	20	1	**1696.8**	20	37	**1696.8**	20	427	x		
	rc104	1702.2	20	1	**1696.7**	20	54	**1696.7**	20	677	x		
	rc105	1703.9	20	0	**1700.1**	20	73	**1700.1**	20	1132	x		
	rc107	1704.1	20	1	**1698.6**	20	58	x			x		
	rc108	1702.2	20	2	**1696.7**	20	83	**1696.7**	20	645	x		
100	rc101	994.6	10	2	**993.8**	10	257	**984.4**	10	524	x		
	rc102	961.0	10	1	**960.2**	10	2657	x			x		
	rc103	936.2	10	4	936.2	10	837	x			x		
	rc104	915.9	10	4	915.9	10	198	915.9	10	2140	x		
	rc105	957.4	10	2	957.4	10	82	957.4	10	536	957.4	10	2940
	rc106	937.0	10	1	937.0	10	58	937.0	10	742	x		
	rc107	915.1	10	1	915.1	10	33	915.1	10	515	915.1	10	2064
	rc108	911.9	10	3	911.9	10	110	911.9	10	398	911.9	10	3491

Table A.5: *Optimal solutions for class RC1,* $\mathfrak{n} = 50$ *customers.*

				DDST			**CST**		
id	n	Q	scen	z_{IP}	veh	t	z_{IP}	veh	t
r105	25	50	C	585.4	8	12.91	588.0	8	1.50
r110	25	50	C	526.0	7	118.96	529.1	7	9.70
rc103	25	100	A	513.7	6	2.94	514.7	6	1.18
rc103	25	100	B	513.7	6	11.43	514.7	6	3.72
r101	50	30	A	1664.6	26	1009.71	1676.7	26	64.39
r101	50	50	A	1211.1	16	127.27	1211.8	16	9.15
r102	50	50	A	1125.1	16	3403.82	1127.7	16	121.27
rc107	50	50	A	1698.6	20	57.84	1699.0	20	5.07
rc101	50	100	B	984.4	10	524.38	990.5	10	101.87
r101	50	50	B	1198.7	15	384.96	1207.7	15	39.01
r102	50	100	B	911.9	11	311.12	913.2	11	19.14
r101	50	50	C	1198.7	15	384.96	1203.3	15	64.83
r101	50	100	C	1040.6	12	54.21	1044.6	12	4.59
r102	50	100	C	911.9	11	1016.34	913.2	11	53.24

Table A.6: *Constant service time vs delivery dependent service time (optimal solutions).*

Appendix B

Two-stage column generation: computational results

Two-stage column generation for the DSDVRPTW In Section 6.6, we have reported a summary of the computational results that were obtained by standard and two-stage column generation for the DSDVRPTW. In this appendix, we present the corresponding detailed results for each instance that was solved to optimality within one hour of computational time. The results are given in Tables B.1 to B.12.

Table B.1 provides a comparison between standard column generation and two-stage column generation at the root node. For standard column generation we report the number of generated columns (cols) and the total computational time in seconds (t). For the two-stage framework, three initializations (opt_basis, opt_master, opt_lp) were tested and we report for each one of them the number of iterations of CG2 (it), the total computational time (t_{tot}) and the amount of time spent in the CG2 step (t_{CG2}). The reduction of columns with respect to standard column generation is denoted by $\%_{cols}$.

A comparison between exact and relaxed CG2 dynamic programming for two-stage column generation with different initializations is provided in Tables B.2 (opt_basis), B.3 (opt_master) and B.4 (opt_lp). With respect to the previous table, we report additional information on the percentage of orders (%ord) and arcs (%arc) that are active at the end of the root node.

Tables B.5, B.6, B.7 and B.8 provide a comparison for different strategies for adding CG2 columns.

Tables B.9 and B.10 provide a sensitivity analysis with respect to the increasing number of orders. The reduction of number of columns generated the end of the root node is denoted by $\%_{c}$ols.

Finally, variable elimination is compared to two-stage column generation in Tables B.11 and B.12. For every instance, the total number of orders ($|C|$) and arcs ($|E|$), the number of suboptimal orders ($|C_{sub}|$ and suboptimal arcs ($|E_{sub}|$), the number of inactive orders ($|C_{ina}|$) and inactive arcs ($|E_{ina}|$), as well as the total computational time (t_{tot}) is reported.

Instance	Stand.CG cols	t	2stage - init:opt_basis cols	%cols	it	ttot	tCG2	2stage - init:opt_master cols	%cols	it	ttot	tCG2	2stage - init:opt_LP cols	%cols	it	ttot	tCG2
R101_25_C_100	990	1	616	-38%	8	10	9	425	-57%	9	10	10	531	-46%	9	20	20
R102_25_C_100	2567	5	945	-63%	9	109	108	825	-68%	10	350	350	x		x	x	x
R103_25_C_100	3629	13	949	-74%	5	2381	2380	1078	-70%	11	1356	1355	x		x	x	x
R104_25_C_100	4021	33	x		x	x	x	x		x	x	x	x		x	x	x
R105_25_C_100	1475	5	780	-47%	6	44	43	653	-56%	10	54	53	846	-43%	11	250	249
R106_25_C_100	3656	14	991	-73%	8	2456	2454	665	-82%	9	322	322	x		x	x	x
R107_25_C_100	4367	27	x		x	x	x	x		x	x	x	x		x	x	x
R108_25_C_100	4434	34	x		x	x	x	x		x	x	x	x		x	x	x
R109_25_C_100	2916	16	975	-67%	7	161	160	1136	-61%	10	209	207	x		x	x	x
R110_25_C_100	2546	24	945	-63%	5	2890	2888	645	-75%	9	1966	1964	x		x	x	x
R111_25_C_100	3754	23	x		x	x	x	x		x	x	x	x		x	x	x
R112_25_C_100	3630	27	1346	-63%	6	3041	3039	798	-78%	10	1433	1432	x		x	x	x
R101_50_A_50	1035	2	859	-17%	6	16	15	458	-56%	6	14	14	1143	10%	16	185	184
R102_50_A_50	1746	9	1027	-41%	7	2421	2419	x		x	x	x	x		x	x	x
R103_50_A_50	1975	22	x		x	x	x	x		x	x	x	x		x	x	x
R104_50_A_50	2287	68	x		x	x	x	x		x	x	x	x		x	x	x
R105_50_A_50	1201	6	774	-36%	7	65	64	546	-55%	6	69	68	x		x	x	x
R106_50_A_50	1930	16	x		x	x	x	x		x	x	x	x		x	x	x
R107_50_A_50	1995	33	x		x	x	x	x		x	x	x	x		x	x	x
R108_50_A_50	2280	83	x		x	x	x	x		x	x	x	x		x	x	x
R109_50_A_50	1392	12	x		x	x	x	570	-59%	7	1696	1695	x		x	x	x
R110_50_A_50	1475	29	x		x	x	x	x		x	x	x	x		x	x	x
R111_50_A_50	1950	82	x		x	x	x	x		x	x	x	x		x	x	x
R112_50_A_50	2205	77	x		x	x	x	x		x	x	x	x		x	x	x

Table B.1: *Standard column generation vs Two stage column generation: exact dynamic programming and 3 initializations.*

Instance	Stand.CG		Exact 2stage - init:opt_basis						Relaxed 2stage - init:opt_basis					
	cols	t	cols	it	%ord	%arc	t_{tot}	t_{CG2}	cols	it	%ord	%arc	t_{tot}	t_{CG2}
R101_25_C_100	990	1	616	8	55%	6%	10	9	504	6	51%	6%	0	0
R102_25_C_100	2567	5	945	9	73%	16%	109	108	798	7	76%	15%	5	5
R103_25_C_100	3629	13	949	5	42%	9%	2381	2380	958	12	91%	28%	66	65
R104_25_C_100	4021	33	x	x			x	x	1236	10	91%	25%	44	42
R105_25_C_100	1475	5	780	6	46%	8%	44	43	671	9	83%	16%	2	2
R106_25_C_100	3656	14	991	8	77%	17%	2456	2454	1192	14	93%	29%	33	31
R107_25_C_100	4367	27	x	x			x	x	1593	23	97%	47%	410	401
R108_25_C_100	4434	34	x	x			x	x	1384	9	92%	23%	84	70
R109_25_C_100	2916	16	975	7	49%	10%	161	160	750	12	98%	26%	17	16
R110_25_C_100	2546	24	945	5	77%	18%	2890	2888	974	12	95%	27%	63	59
R111_25_C_100	3754	23	x	x			x	x	1410	11	95%	28%	62	59
R112_25_C_100	3630	27	1346	6	74%	16%	3041	3039	1265	10	96%	24%	78	75
R101_50_A_50	1035	2	859	6	97%	6%	16	15	733	5	97%	6%	1	0
R102_50_A_50	1746	9	1027	7	97%	7%	2421	2419	920	7	97%	7%	4	3
R103_50_A_50	1975	22	x	x	x	x	x	x	996	19	97%	12%	31	25
R104_50_A_50	2287	68	x	x	x	x	x	x	1078	17	97%	11%	47	40
R105_50_A_50	1201	6	774	7	95%	6%	65	64	815	8	99%	6%	2	1
R106_50_A_50	1930	16	x	x	x	x	x	x	764	10	96%	7%	6	5
R107_50_A_50	1995	33	x	x	x	x	x	x	1066	18	99%	11%	32	27
R108_50_A_50	2280	83	x	x	x	x	x	x	1069	20	98%	12%	56	46
R109_50_A_50	1392	12	x	x	x	x	x	x	823	10	99%	8%	7	5
R110_50_A_50	1475	29	x	x	x	x	x	x	915	12	99%	9%	18	15
R111_50_A_50	1950	82	x	x	x	x	x	x	1052	20	98%	12%	44	33
R112_50_A_50	2205	77	x	x	x	x	x	x	946	16	97%	11%	45	38

Table B.2: *Exact 2stage column generation vs Relaxed 2stage column generation: initialization with opt basis.*

	Stand.CG		Exact 2stage - init:opt_master						Relaxed 2stage - init:opt_master					
Instance	**cols**	**t**	**cols**	**it**	**%ord**	**%arc**	$\mathbf{t_{tot}}$	$\mathbf{t_{CG2}}$	**cols**	**it**	**%ord**	**%arc**	$\mathbf{t_{tot}}$	$\mathbf{t_{CG2}}$
R101_25_C_100	990	1	425	9	48%	6%	10	10	343	10	48%	6%	1	0
R102_25_C_100	2567	5	825	10	69%	16%	350	350	653	12	67%	16%	7	6
R103_25_C_100	3629	13	1078	11	50%	10%	1356	1355	1336	14	85%	20%	39	37
R104_25_C_100	4021	33	x	x	x	x	x	x	993	14	87%	23%	100	97
R105_25_C_100	1475	5	653	10	51%	8%	54	53	767	14	83%	17%	4	3
R106_25_C_100	3656	14	665	9	63%	17%	322	322	785	14	86%	22%	19	18
R107_25_C_100	4367	27	x	x	x	x	x	x	923	18	95%	33%	166	160
R108_25_C_100	4434	34	x	x	x	x	x	x	1323	15	92%	24%	317	313
R109_25_C_100	2916	16	1136	10	67%	16%	209	207	1638	19	98%	32%	43	37
R110_25_C_100	2546	24	645	9	62%	16%	1966	1964	707	15	94%	27%	144	140
R111_25_C_100	3754	23	x	x	x	x	x	x	1401	19	96%	32%	97	93
R112_25_C_100	3630	27	798	10	66%	17%	1433	1432	778	14	89%	24%	115	113
R101_50_A_50	1035	2	458	6	85%	5%	14	14	458	6	85%	5%	1	0
R102_50_A_50	1746	9	x	x	x	x	x	x	607	9	88%	6%	4	4
R103_50_A_50	1975	22	x	x	x	x	x	x	655	22	93%	12%	38	33
R104_50_A_50	2287	68	x	x	x	x	x	x	793	22	96%	12%	75	68
R105_50_A_50	1201	6	546	6	85%	5%	69	68	580	8	89%	6%	2	1
R106_50_A_50	1930	16	x	x	x	x	x	x	666	10	89%	7%	7	6
R107_50_A_50	1995	33	x	x	x	x	x	x	738	18	95%	11%	30	26
R108_50_A_50	2280	83	x	x	x	x	x	x	784	19	97%	12%	59	51
R109_50_A_50	1392	12	570	7	88%	6%	1696	1695	576	12	92%	7%	6	6
R110_50_A_50	1475	29	x	x	x	x	x	x	627	12	94%	8%	15	14
R111_50_A_50	1950	82	x	x	x	x	x	x	810	24	96%	12%	53	44
R112_50_A_50	2205	77	x	x	x	x	x	x	704	17	95%	11%	42	35

Table B.3: *Exact 2stage column generation vs Relaxed 2stage column generation: initialization with opt master.*

Instance	Stand.CG		Exact 2stage - init:opt_lp						Relaxed 2stage - init:opt_lp					
	cols	t	cols	it	%ord	%arc	t_{tot}	t_{CG2}	cols	it	%ord	%arc	t_{tot}	t_{CG2}
R101_25_C_100	990	1	531	9	43%	6%	20	20	575	10	50%	7%	1	1
R102_25_C_100	2567	5	x	x	x	x	x	x	1009	12	78%	19%	8	8
R103_25_C_100	3629	13	x	x	x	x	x	x	1642	14	87%	23%	69	66
R104_25_C_100	4021	33	x	x	x	x	x	x	2075	16	95%	27%	283	276
R105_25_C_100	1475	5	846	11	46%	8%	250	249	828	13	84%	17%	4	3
R106_25_C_100	3656	14	x	x	x	x	x	x	1791	14	85%	21%	39	36
R107_25_C_100	4367	27	x	x	x	x	x	x	2300	20	94%	34%	296	289
R108_25_C_100	4434	34	x	x	x	x	x	x	2082	16	95%	27%	488	482
R109_25_C_100	2916	16	x	x	x	x	x	x	1859	21	100%	34%	34	26
R110_25_C_100	2546	24	x	x	x	x	x	x	1433	15	95%	25%	96	89
R111_25_C_100	3754	23	x	x	x	x	x	x	2116	16	99%	28%	151	143
R112_25_C_100	3630	27	x	x	x	x	x	x	1719	16	95%	26%	211	205
R101_50_A_50	1035	2	1143	16	100%	7%	185	184	1143	16	100%	7%	2	1
R102_50_A_50	1746	9	x	x	x	x	x	x	1646	21	100%	8%	17	14
R103_50_A_50	1975	22	x	x	x	x	x	x	1701	36	100%	15%	93	83
R104_50_A_50	2287	68	x	x	x	x	x	x	2460	33	100%	14%	604	576
R105_50_A_50	1201	6	x	x	x	x	x	x	1181	18	100%	7%	6	4
R106_50_A_50	1930	16	x	x	x	x	x	x	1442	21	100%	9%	35	30
R107_50_A_50	1995	33	x	x	x	x	x	x	1867	32	100%	13%	149	136
R108_50_A_50	2280	83	x	x	x	x	x	x	2393	33	100%	14%	834	801
R109_50_A_50	1392	12	x	x	x	x	x	x	1335	23	100%	10%	22	18
R110_50_A_50	1475	29	x	x	x	x	x	x	1656	24	100%	10%	87	78
R111_50_A_50	1950	82	x	x	x	x	x	x	1674	30	100%	12%	102	88
R112_50_A_50	2205	77	x	x	x	x	x	x	1778	31	100%	13%	356	335

Table B.4: *Exact 2stage column generation vs Relaxed 2stage column generation: initialization with opt lp.*

	Stand.CG		Relaxed 2stage - init:opt_master													
			10ord-10arc			50ord-50arc			10ord-100arc			10ord-150arc			50ord-150arc		
Instance	**cols**	**t**	**cols**	**it**	**t**	**cols**	**it**	**t**	**cols**	**it**	**t**	**cols**	**it**	**t**	**cols**	**it**	**t**
R101_25_C_100	990	1	343	10	1	522	5	1	401	9	1	371	8	0	593	4	1
R102_25_C_100	2567	5	653	12	7	791	4	3	417	8	3	415	8	3	699	4	2
R103_25_C_100	3629	13	1336	14	39	2068	6	22	801	14	26	761	12	29	1560	4	25
R104_25_C_100	4021	33	993	14	100	1741	5	42	731	12	50	775	11	83	1330	5	50
R105_25_C_100	1475	5	767	14	4	1093	5	2	574	11	3	614	10	3	1101	5	2
R106_25_C_100	3656	14	785	14	19	977	5	9	547	13	17	605	14	29	873	5	17
R107_25_C_100	4367	27	923	18	166	1489	5	49	723	15	105	671	14	96	1150	5	44
R108_25_C_100	4434	34	1323	15	317	1575	4	30	844	15	105	930	15	147	1297	4	43
R109_25_C_100	2916	16	1638	19	43	1971	6	17	686	16	36	570	16	23	1214	4	11
R110_25_C_100	2546	24	707	15	144	1332	5	44	604	14	84	651	15	72	1121	4	29
R111_25_C_100	3754	23	1401	19	97	1886	6	34	780	15	92	991	16	92	1346	4	28
R112_25_C_100	3630	27	778	14	115	1373	5	49	664	15	86	913	16	118	1262	4	34
C101_25_C_100	3050	34	939	11	18	1265	5	22	972	12	32	717	10	18	889	4	20
C102_25_C_100	6062	149	997	14	567	1402	7	682	995	14	1579	906	14	847	1438	4	551
C103_25_C_100	6787	2412	x	x	x	x		x	x	x	x	x	x	x	x	x	x
C104_25_C_100	x	x	x	x	x	1441	7	3440	x	x	x	x	x	x	x	x	x
C105_25_C_100	3420	454	807	14	85	1716	5	131	890	14	155	748	14	127	1132	4	57
C106_25_C_100	3222	25	760	14	30	1019	4	17	872	13	35	773	13	44	929	4	15
C107_25_C_100	2979	93	856	14	192	1057	5	142	903	15	884	808	14	245	578	4	289
C108_25_C_100	3050	142	845	15	312	978	6	440	876	15	766	841	15	682	867	4	307
C109_25_C_100	3055	1456	965	17	1081	908	7	2101	836	15	2311	841	15	3076	1149	4	3031

Table B.5: *Sensitivity analysis with respect to the number of added cols: 25 customers, initialization with opt master.*

	Stand.CG		Relaxed 2stage - init:opt_lp													
			10ord-10arc			50ord-50arc			10ord-100arc			10ord-150arc			50ord-150arc		
Instance	**cols**	**t**	**cols**	**it**	**t**	**cols**	**it**	**t**	**cols**	**it**	**t**	**cols**	**it**	**t**	**cols**	**it**	**t**
R101_25_C_100	990	1	575	10	1	712	4	1	370	7	0	387	10	1	539	5	1
R102_25_C_100	2567	5	1009	12	8	1598	6	5	899	10	5	766	10	4	1576	5	4
R103_25_C_100	3629	13	1642	14	69	3742	6	43	856	13	45	724	13	80	1567	5	38
R104_25_C_100	4021	33	2075	16	283	4495	6	99	1191	14	110	1213	15	84	3883	5	73
R105_25_C_100	1475	5	828	13	4	1333	5	3	681	12	4	648	12	4	1091	5	4
R106_25_C_100	3656	14	1791	14	39	3291	6	26	858	13	19	787	15	31	1701	4	13
R107_25_C_100	4367	27	2300	20	296	3743	6	105	1260	15	218	1090	15	171	2626	4	75
R108_25_C_100	4434	34	2082	16	488	5136	6	319	1317	15	245	1461	15	230	1372	4	185
R109_25_C_100	2916	16	1859	21	34	2584	6	12	707	17	16	706	16	13	1391	5	8
R110_25_C_100	2546	24	1433	15	96	2839	6	56	893	15	55	853	14	81	1513	5	31
R111_25_C_100	3754	23	2116	16	151	3607	6	60	1077	16	139	1134	15	104	2856	5	54
R112_25_C_100	3630	27	1719	16	211	3805	6	130	959	15	151	1007	15	139	2147	4	140
C101_25_C_100	3050	34	1870	14	100	3218	5	65	1195	14	103	1116	13	56	2523	4	46
C102_25_C_100	6062	149	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
C103_25_C_100	6787	2412	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
C104_25_C_100	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
C105_25_C_100	3420	454	1625	16	421	x	x	x	1166	15	201	1133	14	259	2316	4	210
C106_25_C_100	3222	25	1554	14	125	4415	7	226	1316	14	134	1291	13	111	2981	4	102
C107_25_C_100	2979	93	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
C108_25_C_100	3050	142	x	x	x	x	x	x	x	x	x	x	x	x	2762	4	3400
C109_25_C_100	3055	1456	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

Table B.6: *Sensitivity analysis with respect to the number of added cols: 25 customers, initialization with opt lp.*

	Stand.CG		Relaxed 2stage - init:opt_master													
			10ord-10arc			50ord-50arc			10ord-100arc			10ord-150arc			50ord-150arc		
Instance	**cols**	**t**	**cols**	**it**	**t**	**cols**	**it**	**t**	**cols**	**it**	**t**	**cols**	**it**	**t**	**cols**	**it**	**t**
R101_50_A_50	1035	2	458	6	1	502	4	1	469	5	1	462	5	1	479	4	1
R102_50_A_50	1746	9	607	9	4	693	4	2	699	6	4	716	6	5	663	4	3
R103_50_A_50	1975	22	655	22	38	771	6	10	759	6	14	707	6	12	753	4	9
R104_50_A_50	2287	68	793	22	75	721	6	20	761	6	32	779	6	31	771	5	24
R105_50_A_50	1201	6	580	8	2	565	5	1	561	7	2	592	5	2	599	4	1
R106_50_A_50	1930	16	666	10	7	652	4	3	655	6	6	649	7	7	676	5	6
R107_50_A_50	1995	33	738	18	30	706	6	10	728	6	13	736	6	18	759	3	9
R108_50_A_50	2280	83	784	19	59	717	5	19	829	6	38	865	6	66	874	3	29
R109_50_A_50	1392	12	576	12	6	615	5	3	611	6	5	573	6	5	581	5	4
R110_50_A_50	1475	29	627	12	15	710	5	8	770	6	11	743	7	15	753	4	11
R111_50_A_50	1950	82	810	24	53	844	7	18	923	9	36	870	8	39	976	5	23
R112_50_A_50	2205	77	704	17	42	670	6	17	682	6	28	746	6	28	763	4	23
C101_50_A_100	2190	11	929	10	4	889	4	3	845	8	8	787	8	10	1000	4	5
C102_50_A_100	4655	283	909	19	25	1161	6	14	1055	9	16	1020	10	81	1381	5	19
C103_50_A_100	4924	3250	796	18	55	1288	6	22	1015	10	286	945	10	370	1359	4	21
C104_50_A_100	x	x	n/a			n/a			n/a			n/a			n/a		
C105_50_A_100	2232	19	838	12	8	889	6	5	899	9	23	815	10	20	922	4	7
C106_50_A_100	2216	17	824	12	6	969	6	5	748	9	26	852	9	19	868	4	5
C107_50_A_100	2276	26	927	17	15	1344	8	14	885	10	32	934	10	37	1256	6	15
C108_50_A_100	2312	84	782	20	39	860	6	16	913	9	68	783	9	127	894	4	18
C109_50_A_100	2475	316	814	24	85	870	7	54	779	10	173	839	10	320	977	5	52

Table B.7: *Sensitivity analysis with respect to the number of added cols: 50 customers, initialization with opt master.*

	Stand.CG		Relaxed 2stage - init:opt_lp													
			10ord-10arc			50ord-50arc			10ord-100arc			10ord-150arc			50ord-150arc		
Instance	**cols**	**t**	**cols**	**it**	**t**	**cols**	**it**	**t**	**cols**	**it**	**t**	**cols**	**it**	**t**	**cols**	**it**	**t**
R101_50_A_50	1035	2	1143	16	2	1219	7	2	887	10	2	837	9	2	1059	4	1
R102_50_A_50	1746	9	1646	21	17	1911	9	9	1412	10	12	1491	10	13	1944	5	9
R103_50_A_50	1975	22	1701	36	93	2089	10	51	1801	11	53	1740	10	59	2287	6	39
R104_50_A_50	2287	68	2460	33	604	2657	11	366	1982	10	337	1805	10	506	2457	7	308
R105_50_A_50	1201	6	1181	18	6	1343	7	3	1007	10	4	1002	10	5	1345	5	4
R106_50_A_50	1930	16	1442	21	35	2231	9	16	1703	10	22	1554	10	23	1896	6	16
R107_50_A_50	1995	33	1867	32	149	2057	10	65	1792	10	78	1752	10	86	2373	6	96
R108_50_A_50	2280	83	2393	33	834	2690	11	407	1998	11	423	1939	10	394	2521	7	382
R109_50_A_50	1392	12	1335	23	22	1601	9	11	1339	10	15	1187	9	13	1587	6	11
R110_50_A_50	1475	29	1656	24	87	1865	10	31	1456	11	37	1532	10	43	1775	5	25
R111_50_A_50	1950	82	1674	30	102	2289	10	49	1729	11	70	1702	11	121	2330	7	81
R112_50_A_50	2205	77	1778	31	356	1917	10	83	1826	11	115	1761	10	214	2065	6	99
C101_50_A_100	2190	11	1996	15	10	2337	6	8	1946	11	24	1535	11	18	3270	6	13
C102_50_A_100	4655	283	3832	24	150	5265	10	111	2561	10	124	2776	10	216	4673	7	91
C103_50_A_100	4924	3250	4319	33	1399	5344	11	958	3449	11	1503	3260	10	804	5230	7	383
C104_50_A_100	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
C105_50_A_100	2232	19	2847	20	28	3636	8	24	2033	10	25	2100	10	38	3234	6	23
C106_50_A_100	2216	17	2359	18	17	2775	7	11	1725	10	15	1961	11	26	2651	5	11
C107_50_A_100	2276	26	2065	21	34	2719	8	23	1895	10	32	1583	10	46	3963	6	31
C108_50_A_100	2312	84	2599	26	95	3077	9	64	2424	11	75	2162	10	112	3845	6	47
C109_50_A_100	2475	316	3138	37	416	3725	10	236	3472	11	336	2870	11	441	5059	7	214

Table B.8: *Sensitivity analysis with respect to the number of added cols: 50 customers, initialization with opt lp.*

	Stand.CG		2stage - init:opt_master				2stage - init:opt_lp			
Instance	**cols**	**t**	**cols**	**%$_{cols}$**	**it**	**t**	**cols**	**%$_{cols}$**	**it**	**t**
R101_50_A_50	1035	2	479	-54%	4	1	1059	2%	4	1
R102_50_A_50	1746	9	663	-62%	4	3	1944	11%	5	9
R103_50_A_50	1975	22	753	-62%	4	9	2287	16%	6	39
R104_50_A_50	2287	68	771	-66%	5	24	2457	7%	7	308
R105_50_A_50	1201	6	599	-50%	4	1	1345	12%	5	4
R106_50_A_50	1930	16	676	-65%	5	6	1896	-2%	6	16
R107_50_A_50	1995	33	759	-62%	3	9	2373	19%	6	96
R108_50_A_50	2280	83	874	-62%	3	29	2521	11%	7	382
R109_50_A_50	1392	12	581	-58%	5	4	1587	14%	6	11
R110_50_A_50	1475	29	753	-49%	4	11	1775	20%	5	25
R111_50_A_50	1950	82	976	-50%	5	23	2330	19%	7	81
R112_50_A_50	2205	77	763	-65%	4	23	2065	-6%	6	99
R101_50_B_50	1824	9	793	-57%	3	2	1553	-15%	5	6
R102_50_B_50	3783	55	1140	-70%	4	24	3354	-11%	6	100
R103_50_B_50	3809	182	1356	-64%	5	112	3427	-10%	7	748
R104_50_B_50	4429	1356	1320	-70%	5	1006	x	x	x	x
R105_50_B_50	2248	24	1022	-55%	5	8	1786	-21%	7	17
R106_50_B_50	3527	102	1290	-63%	5	31	2969	-16%	5	291
R107_50_B_50	3791	290	1360	-64%	5	119	3551	-6%	7	1444
R108_50_B_50	x	x	1378	$-\infty$	6	780	x	x	x	x
R109_50_B_50	2364	73	1175	-50%	5	33	2382	1%	6	138
R110_50_B_50	2858	208	1255	-56%	5	174	3083	8%	8	287
R111_50_B_50	3556	488	1415	-60%	5	430	3553	0%	7	906
R112_50_B_50	4457	425	1455	-67%	5	511	3315	-26%	7	1473
R101_50_C_50	2727	27	1112	-59%	5	8	2127	-22%	7	18
R102_50_C_50	4837	166	1683	-65%	5	181	4007	-17%	7	870
R103_50_C_50	5287	1051	x	x	x	x	x	x	x	x
R104_50_C_50	x	x	x	x	x	x	x	x	x	x
R105_50_C_50	3041	72	1291	-58%	6	37	2334	-23%	7	78
R106_50_C_50	4895	453	1773	-64%	5	209	4792	-2%	7	2073
R107_50_C_50	5196	1814	1798	-65%	6	1569	x	x	x	x
R108_50_C_50	6242	2924	x	x	x	x	x	x	x	x
R109_50_C_50	3726	556	1543	-59%	5	151	3513	-6%	8	572
R110_50_C_50	4228	1256	1841	-56%	6	1870	2686	-36%	8	2371
R111_50_C_50	x	x	1947	$-\infty$	6	3565	x	x	x	x
R112_50_C_50	5942	1485	x	x	x	x	x	x	x	x

Table B.9: *Analysis of increasing orders for instances of class R1, 50 customers.*

Instance	Stand.CG cols	Stand.CG t	2stage - init:opt_master cols	2stage - init:opt_master %cols	2stage - init:opt_master it	2stage - init:opt_master t	2stage - init:opt_lp cols	2stage - init:opt_lp %cols	2stage - init:opt_lp it	2stage - init:opt_lp t
C101_50_A_100	2190	11	1000	-54%	4	5	3270	49%	6	13
C102_50_A_100	4655	283	1381	-70%	5	19	4673	0%	7	91
C103_50_A_100	4924	3250	1359	-72%	4	21	5230	6%	7	383
C104_50_A_100	x	x	n/a				x		x	x
C105_50_A_100	2232	19	922	-59%	4	7	3234	45%	6	23
C106_50_A_100	2216	17	868	-61%	4	5	2651	20%	5	11
C107_50_A_100	2276	26	1256	-45%	6	15	3963	74%	6	31
C108_50_A_100	2312	84	894	-61%	4	18	3845	66%	6	47
C109_50_A_100	2475	316	977	-61%	5	52	5059	104%	7	214
C101_50_B_100	3883	122	1615	-58%	5	74	4977	28%	6	145
C102_50_B_100	8360	3450	1735	-79%	6	901	x		x	x
C103_50_B_100	x	x	n/a				x		x	x
C104_50_B_100	x	x	n/a				x		x	x
C105_50_B_100	4193	479	1718	-59%	7	609	5653	35%	7	1095
C106_50_B_100	4198	207	1889	-55%	5	106	4305	3%	6	326
C107_50_B_100	4026	615	1936	-52%	6	429	5282	31%	7	1986
C108_50_B_100	4634	1725	1928	-58%	6	2317	x		x	x
C109_50_B_100	x	x	n/a			x		x		x
C101_50_C_100	6298	680	2449	-61%	7	380	7063	12%	7	2479
C102_50_C_100	x	x	n/a				x		x	x
C103_50_C_100	x	x	n/a				x		x	x
C104_50_C_100	x	x	n/a				x		x	x
C105_50_C_100	6324	3023	2480	-61%	7	899	x		x	x
C106_50_C_100	6746	828	2032	-70%	7	355	x		x	x
C107_50_C_100	6402	3215	x		x	x	x		x	x
C108_50_C_100	x	x	n/a				x		x	x
C109_50_C_100	x	x	n/a				x		x	x

Table B.10: *Analysis of increasing orders for instances of class C1, 50 customers.*

Instance	|C|	|E|	Irnich Exact $\|C_{sub}\|$	Irnich Exact $\|E_{sub}\|$	Irnich Exact t_{tot}	2stage Exact - init:opt_master $\|C_{sub}\|$	2stage Exact - init:opt_master $\|C_{ina}\|$	2stage Exact - init:opt_master $\|E_{sub}\|$	2stage Exact - init:opt_master $\|E_{ina}\|$	2stage Exact - init:opt_master t_{tot}	2stage Exact - init:opt_lp $\|C_{sub}\|$	2stage Exact - init:opt_lp $\|C_{ina}\|$	2stage Exact - init:opt_lp $\|E_{sub}\|$	2stage Exact - init:opt_lp $\|E_{ina}\|$	2stage Exact - init:opt_lp t_{tot}
R101_25_C_100	175	625	73	560	2	75	16	40	547	10	74	25	54	531	20
R102_25_C_100	175	625	0	190	10	0	54	0	525	350	x	x	x	x	x
R103_25_C_100	175	625	81	537	800	83	5	41	523	1356	x	x	x	x	x
R104_25_C_100	175	625	x	x	x	x	x	x	x	x	x	x	x	x	x
R105_25_C_100	175	625	79	551	9	85	1	37	538	54	80	15	53	524	250
R106_25_C_100	175	625	0	150	111	0	65		520	322	x	x	x	x	x
R107_25_C_100	175	625	0	77	2014	x	x	x	x	x	x	x	x	x	x
R108_25_C_100	175	625	24	469	660	x	x	x	x	x	x	x	x	x	x
R109_25_C_100	175	625	81	540	44	0	58	0	522	209	x	x	x	x	x
R110_25_C_100	175	625	0	64	116	0	66	0	525	1966	x	x	x	x	x
R111_25_C_100	175	625	0	80	182	x	x	x	x	x	x	x	x	x	x
R112_25_C_100	175	625	0	0	57	0	60	0	521	1433	x	x	x	x	x
R101_50_A_50	150	2500	0	1605	4	0	22	0	2368	14	0	0	17	2319	185
R102_50_A_50	150	2500	0	995	61	x	x	x	x	x	x	x	x	x	x
R103_50_A_50	150	2500	0	601	751	x	x	x	x	x	x	x	x	x	x
R104_50_A_50	150	2500	x	x	x	x	x	x	x	x	x	x	x	x	x
R105_50_A_50	150	2500	0	1326	10	0	22	0	2365	69	x	x	x	x	x
R106_50_A_50	150	2500	0	794	148	x	x	x	x	x	x	x	x	x	x
R107_50_A_50	150	2500	0	462	950	x	x	x	x	x	x	x	x	x	x
R108_50_A_50	150	2500	x	x	x	x	x	x	x	x	x	x	x	x	x
R109_50_A_50	150	2500	0	851	42	0	18	0	2349	1696	x	x	x	x	x
R110_50_A_50	150	2500	0	399	862	x	x	x	x	x	x	x	x	x	x
R111_50_A_50	150	2500	0	415	1335	x	x	x	x	x	x	x	x	x	x
R112_50_A_50	150	2500	x	x	x	x	x	x	x	x	x	x	x	x	x

Table B.11: *Comparison between variable elimination (Irnich et al., 2010) and two stage column generation: exact DP.*

Instance	|C|	|E|	Irnich Relax			2stage Relax - init:opt_master					2stage Relax - init:opt_lp				
			$\|C_{sub}\|$	$\|E_{sub}\|$	$\mathbf{t}_{tot}$	$\|C_{sub}\|$	$\|C_{ina}\|$	$\|E_{sub}\|$	$\|E_{ina}\|$	$\mathbf{t}_{tot}$	$\|C_{sub}\|$	$\|C_{ina}\|$	$\|E_{sub}\|$	$\|E_{ina}\|$	$\mathbf{t}_{tot}$
R101_25_C_100	175	625	0	349	1	74	17	41	545	1	72	15	50	533	1
R102_25_C_100	175	625	0	190	5	0	57	0	523	7	0	38	4	504	8
R103_25_C_100	175	625	0	100	20	0	26	0	501	39	0	22	2	478	69
R104_25_C_100	175	625	0	53	37	0	22	0	481	100	0	9	1	457	283
R105_25_C_100	175	625	0	275	5	0	29	0	521	4	0	28	6	512	4
R106_25_C_100	175	625	0	150	18	0	25	0	488	19	0	26	3	492	39
R107_25_C_100	175	625	0	77	36	0	9	0	421	166	0	10	1	409	296
R108_25_C_100	175	625	0	37	39	0	14	0	478	317	0	9	0	455	488
R109_25_C_100	175	625	0	173	18	0	4	0	425	43	0	0	2	413	34
R110_25_C_100	175	625	0	64	28	0	11	0	454	144	0	10	1	465	96
R111_25_C_100	175	625	0	80	28	0	7	0	428	97	0	2	0	452	151
R112_25_C_100	175	625	0	0	31	0	19	0	473	115	0	9	0	461	211
R101_50_A_50	150	2500	0	1605	2	0	22	0	2368	1	0	0	17	2319	2
R102_50_A_50	150	2500	0	995	10	0	18	0	2350	4	0	0	13	2276	17
R103_50_A_50	150	2500	0	601	24	0	11	0	2192	38	0	0	5	2121	93
R104_50_A_50	150	2500	0	182	71	0	6	0	2190	75	0	0	1	2146	604
R105_50_A_50	150	2500	0	1326	6	0	16	0	2352	2	0	0	14	2304	6
R106_50_A_50	150	2500	0	794	16	0	17	0	2335	7	0	0	9	2271	35
R107_50_A_50	150	2500	0	462	34	0	8	0	2225	30	0	0	4	2160	149
R108_50_A_50	150	2500	0	128	85	0	4	0	2209	59	0	0	0	2145	834
R109_50_A_50	150	2500	0	851	13	0	12	0	2327	6	0	0	7	2255	22
R110_50_A_50	150	2500	0	399	30	0	9	0	2298	15	0	0	2	2246	87
R111_50_A_50	150	2500	0	415	83	0	6	0	2202	53	0	0	3	2187	102
R112_50_A_50	150	2500	0	1	79	0	7	0	2235	42	0	0	0	2168	356

Table B.12: *Comparison between variable elimination (Irnich et al., 2010) and two stage column generation: relaxed DP.*

Bibliography

Addis, B., Carello, G. and Ceselli, A. (2009). A branch-and-price-and-cut approach for a two-level hierarchical location problem, *Proceedings of the International Network Optimization Conference (INOC)*.

Ahuja, R. K., Magnanti, T. L. and Orlin, J. B. (1993). *Network flows: theory, algorithms and applications*, Prentice Hall.

Alves, C. and Valerio de Carvalho, J. M. (2008). A stabilized branch-and-price-and-cut algorithm for the multiple length curring stock problem, *Computers and Operations Research* **35**(4): 1315–1328.

Archetti, C., Savelsbergh, M. W. P. and Speranza, M. G. (2006). Worst-case analysis for split delivery vehicle routing problems, *Transportation Science* **40**: 226–234.

Archetti, C., Savelsbergh, M. W. P. and Speranza, M. G. (2008). To split or not to split: That is the question, *Transportation Research Part E* **44**: 114–123.

Archetti, C. and Speranza, M. G. (2008). The split delivery vehicle routing problem: A survey, *in* B. Golden, S. Raghavan and E. Wasil (eds), *The Vehicle Routing Problem: Latest Advances and New Challenges*, Vol. 43, Springer US, pp. 103–122.

Archetti, C., Speranza, M. G. and Hertz, A. (2006). A tabu search algorithm for the split delivery vehicle routing problem, *Transportation Science* **40**: 64 – 73.

Archetti, C., Speranza, M. G. and Savelsbergh, M. W. P. (2008). An optimization-based heuristic for the split delivery vehicle routing problem, *Transportation Science* **42**(1): 22–31.

Barnhart, C., Hane, C. A. and Vance, P. H. (2000). Using branch-and-price-and-cut to solve origin-destination integer multicommodity flow problems, *Operations Research* **48**(2): 318–326.

Barnhart, C., Johnson, E. L., Nemhauser, G. L., Savelsbergh, M. W. P. and Vance, P. H. (1998). Branch-and-price: Column generation for solving huge integer programs, *Operations Research* **46**(3): 316–329.

Beasley, J. E. and Christofides, N. (1989). An algorithm for the resource constrained shortest path problem, *Networks* **19**: 379–394.

Belenguer, J., Martinez, M. and Mota, E. (2000). A lower bound for the split delivery vehicle routing problem, *Operations Research* **48**: 801–810.

Belfiore, P., Tsugunobu, H. and Yoshizaki, Y. (2009). Scatter search for a real-life heterogeneous fleet vehicle routing problem with time windows and split deliveries in brazil, *European Journal of Operational Research* **198**(3): 750–758.

Ben Amor, H. M. (2002). *Stabilization de l'Algorithme de Génération de Colonnes*, PhD thesis, École Polytechnique de Montréal.

Bierwirth, C. and Meisel, F. (2009). A fast heuristic for quay crane scheduling with interference constraints, *Journal of Scheduling* **12**(4): 345–360.

Bierwirth, C. and Meisel, F. (2010). A survey of berth allocation and quay crane scheduling problems in container terminals, *European Journal of Operational Research* **202**(3): 615–627.

Bish, E. K. (2003). A multiple-crane-constrained scheduling problem in a container terminal, *European Journal of Operational Research* **144**: 83–107.

Bish, E. K., Leong, T. Y., Li, C. L., Ng, J. W. C. and Simchi-Levi, D. (2001). Analysis of a new vehicle scheduling and location problem, *Naval Research Logistics* **48**: 363–385.

Boland, N., Dethridge, J. and Dumitrescu, I. (2006). Accelerated label setting algorithms for the elementary resource constrained shortest path problem, *Operations Research Letters* **34**(1): 58–68.

Bolduc, M.-C., Laporte, G., Renaud, J. and Boctor, F. F. (2010). A tabu search heuristic for the split delivery vehicle routing problem with production and demand calendars, *European Journal of Operational Research* **202**(1): 122–130.

Briant, O., Lemarchal, C., Meurdesoif, P., Michel, S., Perrot, N. and Vanderbeck, F. (2008). Comparison of bundle and classical column generation, *Mathematical Programming* **113**: 299–344.

Brønmo, G., Nygreen, B. and Lysgaard, J. (2010). Column generation approaches to ship scheduling with flexible cargo sizes, *European Journal of Operational Research* **200**(1): 139–150.

Buhrkal, K., Zuglian, S., Ropke, S., Larsen, J. and Lusby, R. (2009). Models for the discrete berth allocation problem: a computational comparison, *Technical report*, Technical University of Denmark.

Canonaco, P., Legato, P., Mazza, R. M. and Musmanno, R. (2008). A queuing network model for the management of berth crane operations, *Computers & Operations Research* **35**(8): 2432 – 2446.

Ceselli, A., Gatto, M., Lübbecke, M. E., Nunkesser, M. and Schilling, H. (2008). Optimizing the cargo express service of swiss federal railways, *Transportation Science* **42**(4): 450–465.

Ceselli, A. and Righini, G. (2005). A branch-and-price algorithm for the capacitated p-median problem, *Networks* **45**(3): 125–142.

Ceselli, A., Righini, G. and Salani, M. (2009a). A column generation algorithm for a vehicle routing problem with economies of scale and additional constraints, *Transportation Science* **43**(1): 56–69.

Ceselli, A., Righini, G. and Salani, M. (2009b). Column generation for the split delivery vehicle routing problem, *Technical report*, Note del Polo - Ricerca 118, Dipartimento di Tecnologie dell'Informazione, Università degli Studi di Milano.

Chen, P., Golden, B. L. and Wasil, E. (2007). The split delivery vehicle routing problem: Applications, algorithms, test problems and computational results, *Networks* **49**(4): 318–329.

Cheng, Y. L., Sen, H. C., Natarajan, K., Teo, C. P. and Tan, K. C. (2005). Dispatching automated guided vehicles in a container terminal, *in* J. Geunes and P. M. Pardalos (eds), *Supply Chain Optimization*, Applied Optimization, Springer.

Christiansen, M., Fagerholt, K. and Ronen, D. (2004). Ship routing and scheduling: status and perspectives, *Transportation Science* **38**: 1–18.

Cohn, A. M. and Barnhart, C. (2003). Improving crew scheduling by incorporating key maintenance routing decisions, *Operations Research* **51**(3): 387–396.

Cordeau, J. F., Gaudioso, M., Laporte, G. and Moccia, L. (2006). A memetic heuristic for the generalized quadratic assignment problem, *INFORMS Journal on Computing* **18**(4): 433–443.

Cordeau, J. F., Gaudioso, M., Laporte, G. and Moccia, L. (2007). The service allocation problem at the Gioia Tauro maritime terminal, *European Journal of Operational Research* **176**(2): 1167–1184.

Cordeau, J. F., Laporte, G., Legato, P. and Moccia, L. (2005). Models and tabu search heuristics for the berth-allocation problem, *Transportation Science* **39**(4): 526–538.

Cordeau, J. F., Laporte, G., Pasin, F. and Ropke, S. (2010). Scheduling technicians and tasks in a telecommunications company, *Journal of Scheduling* **13**: 393–409.

Crainic, T. G. and Kim, K. H. (2007). Intermodal transportation, *in* C. Barnhart and G. Laporte (eds), *Transportation*, Vol. 14 of *Handbooks in Operations Research and Management Science*, Elsevier, pp. 467–537.

Daganzo, C. F. (1989). The crane scheduling problem, *Transportation Research Part B* **23**: 159–175.

Dantzig, G. B. and Wolfe, P. (1960). Decomposition principle for linear programs, *Operations Research* **8**: 101–111.

de Castilho, B. and Daganzo, C. F. (1993). Handling strategies for import containers at marine terminals, *Transportation Research Part B* **27**: 151–166.

Dekker, R., Voogd, P. and van Asperen, E. (2006). Advanced methods for container stacking, *OR Spectrum* **28**: 563–586.

Dell'Amico, M., Righini, G. and Salani, M. (2006). A branch-and-price approach to the vehicle routing problem with simultaneous distribution and collection, *Transportation Science* **40**(2): 235–247.

Desaulniers, G. (2010). Branch-and-price-and-cut for the split-delivery vehicle routing problem with time windows, *Operations Research* **58**(1): 179–192.

Desaulniers, G., Desrosiers, J. and Solomon, M. (eds) (2005). *Column Generation*, GERAD 25th Anniversary Series, Springer.

Desaulniers, G., Lavigne, J. and Soumis, F. (1998). Multi-depot vehicle scheduling problems with time windows and waiting costs, *European Journal of Operational Research* **111**(3): 479 – 494.

Desrochers, G., Desrosiers, J. and Solomon, M. (1992). A new optimization algorithm for the vehicle routing problem with time windows, *Operations Research* **40**: 342–354.

Desrochers, G. and Soumis, F. (1989). A column generation approach to the urban transit crew scheduling problem, *Transportation Science* **23**: 1–13.

Desrosiers, J. and Lübbecke, M. E. (2005). A primer in column generation, *in* G. Desaulniers, J. Desrosiers and M. Solomon (eds), *Column Generation*, GERAD, chapter 1, pp. 1–32.

Desrosiers, J., Soumis, F. and Desrochers, G. (1984). Routing with time-windows by column generation, *Networks* **14**: 545–565.

Diaz, J. and Fernandez, E. (2002). A branch-and-price algorithm for a single source capacitated plant location problem, *Journal of the Operational Research Society* **53**(7): 728–740.

Dror, M. (1994). Note on the complexity of the shortest path models for column generation in vrptw, *Operations Research* **42**: 977–978.

Dror, M., Laporte, G. and Trudeau, P. (1994). Vehicle routing with split deliveries, *Discrete and Applied Mathematics* **50**: 239–254.

Dror, M. and Trudeau, P. (1989). Savings by split delivery routing, *Transportation Science* **23**: 141–145.

Dror, M. and Trudeau, P. (1990). Split delivery routing, *Naval Research Logistics* **37**: 383–402.

du Merle, O., Villeneuve, D., Desrosiers, J. and Hansen, P. (1999). Stabilized column generation, *Discrete Mathematics* **194**: 229–237.

Eggenberg, N. (2009). *Combining robustness and recovery for airline schedules*, PhD thesis, Ecole Polytechnique Fédérale de Lausanne.

Elhallaoui, I., Desaulniers, G., Metrane, A. and Soumis, F. (2008). Bi-dynamic constraint aggregation and subproblem reduction, *Computers & Operations Research* **35**(5): 1713 – 1724.

Elhallaoui, I., Metrane, A., Soumis, F. and Desaulniers, G. (2010). Multi-phase dynamic constraint aggregation for set partitioning type problems, *Mathematical Programming* **123**: 345–370.

Elhallaoui, I., Villeneuve, D., Soumis, F. and Desaulniers, G. (2005). Dynamic Aggregation of Set-Partitioning Constraints in Column Generation, *Operations Research* **53**(4): 632–645.

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

Fukasawa, R., Lysgaard, J., Poggi de Aragão, M., Reis, M., Uchoa, E. and Werneck, R. F. (2006). Robust branch-and-cut-and-price for the capacitated vehicle routing problem, *Mathematical Programming* **106**(3): 491–511.

Gamache, M., Soumis, F., Marquis, G. and Desrosiers, J. (1999). A column generation approach for large-scale aircrew rostering problems, *Operations Research* **47**(2): 247–263.

Gambardella, L. M., Mastrolilli, M., Rizzoli, A. E. and Zaffalon, M. (2001). An optimization methodology for intermodal terminal management, *Journal of Intelligent Manufacturing* **12**: 521–534.

Gambardella, L. M., Rizzoli, A. E. and Zaffalon, M. (1998). Simulation and planning of an intermodal container terminal, *Simulation* **71**: 107–116.

Garey, M. R. and Johnson, D. S. (1979). *Computers and intractability: a guide to the theory of NP-completeness*, Freeman & co., New York.

Gélinas, E., Desrochers, G., Desrosiers, J. and Solomon, M. (1995). A new branching strategy for time constrained routing problems with application to backhauling, *Annals of Operations Research* **61**: 91–109.

Gendreau, M., Dejax, P., Feillet, D. and Gueguen, C. (2006). Vehicle routing with time windows and split deliveries, *Technical report*, Technical report 2006-851, Laboratoire d'Informatique d'Avignon.

Geoffrion, A. M. (1974). Lagrangean relaxation for integer programming, *Mathematical Programming Study* **2**: 82–114.

Giallombardo, G., Moccia, L., Salani, M. and Vacca, I. (2010). Modeling and solving the tactical berth allocation problem, *Transportation Research Part B* **44**(2): 232–245.

Gilmore, P. and Gomory, R. (1961). A linear programming approach to the cutting stock problem, *Operations Research* **9**: 849–859.

Gilmore, P. and Gomory, R. (1963). A linear programming approach to the cutting stock problem, part ii, *Operations Research* **11**: 863–888.

Goodchild, A. V. and Daganzo, C. F. (2006). Double-cycling strategies for container ships and their effect on ship loading and unloading operations, *Transportation Science* **40**: 473–483.

Goodchild, A. V. and Daganzo, C. F. (2007). Crane double cycling in container ports: planning methods and evaluation, *Transportation Research Part B* **41**: 875–891.

Grønhaug, R., Christiansen, M., Desaulniers, G. and Desrosiers, J. (2010). A branch-and-price method for a liquefied natural gas inventory routing problem, *Transportation Science* **44**(3): 400–415.

Gueguen, C. (1999). *Méthodes de résolution exacte pour les problèmes de tournées de véhicules*, PhD thesis, Ecole centrale Paris.

Gulczynski, D., Golden, B. and Wasil, E. (2009). The split delivery vehicle routing problem with minimum delivery amounts, *Transportation Research Part E* **46**: 612–626.

Hahn, P., Kim, B. J., Guignard, M., Smith, J. and Zhu, Y. R. (2008). An algorithm for the generalized quadratic assignment problem, *Computational Optimization and Applications* **40**(3): 351–372.

Han, Y., Lee, L. H., Chew, E. P. and Tan, K. C. (2008). A yard storage strategy for minimizing traffic congestion in a marine container transshipment hub, *OR Spectrum* **30**: 697–720.

Handler, G. Y. and Zang, I. (1980). A dual algorithm for the constrained shortest path problem, *Networks* **10**: 293–310.

Henesey, L. (2006). *Multi-agent Container Terminal Management*, PhD thesis, Karlshamn, Blekinge Institute of Technology.

Hennig, F. (2010). *Optimization in Maritime Transportation: Crude Oil Tanker Routing and Scheduling*, PhD thesis, Norwegian University of Science and Technology.

Ho, S. and Haugland, D. (2004). A tabu search heuristic for the vehicle routing problem with time windows and split deliveries, *Computers and Operations Research* **31**: 1947–1964.

Holberg, K. and Yuan, D. (2003). A multicommodity network-flow problem with side constraints and paths solved by column generation, *Journal on Computing* **15**(1): 42–57.

Huisman, D. (2007). A column generation approach for the rail crew re-scheduling problem, *European Journal of Operational Research* **180**(1): 163–173.

Hwang, H.-S., Visoldilokpun, S. and Rosenberger, J. M. (2008). A branch-and-price-and-cut method for ship scheduling with limited risk, *Transportation Science* **42**(3): 336–351.

Imai, A., Chen, H. C., Nishimura, E. and Papadimitriou, S. (2008). The simultaneous berth and quay crane allocation problem, *Transportation Research Part E* **44**(5): 900–920.

Imai, A., Nagaiwa, K. and Chan, W. T. (1997). Efficient planning of berth allocation for container terminals in Asia, *Journal of Advanced Transportation* **31**(1): 75–94.

Imai, A., Nishimura, E. and Papadimitriou, S. (2001). The dynamic berth allocation problem for a container port, *Transportation Research Part B* **35**(4): 401–417.

Imai, A., Nishimura, E. and Papadimitriou, S. (2003). Berth allocation with service priority, *Transportation Research Part B* **37**(5): 437–457.

Imai, A., Sun, X., Nishimura, E. and Papadimitriou, S. (2005). Berth allocation in a container port: using a continuous location space approach, *Transportation Research Part B* **39**(3): 199–221.

Irnich, S. (2008). Resource extension functions: Properties, inversion, and generalization to segments., *OR Spectrum* **30**(1): 113–148.

Irnich, S. (2010). A new branch-and-price algorithm for the traveling tournament problem, *European Journal of Operational Research* **204**(2): 218 – 228.

Irnich, S., Desaulniers, G., Desrosiers, J. and Hadjar, A. (2010). Path reduced costs for eliminating arcs, *Journal on Computing* **22**(2): 297–313.

ISL (2009). *Shipping Statistics and Market review (SSMR) - Short Comment*, Vol. 53, Institute of Shipping Economics and Logistics, http://www.isl.org.

Jin, M., Liu, J. and Bowden, R. (2007). A two stage algorithm with valid inequalitites for the split delivery vehicle routing problem, *International Journal of Production Economics* **105**: 228–242.

Jin, M., Liu, J. and Eksiouglu, B. (2008). A column generation approach for the split delivery vehicle routing problem, *Operations Research Letters* **36**: 265–270.

KALMAR (2010). Kalmar container handling systems: complete range of products and knowhow, Kalmar Industries. http://www.kalmarind.com/source.php/1039687/Container0brochure.pdf.

Kang, J., Ryu, K. R. and Kim, K. H. (2006). Deriving stacking strategies for export containers with uncertain weight information, *Journal of Intelligent Manufacturing* **17**: 399–410.

Kaparis, K. and Letchford, A. (2010). Separation algorithms for 0-1 knapsack polytopes, *Mathematical Programming* **124**: 69–91.

Kim, K. H. and Bae, J. W. (1998). Re-marshaling export containers in port container terminals, *Computers and Industrial Engineering* **35**: 655–658.

Kim, K. H. and Bae, J. W. (2004). A look-ahead dispatching method for automated guided vehicles in automated port container terminals, *Transportation Science* **38**(2): 224–234.

Kim, K. H., Jeon, S. and Ryu, K. R. (2007). Deadlock prevention for automated guided vehicles in automated container terminals, *Container terminals and cargo systems*, Springer, pp. 243–263.

Kim, K. H. and Kim, K. Y. (1999). Routing straddle carriers for the loading operation of containers using a beam search algorithm, *Computers and Industrial Engineering* **36**(1): 109–136.

Kim, K. H., Lee, K. M. and Hwang, H. (2003). Sequencing delivery and receiving operations for yard cranes in port container terminals, *International Journal of Production Economics* **84**: 283–292.

Kim, K. H., Lee, S. J., Park, Y. M., Yang, C. H. and Bae, J. W. (2006). Dispatching yard cranes in port container terminals, *TRB Transportation Research Board Annual Meeting*.

Kim, K. H. and Park, Y. M. (2004). A crane scheduling method for port container terminals, *European Journal of Operational Research* **156**(3): 752–768.

Kim, K. H., Park, Y. M. and Ryu, K. R. (2000). Deriving decision rules to locate export containers in container yards, *European Journal of Operational Research* **124**: 89–101.

Klabjan, D. (2005). Large-scale models in the airline industry, *in* G. Desaulniers, J. Desrosiers and M. Solomon (eds), *Column generation*, Springer, pp. 163–195.

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

Lan, S., Clarke, J.-P. and Barnhart, C. (2006). Planning for robust airline operations: Optimizing aircraft routings and flight departure times to minimize passenger disruptions, *Transportation Science* **40**: 15–28.

Lau, Y. and Lee, M. (2007). Simulation study of port container terminal quay side traffic, *in* J. Park, T. Kim and Y. Kim (eds), *AsiaSim 2007*, Springer, pp. 227–236.

Lee, D. H., Cao, J. X., Shi, Q. and Chen, J. H. (2009). A heuristic algorithm for yard truck scheduling and storage allocation problems, *Transportation Research Part E* **45**(5): 810–820.

Lee, D. H., Wang, H. Q. and Miao, L. (2008). Quay crane scheduling with non-interference constraints in port container terminals, *Transportation Research Part E* **44**: 124–135.

Lee, L. H., Chew, E. P., Tan, K. C. and Han, Y. (2006). An optimization model for storage yard management in transshipment hubs, *OR Spectrum* **28**: 539–561.

Lee, Y. and Hsu, N. Y. (2007). An optimization model for the container pre-marshalling problem, *Computers and Operations Research* **34**: 3295–3313.

Legato, P. and Mazza, R. M. (2001). Berth planning and resources optimisation at a container terminal via simulation, *European Journal of Operational Research* **133**: 537–547.

Legato, P., Mazza, R. M. and Trunfio, R. (2010). Simulation-based optimization for discharge/loading operations at a maritime container terminal, *OR Spectrum* **32**: 543–567.

Liberatore, F., Righini, G. and Salani, M. (2010). A column generation algorithm for the vehicle routing problem with soft time windows, *4OR : A Quarterly Journal of Operations Research* **to appear**: DOI 10.1007/s10288–010–0136–6. Accepted for publication.

Lim, A. (1998). The berth planning problem, *Operations Research Letters* **22**(2-3): 105–110.

Linn, R. and Zhang, C. (2003). A heuristic for dynamic yard crane deployment in a container terminal, *IIE Transactions* **35**: 161174.

Liu, C. I., Jula, H., Vukadinovic, K. and Ioannou, P. (2004). Automated guided vehicle system for two container yard layouts, *Transportation Research Part C* **12**: 349–368.

Lohatepanont, M. and Barnhart, C. (2004). Airline Schedule Planning: Integrated Models and Algorithms for Schedule Design and Fleet Assignment, *Transportation Science* **38**(1): 19–32.

Lübbecke, M. E. (2010). Column generation, *in* J. J. Cochran (ed.), *Wiley Encyclopedia of Operations Research and Management Science*, John Wiley and Sons, Chichester, UK.

Lübbecke, M. E. and Desrosiers, J. (2005). Selected topics in column generation, *Operations Research* **53**(6): 1007–1023.

Martello, S., Pisinger, D. and Toth, P. (2000). New trends in exact algorithms for the 0-1 knapsack problem, *European Journal of Operational Research* **123**(2): 325–332.

Martello, S. and Toth, P. (eds) (1990). *Knapsack problems: Algorithms and Computer Implementations*, Wiley, Chichester, UK.

Mauri, G., Oliveira, A. and Lorena, L. (2008). A hybrid column generation approach for the berth allocation problem, *in* J. van Hemert and C. Cotta (eds), *Evolutionary Computation in Combinatorial Optimization*, Vol. 4972 of *Lecture Notes in Computer Science*, Springer Berlin / Heidelberg, pp. 110–122.

Meisel, F. and Bierwirth, C. (2006). Integration of berth allocation and crane assignment to improve the resource utilization at a seaport container terminal, *Operations Research Proceedings 2005*, Springer, pp. 105–110.

Meisel, F. and Bierwirth, C. (2009). Heuristics for the integration of crane productivity in the berth allocation problem, *Transportation Research Part E* **45**(1): 196–209.

Moccia, L., Cordeau, J. F., Gaudioso, M. and Laporte, G. (2006). A branch-and-cut algorithm for the quay crane scheduling problem in a container terminal, *Naval Research Logistics* **53**(1): 45–59.

Moccia, L., Cordeau, J. F., Monaco, M. F. and Sammarra, M. (2009). A column generation heuristic for a dynamic generalized assignment problem, *Computers & Operations Research* **36**(9): 2670 – 2681.

Möhring, R. H., Köhler, E., Gawrilow, E. and Stenzel, B. (2005). Conflict-free real-time agv routing, *Operations Research Proceedings 2004*, pp. 18–24.

Moorthy, R. L., Hock-Guan, W., Ng, J. W. C. and Teo, C. P. (2003). Cyclic deadlock prediction and avoidance for zone-controlled agv system, *International Journal of Production Economics* **83**(3): 309–324.

Moorthy, R. and Teo, C. P. (2006). Berth management in container terminal: the template design problem, *OR Spectrum* **28**(4): 495–518.

Nakao, Y. and Nagamochi, H. (2007). A dp-based heuristic algorithm for the discrete split delivery vehicle routing problem, *Journal of Advanced Mechanical Design, Systems, and Manufacturing* **1**(2): 217–226.

Ndiaye, B. M., Tao, P. D. and Thi, H. A. (2008). Single straddle carrier routing problem in port container terminals: Mathematical model and solving approaches, *in* H. A. Le Thi, P. Bouvry and T. Pham Dinh (eds), *Modelling, Computation and Optimization in Information Systems and Management Sciences*, Vol. 14 of *Communications in Computer and Information Science*, Springer Berlin Heidelberg.

Nemhauser, G. L. and Wolsey, L. A. (eds) (1988). *Integer and Combinatorial Optimization*, John Wiley & Sons, New York, N. Y.

Ng, J. W. C. (2005). Crane scheduling in container yards with inter-crane interference, *European Journal of Operational Research* **164**: 64–78.

Ng, J. W. C. and Mak, K. L. (2005). Yard crane scheduling in port container terminals, *Applied Mathematical Modelling* **29**: 263–276.

Notteboom, T. E. (2004). Container shipping and ports: an overview, *The Review of Network Economics* **3**: 86–106.

Nowak, M., Ergun, O. and White III, C. C. (2009). An empirical study on the benefit of split loads with the pickup and delivery problem, *European Journal of Operational Research* **198**(3): 734–740.

Park, Y. M. and Kim, K. H. (2003). A scheduling method for berth and quay cranes, *OR Spectrum* **25**(1): 1–23.

Peeters, M. and Kroon, L. (2008). Circulation of railway rolling stock: a branch-and-price approach, *Computers and Operations Research* **35**(2): 538–556.

Peterkofsky, R. I. and Daganzo, C. F. (1990). A branch and bound solution method for the crane scheduling problem, *Transportation Research Part B* **24**: 159–172.

Pisinger, D. (2005). Where are the hard knapsack problems?, *Computers and Operations Research* **32**(9): 2271–2284.

Poggi de Aragão, M. and Uchoa, E. (2003). Integer program reformulation for robust branch-and-price algorithms, *Proceedings of Mathematical Programming in Rio: A conference in honour of Nelson Maculan*, pp. 56–61.

Raymond, V., Soumis, F. and Orban, D. (2010). A new version of the improved primal simplex for degenerate linear programs, *Computers & Operations Research* **37**(1): 91 – 98.

Ribeiro, C. and Soumis, F. (1994). A column gneration approach to the multiple-depot vehicle scheduling problem, *Operations Research* **142**: 41–52.

Righini, G. and Salani, M. (2006). Symmetry helps: Bounded bi-directional dynamic programming for the elementary shortest path problem with resource constraints, *Discrete Optimization* **3**(3): 255–273.

Righini, G. and Salani, M. (2008). New dynamic programming algorithms for the resource constrained shortest path problem., *Networks* **51**(3): 155–170.

Rousseau, L.-M., Gendreau, M. and Feillet, D. (2007). Interior point stabilization for column generation, *Operations Research Letters* **35**(5): 660 – 668.

Ryan, D. M. and Foster, B. A. (1981). An integer programming approach to scheduling, *in* W. A. (ed.), *Computer Scheduling of Public Transportation Urban Passenger and Crew Scheduling*, North-Holland, pp. 269–280.

Salani, M. (2006). *Branch-and-price algorithms for vehicle routing problems*, PhD thesis, Università degli Studi di Milano.

Salani, M. and Vacca, I. (2009). Branch and price for the vehicle routing problem with discrete split deliveries and time windows, *Technical Report TRANSP-OR 091224*, Ecole Polytechnique Fédérale de Lausanne.

Salani, M., Vacca, I. and Bierlaire, M. (2010). Two-stage column generation, *Technical Report TRANSP-OR 101130*, Ecole Polytechnique Fédérale de Lausanne.

Sammarra, M., Cordeau, J. F., Laporte, G. and Monaco, M. F. (2007). A tabu search heuristic for the quay crane scheduling problem, *Journal of Scheduling* **10**: 327–336.

Sandhu, R. and Klabjan, D. (2007). Integrated Airline Fleeting and Crew-Pairing Decisions, *Operations Research* **55**(3): 439–456.

Senne, E. L., Lorena, L. A. and Pereira, M. A. (2005). A branch-and-price approach to p-median location problems, *Computers and Operations Research* **32**(6): 1655–1664.

Solomon, M. (1983). *Vehicle Routing and Scheduling with Time Windows Constraints: Models and Algorithms*, PhD thesis, University of Pennsylvania.

Stahlbock, R. and Voss, S. (2008). Operations research at container terminals: a literature update, *OR Spectrum* **30**(1): 1–52.

Steenken, D., Hennig, A., Freigang, S. and Voss, S. (1993). Routing of straddle carriers at a container terminal with the special aspect of internal moves, *OR Spectrum* **15**(3): 167–172.

Steenken, D., Voss, S. and Stahlbock, R. (2004). Container terminal operation and operations research - a classification and literature review, *OR Spectrum* **26**(1): 3–49.

Taleb-Ibrahimi, M., de Castilho, B. and Daganzo, C. F. (1993). Storage space vs handling work in container terminals, *Transportation Research Part B* **27**: 13–32.

Tongzon, J. and Heng, W. (2005). Port privatization, efficiency and competitiveness: some empirical evidence from container ports (terminals), *Transportation Research Part A* **39**(5): 405–424.

Toth, P. and Vigo, D. (2002). An overview of vehicle routing problems, *in* P. Toth and D. Vigo (eds), *The vehicle routing problem*, Society for Industrial and Applied Mathematics (SIAM), chapter 1, pp. 1–26.

UNCTAD (2009). *Review of Maritime Transport*, United Nations Conference on Trade and Development, http://www.unctad.org.

Vacca, I., Bierlaire, M. and Salani, M. (2007). Optimization at container terminals: Status, trends and perspectives, *Proceedings of the 7th Swiss Transport Research Conference (STRC)*.

Vacca, I., Salani, M. and Bierlaire, M. (2010a). Optimization of operations in container terminals: hierarchical vs integrated approaches, *Proceedings of the 10th Swiss Transport Research Conference (STRC)*.

Vacca, I., Salani, M. and Bierlaire, M. (2010b). Recursive column generation for the tactical berth allocation problem, *Proceedings of the 7th Triennial Symposium on Transportation Analysis (TRISTAN VII)*.

Valerio de Carvalho, J. M. (1999). Exact solution of bin-packing problems using column generation and branch-and-bound, *Annals of Operations Research* **86**: 629–659.

Vance, P. H., Barnhart, C., Johnson, E. L. and Nemhauser, G. L. (1994). Solving binary cutting stock problem by conlumn generation and branch-and-bound, *Computational Optimization and Applications* **3**: 111–130.

Vance, P. H., Barnhart, C., Johnson, E. L. and Nemhauser, G. L. (1997). Airline crew scheduling: A new formulation and decomposition algorithm, *Operations Research* **45**(2): 188–200.

Vanderbeck, F. (1999). Computational study of a column generation algorithm for bin packing and cutting stock problems, *Mathematical Programming* **86**(3): 565–594.

Vanderbeck, F. (2001). A nested decomposition approach to a three-stage, two-dimensional cutting-stock problem, *Management Science* **47**(6): pp. 864–879.

Vanderbeck, F. (2005). *Implementing Mixed Integer Column Generation*, Springer-Verlag, pp. 331–358.

Vanderbeck, F. and Wolsey, L. A. (1996). An exact algorithm for ip column generation, *Operations Research Letters* **19**: 151–159.

Vanelslander, T. (2005). *The economics behind co-operation and competition in seaport container handling*, PhD thesis, Department of Transport and Regional Economics, University of Antwerp.

Villeneuve, D., Desrosiers, J., Lübbecke, M. E. and Soumis, F. (2005). On compact formulations for integer programs solved by column generation, *Annals of Operations Research* **139**(1): 375–388.

Vis, I. F. A. and de Koster, R. (2003). Transshipment of containers at a container terminal: An overview, *European Journal of Operational Research* **147**(1): 1–16.

Walker, W. (1969). A method for obtaining the optimal dual solution to a linear program using the Dantzig-Wolfe decomposition, *Operations Research* **17**: 368–370.

Xu, J. and Chiu, S. Y. (2001). Effective heuristic procedures for a field technician scheduling problem, *Journal of Heuristics* **7**(5): 495–509.

Yang, J. H. and Kim, K. H. (2006). A grouped storage method for minimizing relocations in block stacking systems, *Journal of Intelligent Manufacturing* **17**: 453–463.

Zhang, C., Liu, J., Wan, Y. W., Murty, K. G. and Linn, R. (2003). Storage space allocation in container terminals, *Transportation Research Part B* **37**: 883–903.

Zhang, C., Wan, Y. W., Liu, J. and Linn, R. (2002). Dynamic crane deployment in container storage yards, *Transportation Research Part B* **36**: 537–555.

Curriculum vitae

Ilaria Vacca

Education

02/2007-02/2011 **PhD in Mathematics**
École Polytechnique Fédérale de Lausanne (EPFL), Switzerland.
Thesis: *Container terminal management: integrated models and large-scale optimization algorithms.*

10/2004-10/2006 **Mathematical Engineering, M.Sc.**
Università degli Studi di Roma Tor Vergata, Italy.
Thesis: *Mass Marketing Optimization in B2B Environments.*

10/2001-10/2004 **Management and Industrial Engineering, B.Sc.**
Università degli Studi di Roma Tor Vergata, Italy.
Thesis: *The Min Sum Vertex Problem On Tree Graphs.*

Professional Experience

02/2007-present **Transport and Mobility Laboratory, EPFL.**
Research and Teaching Assistant.

02/2006-07/2006 **Business Optimization Group, IBM Research, Zurich.**
Intern. The project resulted in my Master thesis.

Publications

Giallombardo G, Moccia L, Salani M and Vacca I (2010)
Modeling and solving the The Tactical Berth Allocation Problem. Transportation Research Part B: Methodological, 44(2) 232-245.

Salani M, Vacca I and Bierlaire M (2010)
Two-stage column generation. Technical report TRANSP-OR 101130. Transport and Mobility Laboratory, EPFL.

Vacca I, Salani M and Bierlaire M (2010)
Optimization of operations in container terminals: hierarchical vs

integrated approaches. Proceedings of the 10th Swiss Transport Research Conference, Sept 1-3, 2010.

Salani M and Vacca I (2009)
Branch and Price for the Vehicle Routing Problem with Discrete Split Deliveries and Time Windows. Technical report TRANSP-OR 091224. Transport and Mobility Laboratory, EPFL.

Vacca I, Bierlaire M and Salani M (2007)
Optimization at Container Terminals: Status, Trends and Perspectives. Proceedings of the 7th Swiss Transport Research Conference, Sept 12-14, 2007.

Project supervision

Spring 2008	Arnaud Vandaele: *Measures of congestion in container terminals* (Master thesis, received the ORBEL award 2009)
Spring 2009	Atoosa Kasirzadeh: *Simulation and optimization in container terminals* (Master thesis)
Spring 2010	Luca Furrer: *The Tactical Berth Allocation Problem: hierarchical vs integrated models in the context of container terminal operations* (Semester project)

Teaching experience

2008,2009	Undergraduate course on Operations Research, EPFL (TA)
2007,2008,2009	Graduate course on Differential Optimization, EPFL (TA)
2010	Doctoral course on Simulation and Optimization, EPFL (TA)

Awards

2007	"Sebastiano e Rita Raeli" graduation award, Università degli Studi di Roma Tor Vergata, Italy.
2010	Doctoral poster award - 2nd prize, ENAC Research Day 2010, Ecole Polytechnique Fédérale de Lausanne, Switzerland.

Computer skills

Programming	C++, Java, SQL
Software	CPLEX, GLPK, Matlab, R, IBM DB2, IBM Intelligent Miner

Languages

Italian (mother tongue), English (fluent), French (fluent).