Large-Scale Optimisation in Operations Management: Algorithms and Applications

Ioannis Fragkos

Department of Management Science and Innovation

University College London

A thesis submitted for the degree of

PhilosophiæDoctor (PhD) in Management Science and Innovation

2013 November

Declaration

I, Ioannis Fragkos, confirm that the work presented in this thesis is my own. Where information has been derived from other sources, I confirm that this has been indicated in the thesis.

Abstract

The main contributions of this dissertation are the design, development and application of optimisation methodology, models and algorithms for large-scale problems arising in Operations Management. The first chapter introduces constraint transformations and valid inequalities that enhance the performance of column generation and Lagrange relaxation. I establish theoretical connections with dual-space reduction techniques and develop a novel algorithm that combines Lagrange relaxation and column generation. This algorithm is embedded in a branch-and-price scheme, which combines large neighbourhood and local search to generate upper bounds. Computational experiments on capacitated lot sizing show significant improvements over existing methodologies. The second chapter introduces a Horizon-Decomposition approach that partitions the problem horizon in contiguous intervals. In this way, subproblems identical to the original problem but of smaller size are created. The size of the master problem and the subproblems are regulated via two scalar parameters, giving rise to a family of reformulations. I investigate the efficiency of alternative parameter configurations empirically. Computational experiments on capacitated lot sizing demonstrate superior performance against commercial solvers. Finally, extensions to generic mathematical programs are presented. The final chapter shows how large-scale optimisation methods can be applied to complex operational problems, and presents a modelling framework for scheduling the transhipment operations of the Noble Group, a global supply chain manager of energy products. I focus on coal operations, where coal is transported from mines to vessels using barges and floating cranes. Noble pay millions of dollars in penalties for delays, and for additional resources hired to minimize the impact of delays. A combination of column generation and dedicated heuristics reduces the cost of penalties and additional resources, and improves the efficiency of the operations. Noble currently use the developed framework, and report significant savings attributed to it.

To my friends and family for their invaluable trust throughout the years

Acknowledgements

I would like to acknowledge the help and support of my supervisors, Bert De Reyck and Zeger Degraeve who contributed the most to my academic progress and development.

Contents

List of Figures

List of Tables

1

Introduction

This thesis consists of three essays that contribute to the methodology, development and applications of large-scale optimisation theory. The focus is on Dantzig-Wolfe decomposition, a well-established reformulation technique that solves large-scale problems. The essays improve state of the art solution methodologies, generalise the decomposition idea so that it is applicable to a wider class of problems, and present computational implementations in a theoretical and a practical context.

The first essay describes methodologies that accelerate two ubiquitous decomposition techniques: column generation and Lagrange relaxation. In particular, I show how (i) the addition of valid inequalities that cut off multiple solutions on the primal space, and (ii) the dual space reduction achieved by a reformulation of the primal space, lead to important efficiency gains. Moreover, I develop a new hybrid column generation and Lagrange relaxation algorithm that combines the advantages of both methods to derive superior performance. The algorithm's key benefit is that it does not suffer from the degeneracy issues that plague simplex-based algorithms. The essay uses capacitated lot sizing as a test-bed to validate the proposed methodology. Finally, the hybrid lower-bounding scheme is combined with a novel branch-and-price implementation, in which the primal space search is directed by a large-scale neighbourhood search heuristic, while three local-search heuristics find upper bounds in each node. Computational experiments show significant improvements over existing methodologies.

The second essay introduces Horizon Decomposition, a novel generic way of applying Dantzig-Wolfe decomposition. The traditional application of Dantzig-Wolfe decomposition postulates a decomposable structure of a problem: by not considering a set of

complicating constraints, the problem is decomposed into a series of smaller-size problems. Horizon Decomposition introduces new constraints and variables in such a way that the problem achieves a decomposable structure. For problems that span over a horizon, this technique provides a family of reformulations. Each member of this family is characterised by the size of each subproblem and the overlap between pairs of consecutive subproblems. The essay demonstrates computationally, using capacitated lot-sizing as a test problem, (i) which configurations of horizon decomposition perform well and in which type of problems; (ii) how strong the obtained lower bound is and (iii) what the performance of a horizon-based branch-and-price algorithm is. The developed algorithm outperforms all state-of-the-art approaches with which it is benchmarked, both heuristic and exact. The generalisation of Horizon Decomposition opens interesting avenues for alternative implementations of Dantzing-Wolfe reformulations, in ways that have not been applied to date.

The third essay describes a modeling framework for the logistics operations of the Noble Group. The Noble Group is one of the biggest players in commodity trading worldwide. The operations include multiple transshipment levels, such as transportation of coal from mines to jetties, to river barges, and to large ocean vessels. Noble incurs penalties for late deliveries, that cost millions of dollars. Drawing upon the methodology developed in the two previous essays, we use Dantzig-Wolfe decomposition to develop a lower-bounding technique that verifies solution quality, and a fast dedicated algorithm that finds high quality feasible solutions. The dedicated algorithm has three main building blocks: (i) a knapsack-based barge allocation model, (ii) a single-vessel decomposition procedure and scheduling algorithm and (iii) a local search procedure that iteratively improves the generated schedules. Noble currently use this framework and report significant savings, resulting from reduced penalty costs and fewer spot infrastructure hired than before.

Overall, the aim of this dissertation is to contribute to the methodology, development and application of large-scale optimisation, within the field of operations management. The advent of information technology has drastically changed the landscape of operations management, in a way that problems of larger size need to be tackled. I hope that this dissertation will add to the growing stream of research that addresses such applications, and inspires other researchers working on large-scale optimisation methodology.

2

Period Decompositions For The Capacity Constrained Lot Size Problem With Setup Times

We study the Capacity Constrained Lot Size Problem with Setup Times (CLST). Based on two strong reformulations of the problem, we present a transformed reformulation and valid inequalities that speed up column generation and Lagrange relaxation. We demonstrate computationally how both ideas enhance the performance of our algorithm and show theoretically how they are related to dual space reduction techniques. We compare several solution methods, and propose a new efficient hybrid scheme that combines column generation and Lagrange relaxation in a novel way. Computational experiments show that the proposed solution method for finding lower bounds is competitive with other state-of-the-art approaches found in the literature. Finally, a branch-and-price based heuristic is designed and computational results are reported. The heuristic scheme compares favorably or outperforms similar approaches.

2.1 Introduction

The Capacity Constrained Lot Size Problem with Setup Times (CLST) is a well-known extension of the classical single-item uncapacitated Wagner-Whitin problem. The seminal paper of Trigeiro et al. (1989) was the first to introduce CLST and demonstrate that it is considerably harder than similar problems without setup times. In their study, they

proposed a per item Lagrange relaxation and designed a smoothing heuristic (TTM) that was able to find good feasible solutions quickly. It is worth noticing that many heuristics approaches have been developed (Jans and Degraeve 2007), but most of them do not provide a lower bound and therefore the solution quality cannot be assessed directly. The aim of this chapter is to design a fast heuristic for CLST that provides good solutions and a strong lower bound used to assess the solution quality. Per period Danzig-Wolfe decompositions of two network reformulations of CLST are proposed. The main advantage of the proposed decompositions is that they provide a lower bound which is stronger than these of the standard and network formulations. The potential downside is their computational tractability: when computed with column generation, the already large number of variables of the network formulations and their inherent degeneracy could lead to long solution times for the restricted master problem, and only minor bound improvements per iteration. Although there exists limited computational experience with decompositions of extended formulations on this problem, the computational experiments of Jans and Degraeve (2004) suggest that simplex-based solvers do not exhibit good convergence behavior. In particular, they tend to require a large number of iterations to converge, while the improvement of the lower bound in the last iterations is marginal, creating a well-known tailing-off effect (Vanderbeck, 2006). We propose a novel, considerably faster subgradient-based hybrid scheme that combines Lagrange relaxation and column generation. This scheme gives valid lower bounds of excellent quality and outperforms pure simplex-based column generation, Lagrange relaxation and subgradient-based column generation (in which the restricted master problems are solved with subgradient optimization). Further, we enhance the performance of our algorithm by utilizing two new dual space reduction techniques. First, we show how a primal space reformulation can lead to improved performance of dual based algorithms during column generation. Second, we employ a class of valid inequalities and show how this corresponds to adding dual optimal inequalities in the dual space of the restricted master problem (Ben Amor et al. 2007). The new subproblems remain tractable and the performance of the hybrid column generation scheme is improved further. The new hybrid scheme is embedded in a heuristic branch-and-price framework, designed specifically to obtain good feasible solutions fast. To achieve this, we recover a primal solution of the restricted master problem using the volume algorithm of Barahona and Anbil (2000) and branch on the resulting fractional setup variables.

Moreover, we integrate in a customized fashion recent MIP-based heuristic approaches, such as relaxation induced neighborhoods and selective dives (Danna et al. 2005), with established ones such as the forward/backward smoothing heuristic of Trigeiro et al. (1989). Extensive computational experiments show that the branch-and-price heuristic performs very well against other competitive approaches. The remainder of this chapter is organised as follows. Section 2 provides a brief literature review. Section 3 describes CLST formulations, and Section 4 their Dantzig-Wolfe decompositions. Section 5 describes customized procedures for solving the subproblems. Section 6 describes the hybrid scheme and Section 7 the branch-and-price heuristic. Finally, Section 8 presents computational results and Section 9 concludes with comments and directions for future research.

2.2 Literature Review

The literature on capacity constrained lot size problems is vast. A broad categorization would distinguish two research streams: theoretical and computational. Theoretical results include mainly complexity proofs, polyhedral studies and approximation algorithms and are not related directly to the scope of this work. Computational studies address the development and computational assessment of exact and heuristic approaches. With respect to the more recent research on exact approaches, Van Vyve and Wolsey (2006) suggest an extended formulation based on the network reformulation of Eppen and Martin (1987) that uses a single parameter to control the trade off between the number of variables and the lower bound strength. They show that selecting small values of that parameter is sufficient to solve hard problems, especially when a redundant row that facilitates the solver to generate cuts is added in the formulation. Degraeve and Jans (2007) discuss the structural deficiency of the decomposition proposed by Manne (1958) which only allows the computation of a lower bound for the problem, show the correct implementation of the Dantzig-Wolfe decomposition principle for CLST and develop a branch-and-price algorithm. Pimentel et al. (2010) propose three Dantzig-Wolfe decompositions of the standard formulation of CLST, and compare the performance of the corresponding branch-and-price algorithms. Specifically, they describe and compare the per-item, per-period and simultaneous per-item and

per-period decompositions. Belvaux and Wolsey (2000) developed BC-PROD, a specialized branch-and-cut system for generic lot size problems. Some of the cornerstone work that is less recent are the variable redefinition approach of Eppen and Martin (1987), the use of valid inequalities (Barany et al. 1984) and the simple plant location reformulation of Krarup and Bilde (1977). Recently, many authors (Alfieri et al. 2002, Pochet and Van Vyve 2004, Denizel et al. 2008 and Süral et al. 2009) have used such alternative formulations for the CLST with stronger linear relaxations. The seminal paper of Trigeiro et al. (1989) proposed a per-item Lagrange relaxation and designed a smoothing heuristic (TTM) that was able to find good feasible solutions quickly. Süral et al. (2009) designed a Lagrange relaxation based heuristic that outperforms TTM for problems without setup costs. Other recent heuristic approaches include among others, the cross entropy-Lagrange hybrid algorithm of Caserta and Rico (2009), the adaptive large neighbourhood search algorithm of Müller et al. (2012), the LP-based heuristic and curtailed branch-and-bound of Denizel and Süral (2006) and the iterative production estimate (IPE) heuristic of Pochet and Van Vyve (2004). Regarding metaheuristic approaches, a good review can be found in Jans and Degraeve (2007). Relevant to the current work is also the decomposition approach proposed in Jans and Degraeve (2004). The authors propose a per-period decomposition of a strong reformulation proposed in Eppen and Martin (1987). They obtain improved lower bounds, but their computational experiments show that standard computation schemes may be very time consuming for hard problems. Specifically, their computational results for the instances by Trigeiro et al. (1989) show that they need more than 11 minutes to compute the root node lower bound for G69, while their algorithm fails to converge within 1 hour for the larger instances G57 and G72. To put these results in perspective, the same authors compute in less than 4 seconds the root node lower bounds of the same instances, by using an item decomposition of the regular CLST formulation, which does not pose the convergence challenges of the period decomposition of network formulations (Degraeve and Jans, 2007). The main contributions of the present work are (a) the development and comparison of two period Dantzig-Wolfe network reformulations for CLST; (b) the development of a methodology that circumvents the computational difficulties of extended formulations through the design of a stabilization algorithm, and the use of problem-specific inequalities in the customized algorithm that solves the subproblems; (c) the development of a state-of-the-art branch-and-price heuristic that integrates and

customizes several recent advances such as the volume algorithm (Barahona and Anbil 2000, 2002), relaxation induced neighbourhoods and selected dives (Danna et al. 2005) and finally (d) the presentation of computational results that suggest the competitiveness of the proposed scheme against other approaches. The next section gives an overview of several formulations that are used throughout the chapter.

2.3 Formulations for the Capacity Constrained Lot Size Problem

In this section we present different formulations for the capacity constrained lot size problem: the regular formulation (CL); the shortest path formulation (SP) proposed in Eppen and Martin (1987); a transformed shortest path formulation (SPt); the facility location formulation (FL) studied in Krarup and Bilde (1977); and the facility location formulation with precedence constraints (FLp).

2.3.1 The Regular Formulation (CL)

The regular formulation for the Capacity Constrained Lot Size Problem with Setup Times is described by the following sets, parameters and decisions variables (Trigeiro et al. 1987, Degraeve and Jans, 2007).

Sets

$I = \{1, ..., n\}$: Set of items, indexed by i.
$T = \{1, ..., m\}$: Set of periods, indexed by t.

Parameters

d_{it}: demand of item i in period t, $\forall i \in I, \forall t \in T$.
sd_{itk}: sum of demand of item i from period t till period k, $\forall i \in I, \forall t, k \in T : t \leq k$.
hc_{it}: cost of holding inventory for item i from period $t-1$ to period t, $\forall i \in I, \forall t \in T$.
sc_{it}: setup cost of item i in period t, $\forall i \in I, \forall t \in T$.
vc_{it}: production cost of item i in period t, $\forall i \in I, \forall t \in T$.
st_{it}: setup time of item i in period t, $\forall i \in I, \forall t \in T$.
vt_{it}: variable production time of item i in period t, $\forall i \in I, \forall t \in T$.
M_{it}: big-M quantity, defined as $M_{it} = \min\{sd_{itm}, \frac{cap_t - st_{it}}{vt_{it}}\}, \forall i \in I, \forall t \in T$.

cap_t: time capacity in period t, $\forall t \in T$.

Decision Variables

x_{it}: production quantity of item i in period t, $\forall i \in I, \forall t \in T$.
s_{it}: inventory quantity of item i at the beginning of period t, $\forall i \in I, \forall t \in T \cup \{m+1\}$.
y_{it}: =1 if setup for item i in period t, =0 otherwise, $\forall i \in I, \forall t \in T$.

The mathematical formulation of CLST is then as follows:

$$\min \quad \sum_{i\in I}\sum_{t\in T} sc_{it}y_{it} + \sum_{i\in I}\sum_{t\in T} vc_{it}x_{it} + \sum_{i\in I}\sum_{t\in T} hc_{it}s_{it} \tag{2.1}$$

$$\text{s.t.} \quad s_{it} + x_{it} = d_{it} + s_{i,t+1} \qquad \forall i \in I, \forall t \in T \tag{2.2}$$

$$x_{it} \leq M_{it}y_{it} \qquad \forall i \in I, \forall t \in T \tag{2.3}$$

$$\sum_{i\in I} st_{it}y_{it} + \sum_{i\in I} vt_{it}x_{it} \leq cap_t \qquad \forall t \in T \tag{2.4}$$

$$x_{it}, s_{it} \geq 0, s_{i,m+1} = 0, y_{it} \in \{0,1\} \quad \forall i \in I, \forall t \in T \tag{2.5}$$

The objective function (2.1) minimizes the setup cost, the variable production cost, the inventory holding cost and initial inventory cost. Constraints (2.2) are the demand constraints: inventory carried over from the previous period and production in the current period can be used to satisfy current demand and build up inventory. As in Vanderbeck (1998), we deal with possible infeasible problems by allowing for initial inventory (s_{i0}) which is available in the first period at a large cost, fc_i. There is no setup required for initial inventory. Constraint (2.3) forces the setup variable to one if any production takes place in that period. Next, there is a constraint on the available capacity in each period (2.4). Finally, we have the non-negativity and integrality constraints (2.5) and the ending inventory is set to zero.

2.3.2 The Shortest Path Formulation (SP)

Next, model (2.1)-(2.5) is reformulated using the variable redefinition approach of Eppen and Martin (1987).
Define the following parameters:
cv_{itk}: total production and holding cost for producing item i in period t to satisfy demand for the periods t until k, $cv_{itk} = vc_{it}sd_{itk} + \sum_{s=t+1}^{k}\sum_{u=t}^{s+1} hc_{iu}d_{is}$,

ci_{it}: total production and holding cost for initial inventory for item i to satisfy demand from period 1 up to period t, $ci_{it} = fc_i sd_{i1t} + \sum_{s=2}^{t} \sum_{u=1}^{s-1} hc_{iu} d_{is}$.
We also define the following new variables:
z_{itk}: fraction of the production plan for item i where production in period t satisfies demand from period t to period k, $x_{it} = \sum_{k=t}^{m} sd_{itk} z_{itk}, \forall i \in I, \forall t \in T$.
p_{it}: fraction of the initial inventory plan for item i where demand is satisfied for the first t periods, $s_{i0} = \sum_{t=1}^{m} sd_{i1t} p_{it}, \forall i \in I$.
The shortest path formulation (SP) is then as follows:

$$\min \quad \sum_{i\in I}\sum_{t\in T}(sc_{it}y_{it} + ci_{it}p_{it}) + \sum_{i\in I}\sum_{t\in T}\sum_{k=t}^{m} cv_{itk}z_{itk} \tag{2.6}$$

$$\text{s.t.} \quad \sum_{k=1}^{m}(z_{i1k} + p_{ik}) = 1 \qquad \forall i \in I \tag{2.7}$$

$$\sum_{j=1}^{t-1} z_{ijt-1} + p_{it-1} = \sum_{k=t}^{m} z_{itk} \qquad \forall i \in I, \forall t \in T\backslash\{1\} \tag{2.8}$$

$$\sum_{i\in I} st_{it}y_{it} + \sum_{i\in I}\sum_{k=t}^{m} vt_{it}sd_{itk}z_{itk} \leq cap_t \qquad \forall t \in T \tag{2.9}$$

$$\sum_{k=t}^{m} z_{itk} \leq y_{it} \qquad \forall i \in I, \forall t \in T \tag{2.10}$$

$$y_{it} \in \{0,1\}, p_{it} \geq 0 \qquad \forall i \in I, \forall t \in T \tag{2.11}$$

$$z_{itk} \geq 0 \qquad \forall i \in I, \forall t, k \in T : k \geq t \tag{2.12}$$

Formulation (2.6)–(2.12) is based on a graph representation of the CLST. Specifically, an acyclic graph is defined for each item $i \in I$, where each node corresponds to a period $t \in \{0 \dots |T|\}$. Nodes $(0,i), \forall i \in I$, receive an inflow of 1 unit, which is then distributed through the entire network of each item $i \in I$. Production arcs that link nodes (i,t_1) and (i,t_2) represent the fraction of demand of periods $\{t_1+1 \dots t_2\}$ that is covered by production in period t_1+1. Nodes $(0,i)$ have an additional type of arcs that model initial inventory. Each initial inventory arc links node $(i,0)$ with node $(i,t), \forall t \in T$, and represents the fraction of demand of periods $\{1 \dots t\}$ that is satisfied by initial inventory. Figure 2.1 shows a single item, four period example.

The objective function (2.6) minimizes the total costs. Constraints (2.7) and (2.8) define the flow balance constraints of each node (i,t), which ensure that demand is sat-

isfied. For each item, a unit flow is sent through the network, imposing that its demand has to be satisfied without backlogging. The capacity constraints (2.9) limit the sum of the total setup times and production times to the available capacity in each period. Constraint (2.10) defines the setup forcing for each item. Finally, setup decisions are binary (2.11), initial inventory non-negative and the shortest path variables z_{itk} (2.12) non-negative as well.

2.3.3 Transformed Shortest Path Formulation (SPt)

We transform formulation SP to an equivalent formulation, which leads to a better convergence of the Lagrange multipliers due to dual space reduction, as explained below. The idea is to substitute, for each item, the demand balance constraint of period t with the sum of the demand balance constraints of the first t periods. Doing so, constraints (2.7) and (2.8) are replaced with:

$$\sum_{k=t}^{m} p_{ik} + \sum_{j=1}^{t}\sum_{k=t}^{m} z_{ijk} = 1 \qquad \forall i \in I, \forall t \in T \tag{2.13}$$

We denote the resulting transformed formulation SPt. Eppen and Martin (1987) showed that their network reformulation corresponds to a shortest path problem defined on an acyclic graph. The demand constraints (2.7) and (2.8) express the flow balance in each node of this graph. Equation (2.13) imposes, for each item $i \in I$, that the flow of the cut-set defined by the $s-t$ cut $C_t = (\{0, ..., t-1\} \cup \{t, ..., m\})$ is equal to one. Figure 2.1 illustrates a single-item four period example with no initial inventory. Equation (2.8) for period 3 reads $z_{12} + z_{22} = z_{33} + z_{34}$, which corresponds to the flow balance of node 2. For the same period, (2.13) sets the total flow through the edges connecting the sets of nodes {0, 1, 2} and {3, 4} equal to one, i.e., $1 = z_{13} + z_{14} + z_{23} + z_{24} + z_{33} + z_{34}$.

The idea behind this transformation is that of *dual space reduction*. Let D be the feasible dual space associated with constraints (2.7) and (2.8), and D' the one associated with constraints (2.13). If v_{i1} and v_{it} are the dual values of (2.7) and (2.8) and $\bar{v}_{i1}$ and $\bar{v}_{it}$ the dual values of (2.13), then

$$D = \left\{ v_{it} \in \Re \;\middle|\; \begin{array}{c} v_{it} - v_{i,k+1} \leq cv_{itk}\ \forall i \in I, t, k \in T : t \leq k < m \\ v_{i1} - v_{it+1} \leq ci_{it}\ \forall i \in I, t \in T \backslash \{m\} \\ v_{it} \leq cv_{itm}\ \forall i \in I, t \in T \\ v_{i1} \leq ci_{im}\ \forall i \in I \end{array} \right\};$$

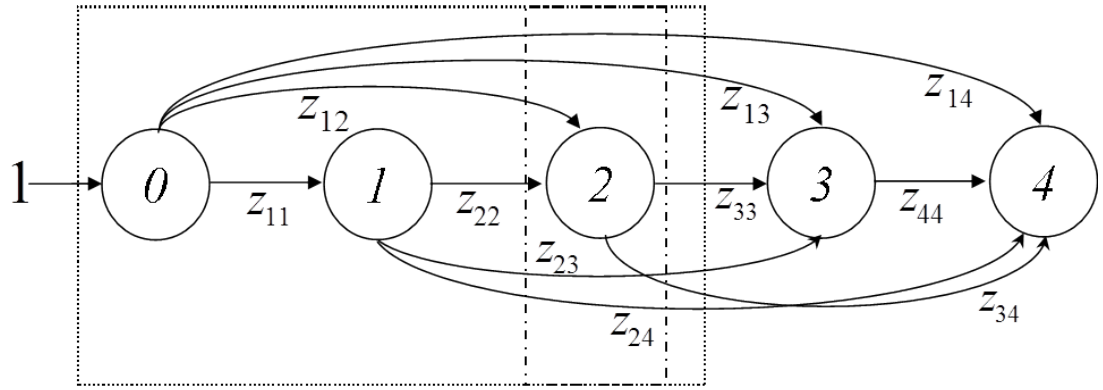


Figure 2.1: A single-item four period example with no initial inventory -

$$D' = \left\{ \bar{v}_{it} \in \Re \;\middle|\; \begin{array}{l} \sum_{l=1}^{t} \bar{v}_{il} \leq \min\{ci_{it}, cv_{i1t}\} \;\forall i \in I, t \in T \\ \sum_{l=t}^{k} \bar{v}_{il} \leq cv_{itk} \;\forall i \in I, t, k \in T : k \geq t \end{array} \right\}$$

Theorem 2.1 $D' \subseteq D$.

Proof See Appendix. ■

Transformed formulations like the SPt imply a denser form of the constraint matrix, which, in the context of the simplex method, might result in more pivots and therefore be less efficient. Since our algorithm works in the dual space, it is interesting to investigate computationally if the dual space reduction of subsystem (2.7) and (2.8) is beneficial.

2.3.4 Facility Location Formulation (FL)

CL can be reformulated using variables originally employed in Facility Location problems. Facility Location involves allocating facilities to a discrete set of locations such that the market demand of all locations is served and the joint facility setup and variable costs are minimized. To the best of our knowledge, the first paper that used facility location variables to formulate lot size models was the work of Krarup and Bilde (1977). The resulting model (FL) is described below.

Parameters:

cs_{itk}: total production and holding cost for producing item i in period t to satisfy demand of period k, $cs_{itk} = (vc_{it} + \sum_{u=t}^{k-1} hc_{iu})d_{ik}$

cu_{it}: initial inventory and holding cost for item i, to satisfy demand in period t, $cu_{it} = (fc_i + \sum_{u=1}^{t-1} hc_{iu})d_{it}$.

We also define the following variables:
w_{itk} :fraction of demand for item i in period k that is satisfied by production in period t, $x_{it} = \sum_{k=t}^{m} d_{ik} w_{itk}$,$\forall i \in I, \forall t \in T$,
sp_{it}:fraction of demand for item i in period k that is satisfied by initial inventory, $s_{i0} = \sum_{t=1}^{m} sp_{it} d_{it}, \forall i \in I$.
The facility location (FL) reformulation is then as follows:

$$\min \quad \sum_{i\in I}\sum_{t\in T}(sc_{it}y_{it} + cu_{it}sp_{it}) + \sum_{i\in I}\sum_{t\in T}\sum_{k=t}^{m} cs_{itk}w_{itk} \tag{2.14}$$

$$\text{s.t.} \quad sp_{it} + \sum_{k=1}^{t} w_{ikt} = 1 \qquad \forall i \in I, \forall t \in T \tag{2.15}$$

$$\sum_{i\in I} st_{it}y_{it} + \sum_{i\in I}\sum_{k=t}^{m} vt_{it}d_{ik}w_{itk} \leq cap_t \qquad \forall t \in T \tag{2.16}$$

$$w_{itl} \leq y_{it} \qquad \forall i \in I, \forall t, k \in T : k \geq t \tag{2.17}$$

$$y_{it} \in \{0,1\}, sp_{it} \geq 0 \qquad \forall i \in I, \forall t \in T \tag{2.18}$$

$$w_{itk} \geq 0 \qquad \forall i \in I, \forall t, k \in T : k \geq t \tag{2.19}$$

Formulation (2.14) – (2.19) corresponds to a facility location network for each item $i \in I$. Specifically, for each $i \in I$, each period $t \in T$ denotes a location with demand d_{it}, in which a facility can be built. A facility in location $t \in T$ can serve market demand of locations $t' \geq t$ only. Setup variables y_{it} denote for each item, $i \in I$, if a facility is built at location $t \in T$. Variables w_{itk} show the fraction of demand of location k that is satisfied by production of a facility in location t. The objective function (2.14) minimizes the total cost, which consists of the setup cost, the aggregated production and holding costs, and the initial inventory and holding costs. Equations (2.15) correspond to the flow balance constraints (2.2) and state that period t demand must be covered by a combination of initial inventory and production in periods $\{1, ..., t\}$. The capacity constraints (2.16) are in exact correspondence with (2.4). The setup constraints (2.17) do not allow any production in period t unless a setup is done and the non-negativity conditions (2.18) and (2.19) complete the FL formulation.

2.3.5 Facility Location Formulation with Precedence Constraints (FLp)

Many extended formulations are often degenerate or have multiple optimal solutions. This is also the case with the facility location formulation (2.14)-(2.19). The existence of multiple solutions in the extended formulation may degrade the efficiency of column generation. Hence, the addition of valid inequalities in the subproblem that cut off some of the primal space containing alternative optimal solutions, may lead to improved convergence, as it prevents the subproblem from generating columns that describe alternative solutions. A class of such valid inequalities is described below.

Observation 2.2 *There exists an optimal solution of FL with* $w_{it,k-1} \geq w_{itk}$ *for all* $i \in I, t, k \in T : t+1 \leq k \leq m$ *and* $sp_{it-1} \geq sp_{it}$ *for all* $i \in I$ *and* $t \in T \backslash \{1\}$.

We will refer to the above valid inequalities as *Precedence Constraints*. Observation 2.2 is used in Wolsey (1989) in his study of the facility location formulation in the context of lot size problems with start-up costs and no capacity constraints. A short proof can be found in the Appendix. FL is not a minimal image of $conv(CL)$, the convex hull of CL, in the sense that there exists a subset of $conv(FL)$ that is the image of all extreme points of $conv(CL)$. This is important because $conv(CL)$ might have more extreme points when the precedence constraints are not considered. The precedence constraints $w_{it,k-1} \geq w_{itk}$ can be used alongside the setup forcing $w_{itt} \leq y_{it}$ in place of $w_{itk} \leq y_{it}$ for all $i \in I, t, k \in T : t+1 \leq k$. The FL formulation with the primal valid inequalities is written as:

$$\min \quad \sum_{i \in I} \sum_{t \in T} (sc_{it} y_{it} + cu_{it} sp_{it}) + \sum_{i \in I} \sum_{t \in T} \sum_{k=t}^{m} cs_{itk} w_{itk} \tag{2.20}$$

$$\text{s.t.} \quad w_{itt} \leq y_{it} \qquad \forall i \in I, \forall t \in T \tag{2.21}$$

$$w_{it,k-1} \geq w_{itk} \qquad \forall i \in I, \forall t, k \in T : k > t \tag{2.22}$$

$$(2.15) - (2.16), (2.18) - (2.19) \tag{2.23}$$

The idea behind the inclusion of constraints (2.22) is that of primal space reduction. The idea behind the inclusion of constraints (2.22) is that of primal space reduction. Although it is usually the case that the inclusion of a set of constraints, such as (2.22), at the subproblem level improves the problem lower bound, we show that this does not hold in the context we examine. Rather, (2.22) accelerates the column generation

convergence because the feasible space of columns generated by the subproblem is reduced.

2.4 Period Dantzig-Wolfe Decompositions

2.4.1 Formulations

We develop period decompositions for formulations SP, SPt, FL and FLp. In a period decomposition, demand covering constraints of each item are considered complicated constraints. Therefore, by not taking into account the demand constraints, the problem decomposes in a series of problems, each defined over a single period $t, \forall t \in T$. We denote each decomposition formulation by appending /P to the original notation. For example, SP/P denotes the period decomposition of the shortest path formulation. The demand balance constraints are the complicating constraints and the capacity and setup forcing constraints of each period form the period subproblems. We focus on period decompositions of network formulations because the resulting relaxations provide improved lower bounds compared to the linear programming (LP) relaxation of the extended formulations, since the period capacity polytope does not have the integrality property (Geoffrion, 1974).

The formulation of the per period decomposition of SP, SP/P, is described in detail in Jans and Degraeve (2004). The formulation of the per period decomposition of SPt, SPt/P is very similar to SP/P and we skip it for brevity. Both formulations SP/P and SPt/P can be found in the Appendix. Instead, we present the per period decompositions of FL and FLp. To this end, let us define by S_t the index set of extreme point production plans of the subproblem for period t, i.e., $S_t := \left\{ q \in extr\left(conv\left\{ (w_{itk}, y_{it})_{i \in I, k \geq t} \mid (2.16) - (2.19) \right\} \right) \right\}$, where $extr(S)$ denotes the set of extreme points of set S. We associate a decision variable with the fraction of the extreme point q of subproblem t that is used in a feasible solution. If we denote by $(\bar{w}_{itkq}, \bar{y}_{itq})$ the components of extreme point q, its cost can be written as $ct_{tq} = \sum_{i \in I} (sc_{it} \bar{y}_{itq} + \sum_{k=t}^{m} cs_{itk} \bar{w}_{itkq})$. Then the master problem FL/P can be formulated as follows:

$$\min \quad \sum_{t \in T} \sum_{q \in S_t} ct_{tq} wt_{tq} + \sum_{i \in I} \sum_{t \in T} cu_{it} sp_{it} \tag{2.24}$$

$$\text{s.t.} \quad sp_{it} + \sum_{l=1}^{t} \sum_{q \in S_l} \bar{w}_{iltq} wt_{lq} = 1 \qquad \forall i \in I, \forall t \in T \qquad [\pi_{it}] \qquad (2.25)$$

$$\sum_{q \in S_t} wt_{tq} = 1 \qquad \forall t \in T \qquad [\mu_t] \qquad (2.26)$$

$$y_{it} = \sum_{q \in S_t} \bar{y}_{itq} wt_{tq} \qquad \forall i \in I, \forall t \in T \qquad (2.27)$$

$$wt_{tq} \geq 0, \; y_{it} \in \{0,1\} \qquad \forall i \in I, \forall q \in S_t, \forall t \in T \qquad (2.28)$$

The objective function (2.24) minimizes the total cost of the initial inventory and the cost of the production plans chosen in each period. Constraints (2.25) model demand and correspond to constraints (2.2) in the standard formulation. The convexity constraints (2.26) and the nonnegativity constraints (2.28) enforce a convex combination. The setup variables definition is given in (2.27). The integrality must be imposed on the original setup variables (2.28). The constraint coefficient parameters $(\bar{w}_{itkq}, \bar{y}_{itq})$ are defined by the subproblem extreme points. The subproblem objective function minimizes the reduced cost over the extreme points. Specifically, the period t subproblem (SUB) reads:

$$\min \quad \sum_{i \in I} sc_{it} y_{it} + \sum_{i \in I} \sum_{k=t}^{m} (cs_{itk} - \pi_{ik}) w_{itk} - \mu_t \qquad (2.29)$$

$$\text{s.t.} \quad \sum_{i \in I} st_{it} y_{it} + \sum_{i \in I} \sum_{k=t}^{m} vt_{it} d_{ik} w_{itk} \leq cap_t \qquad \forall t \in T \qquad (2.30)$$

$$w_{itk} \leq y_{it} \qquad \forall i \in I, \forall t, k \in T : k \geq t \qquad (2.31)$$

$$y_{it} \in \{0,1\}, w_{itk} \geq 0 \qquad \forall i \in I, \forall t, k \in T : k \geq t \qquad (2.32)$$

The period decomposition of the formulation FLp has the same master problem. However, the subproblem, denoted SUBp, is different because it includes the precedence constraints (2.21) and (2.22) in place of (2.31).

2.4.2 Lower Bounds

It is interesting to investigate the lower bound quality of the proposed decompositions. To this end, let $\bar{v}_{FL/P}$ be the optimal objective value of the linear relaxation of the decomposed model FL/P (2.24)-(2.28). Also, let $\bar{v}_{FLp/P}, \bar{v}_{SP/P}$ and $\bar{v}_{SPt/P}$ be the corresponding lower bounds obtained by the linear relaxation of FLp/P, SP/P, and

SPt/P respectively. To obtain a first result, we will need the following definition and lemma.

Definition 2.3 *(Ben Amor et al. 2007). Let D be the dual space polyhedron of a linear program and D^* be the set of its optimal solutions. Also, assume a new set of constraints $E^T\pi \leq \mathbf{e}$ that cuts off part of the dual space. If $D^* \subseteq \left\{\pi : E^T\pi \leq \mathbf{e}\right\}$, then $E^T\pi \leq \mathbf{e}$ is called a set of* dual-optimal inequalities*. If $E^T\pi \leq \mathbf{e}$ cuts off a nonempty set of dual-optimal solutions but is still satisfied by at least one dual-optimal solution, it is called a set of deep dual-optimal inequalities.*

Lemma 2.4 *Using subproblem SUBp is equivalent to using subproblem SUB and imposing constraints*

$$(cs_{itk} - \pi_{ik})\, d_{i,k+1} \geq (cs_{it,k+1} - \pi_{i,k+1})\, d_{ik} \;\; \forall i \in P, \;\; t,k \in T : t \leq k \leq m-1 \quad (2.33)$$

in the dual space of the LP master problem of FL/P. Moreover, these constraints are deep dual optimal inequalities.

Proof It suffices to show that SUB has the same optimal solution as SUBp whenever (2.33) holds. Note that vt_{it} can be added to both sides of the inequality, but is omitted since they cancel each other out. Therefore, (2.33) states that the profit to weight ratio of w_{itk} should be greater than that of $w_{it,k+1}$. It follows that there exists an optimal solution of the LP relaxation of SUB when (2.33) holds which is also optimal for the LP relaxation of SUBp. Also, SUB has the same structure after branching, so if (2.33) holds, the precedence constraints hold at any node of the branch-and-bound tree, and therefore at an integer optimal solution. Hence, from construction, (2.33) imply the precedence constraints, that do not cut-off all optimal solutions. Their equivalence with the precedence constraints and definition 1 imply that they are deep dual optimal inequalities. ■

Proposition 2.5

$$\bar{v}_{SP/P} = \bar{v}_{SPt/P} = \bar{v}_{FLp/P} = \bar{v}_{FL/P}$$

Proof First, note that the corresponding subproblems have the same set of extreme points, and the linking constraints of SPt/P are linear combinations of those of SP/P. Second, consider subproblem SUBp. The linear transformation $z_{itt} = w_{tt}$; $z_{itk} =$

$w_{itk} - w_{it,k-1}$, $k > t$ maps every extreme point (y_{it}, w_{itk}) of SUBp to a unique extreme point (y_{it}, z_{itk}) of the SP/P subproblem. Using this transformation in FLp/P, the resulting model is exactly SPt/P. On the other hand, every feasible solution of the SPt/P subproblem can be mapped to FLp/P with the inverse transformation, i.e., $w_{itt} = z_{tt}$; $w_{itk} = \sum_{l=t}^{k} z_{itl}$. Hence, $\bar{v}_{SPt/P} = \bar{v}_{FLp/P}$. Finally, we show that $\bar{v}_{FLp/P} = \bar{v}_{FL/P}$. To this end, notice that $\bar{v}_{FLp/P} \geq \bar{v}_{FL/P}$, since adding constraints to the subproblem can never lead to a worse bound. Denote by $\bar{v}'_{FL/P}$ the optimal value obtained when solving the dual of FL/P amended with the dual restrictions (lemma 2.4). Then $\bar{v}'_{FL/P} = \bar{v}_{FLp/P}$ from lemma 2.4 and $\bar{v}'_{FL/P} \leq \bar{v}_{FL/P}$, since adding cuts to the dual of a primal minimization problem can never increase its optimal value. The result follows.■

Interestingly, the above lower bound is stronger than the one obtained by the simultaneous per item and per period decomposition of formulation CL studied by Pimentel et al. (2010). Let us denote the latter lower bound by $\bar{v}_{CL/P/I}$.

Proposition 2.6

$$\bar{v}_{CL/P/I} \leq \bar{v}_{SP/P}$$

In addition, the above inequality can be strict.

Proof See Appendix. ■

2.5 Solving the Subproblems

This section describes two fast customized algorithms used to solve subproblems SUB and SUBp. Note that since the feasible solutions of FLp are in exact correspondence with those of SPt, and since the subproblem of SP and SPt is common, solving SUB and SUBp implies a solution for all four subproblems.

2.5.1 A customized algorithm for SUB

For SUB, the following simple observation plays a key role in the subsequent solution approach.

Observation 2.7 *In an optimal solution of the LP relaxation of SUB there exists some* $k_i \geq t$ *such that* $w_{itk_i} = y_{it}$, $\forall i \in I$.

This implies that the subset of tight inequalities (2.31) can be used to substitute out the y_{it} variables in (2.30). As a result, the LP relaxation of SUB reduces to a linear knapsack problem which admits a greedy solution, similar to the one proposed in Holmberg and Yuan (2000). It follows that an optimal solution of SUB has fractional continuous variables for at most one item. The following algorithm identifies the subset of tight inequalities in phase I and solves a regular linear knapsack problem in phase II.

Algorithm 1 Algorithm for Subproblem SUB

Phase I

Initialize $r_{itk} \leftarrow \frac{cs_{itk}-\pi_{ik}}{vt_{it}d_{ik}}, \forall i \in I, \forall t, k \in T : t \geq k$

Initialize $K_i = \emptyset, P = \{t, ..., m\}, Best_i = False, \forall i \in I$

1: **for** $i \in I$ **do**
2: **while** $Best_i = False$ **do**
3: $r_{it} \leftarrow \min_{k \in P \setminus K_i} \{r_{itk}\}$; $k_i \leftarrow \arg r_{it}$ \\ Calculate min ratio
4: $K_i \leftarrow K_i \cup \{k_i\}$
5: $r_{it} \leftarrow \frac{sc_{it}+\sum_{k \in K_i}(cs_{itk}-\pi_{ik})}{st_{it}+\sum_{k \in K_i} vt_{it}d_{ik}}$ \\ Update ratio of merged periods
6: **if** $r_{it} > \min_{k \in P \setminus K_i} \{r_{itk}\}$ or $K_i = P$ **then** \\ If ratio is min, exit
7: $Best_i = True$
8: **end if**
9: **end while**
10: **end for**

Phase II

11: $csm_{it} \coloneqq sc_{it} + \sum_{k \in K_i}(cs_{itk} - \pi_{ik})$
12: $vtm_{it} \coloneqq st_{it} + \sum_{k \in K_i} vt_{it}d_{ik}$
13: **Solve** a linear Knapsack with weights $\{ vtm_{it};\ vt_{it}d_{ik}\ \forall k \notin K_i, i \in I \}$ and costs $\{ csm_{it};\ cs_{itk} \forall k \notin K_i, i \in I \}$. Let $\{ wm_{it};\ w_{itk}\ \forall k \notin K_i, i \in I \}$ denote the optimal solution.
14: $y_{it} \leftarrow wm_{it}, \forall k \in K_i, i \in I$
15: **Return** $(y_{it}, w_{itk})\ \forall i \in I, \forall k \geq t$

Phase I identifies which of the variable upper bound constraints (2.31) are satisfied as equalities. Then, the value of the setup variable is set equal to the value of the production variable with the most attractive cost-to-weight ratio. If the new ratio is still the most attractive after the substitution, *Phase I* terminates. If it is not, another

constraint (2.31) is tight for the same item, and the corresponding substitution is made. *Phase II* solves a linear knapsack problem where all w_{itk} with $k \in K_i$ are equated with y_{it} and substituted out. Note that adjusting the algorithm to tackle items with zero setup time or cost is straightforward. Due to observation (2.7), algorithm 1 produces an optimal solution of the SUB linear relaxation.

After solving the relaxation, a depth-first branch-and-bound algorithm is implemented. Branching is needed only when some wm_{it}, the variable that indicates the used fraction of the merged periods, was the last variable to enter the knapsack at fractional level, and therefore $y_{it} < 1$. Thus, at most one setup variable is fractional. In addition, branching does not change the problem structure because it either fixes w_{itk} to zero when $y_{it} = 0$ or reduces the problem capacity by st_{it} and fixes sc_{it} in the objective, when $y_{it} = 1$. For this reason, algorithm SUB can be called at each node of the branch-and-bound tree, with different input data. Computational experiments confirm the superiority of this approach compared to simplex-based branch-and-bound.

2.5.2 A customized algorithm for SUBp

The addition of the precedence constraints changes the structure of subproblem SUB and the previous algorithm cannot be employed. However, we can take advantage of the fact that each solution of SUBp corresponds to a solution of the SP/P subproblem, whose relaxation is a linear multiple choice knapsack problem (Jans and Degraeve 2004). Therefore, we solve the latter subproblem and then map back the solution to SUBp. A customized depth-first branch-and-bound algorithm is also developed for this subproblem. The difference with the usual multiple choice knapsack algorithms is that when some item is branched to 1, the dominated periods change (Pisinger 1995). We use an efficient variant of the algorithm suggested by Graham (1972) to construct and update the convex hulls, and then solve the linear relaxations using the algorithm of Sinha and Zoltners (1979).

2.6 Solving the Master Problem: a new Hybrid Algorithm

In many practical cases the convergence of column generation can be slow as a tailing-off effect is observed (Vanderbeck and Savelsbergh 2006). This behavior is mainly due

to high degeneracy of the restricted master problem. Specifically, when the restricted master linear programs are solved with a simplex algorithm, two problems may arise. First, the solution time can be large, and second, the optimal dual values that are passed to the subproblems may be of bad quality. The latter issue stems from the fact that degeneracy implies multiple dual optimal solutions, and simplex-based solvers typically return an optimal extreme point of the dual polyhedron. Such corner points may not provide an accurate representation of the optimal face, whereas a point that lies within the optimal face may lead to faster convergence (Lübbecke and Desrosiers 2005 and Mitchell 2005). To tackle this problem, some authors have proposed to solve the linear programs with an interior point method, or to employ stabilization techniques, such as those described in du Merle et al. (1999), Ben Amor et al. (2005), Elhallaui et al. (2005) and Gondzio et al. (2013). Recently, Subramanian and Sherali (2008) gave insights in the way that poor dual values result in the generation of poor quality columns, and designed a subgradient-based scheme to solve large set-partitioning problems arising in aircraft crew scheduling. Columns coming from SUB or SUBp can have some fractional components. Therefore, the row aggregation technique of Elhallaui et al. (2005) cannot be used, since it is developed for pure set-partitioning problems.

Huisman et al. (2005) discuss how the relationship between Dantzig-Wolfe decomposition and Lagrange relaxation can lead to the development of improved algorithms that combine the strengths of both methods. They explore two different ways to combine column generation and Lagrange relaxation. First, Lagrange relaxation can be used to solve the master problem, by dualising the linking constraints, and the resulting near-optimal dual values can be used to price out the subproblems. Second, Lagrange relaxation can be employed within column generation, exploiting the fact that both methods solve the same subproblem. Specifically, the restricted master problem are solved with the simplex method, and next the master problem dual prices are used as a starting point for a number of Lagrange relaxation iterations, during which several negative reduced cost columns can be found and added to the master problem at once. We propose a new method, which is a combination of the two previously discussed methods. Specifically, it uses Lagrange relaxation to solve the master problems, but also to generate new columns. Figure 2.2 gives a schematic representation of the algorithm.

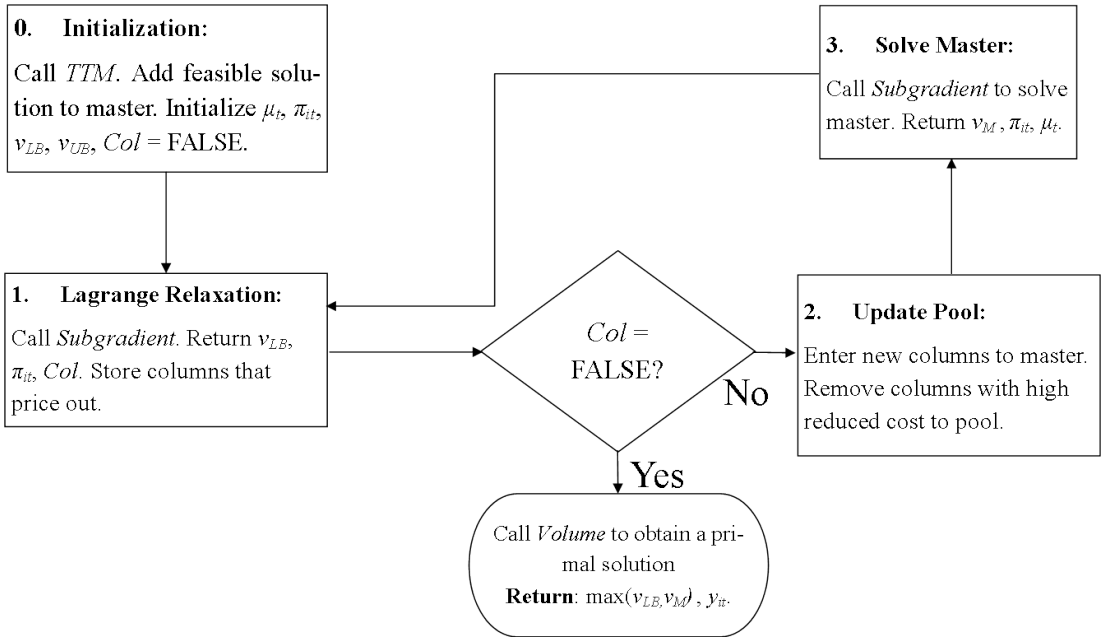


Figure 2.2: The main building blocks of the new hybrid algorithm -

We initialize the column generation process using the heuristic of Trigeiro et al. (1989). This heuristic returns a feasible solution, a lower bound and a set of Lagrange dual prices of the capacity constraints. Using the dual of the relaxation of SP or FL, we can then calculate the starting dual prices π_{it}. Also, the feasible solution $(y_{it}, x_{it})^F$ is split into per period columns of the SPt or FLp master problems. To perform this operation, it is necessary to transform the $(y_{it}, x_{it})^F$ solution to the space of the extended formulation variables. This can be done by fixing the setup variables to their given values and solving the corresponding formulation, which is a shortest path problem. Next, the Lagrange relaxation subroutine uses the modified subgradient method of Crowder (1976) to update the dual prices π_{it}. Moreover, for each updated set of dual prices we check if the subproblem solutions price out, using the last obtained dual prices of the convexity constraints, μ_t. The columns that price out are added to the master problem. Existing columns with a high reduced cost are removed from the master problem and kept into a separate column pool. Then, the restricted master problem is solved using subgradient optimization by relaxing the linking constraints (2.25) in the master problem (Huisman et al. 2005). The scheme iterates until no column prices out, where we invoke the volume algorithm to obtain an approximate

primal solution. In this way, we are able to make better informed branching decisions in the subsequent branch-and-price phase.

Contrary to existing hybrid algorithms (Barahona and Jensen, 1998, Degraeve and Peeters 2003, Degraeve and Jans 2007), this scheme relies exclusively on Lagrange relaxation. It has the benefits that it (a) avoids the degeneracy issues of the simplex method and therefore exhibits faster convergence and (b) returns good quality dual prices that help the column generation convergence. Note that the intermediate value of the master problems, v_M , is not necessarily a valid upper bound of the column generation lower bound, as it is calculated using lagrange relaxation. It is a valid lower bound (Huisman et al. 2005) when no column prices out, and therefore the scheme returns the best bound that is obtained by the master problem and by Lagrange relaxation.

2.7 A Branch-and-Price Heuristic

We have combined the hybrid column generation algorithm with heuristic techniques, and integrated them in an enumeration scheme. Our implementation uses formulation FLp to calculate the lower bounds. The goal is to find good feasible solutions fast by exploiting the structure of the network formulation and its subproblem. It is worth noting that the development of an exact approach is possible, but the nature of the lower bounding process would make it inefficient. For an exact approach, branching is needed even when the approximate primal solution returned by the volume algorithm has all setup variables integral, because the exact primal solution of the restricted master problem may still be fractional. Therefore, nodes can be implicitly enumerated only by dominance and infeasibility pruning, and the resulting trees are large. Since a heuristic search is performed in each node and the branching decisions dictate the search space in the tree, the overall scheme can be seen as integration of exploration and exploitation heuristics. The next section describes the heuristics employed in each node of the tree.

2.7.1 Node Heuristics

We employ a successive rounding heuristic that uses a smoothing subroutine, which in turn employs some operations of the heuristic of Trigeiro et al. (1989). The successive

rounding heuristic gets as input the primal setup variables produced by the volume algorithm, determines a threshold level and fixes the setup variables below and above that threshold to 0 and 1, respectively. The process iterates for increasing threshold values. Typically, feasible solutions obtained in this way follow a U-shaped fashion (Degraeve and Jans 2007). Therefore, when the solution quality starts to degrade the heuristic terminates. Starting from this solution, the smoothing heuristic that searches for an improved feasible solution is applied. The smoothing part starts from the last period and tries to push production backwards, such that the Lagrange costs are minimized. This is done iteratively until the first period, and if any capacity constraints are violated it performs a similar forward operation to recover feasibility. Thus, the rounding/smoothing heuristic scheme searches the feasible space by modifying incrementally the proposed setup schedules and production plans and can be considered an exploitation heuristic.

2.7.2 Node Lower Bounds

At each node we solve the restricted master problem using subgradient optimization, without generating any columns. If its objective value is lower than the incumbent value, we recover an approximate primal solution using the volume algorithm and use that solution to implement a branching strategy. Else, if the restricted master problem objective value is higher than the incumbent value, the hybrid algorithm is invoked to generate columns. During the hybrid algorithm iterations, a valid non-decreasing node lower bound is recorded. If that lower bound comes to exceed the incumbent value, the node is pruned and the columns generated at this node are deleted. When the hybrid algorithm terminates and the lower bound is below the incumbent value, branching is needed. In this case, the volume algorithm is invoked to recover an approximate primal solution of the restricted master problem, which is then used for branching. When the volume algorithm returns all-integer setup variables, we fix these variables and solve the resulting LP problem using formulation CL, which is a generalized network flow problem for fixed setups (Degraeve and Jans 2007). To ensure that the approximate primal solution is of good quality, additional columns are generated when the maximum violation of the linking constraints is higher than some predefined tolerance (we used 0.05). More details about this procedure can be found on the Appendix of this chapter.

2.7.3 Branch and Price Search

The below procedure summarizes the main building blocks the heuristic Branch and Price algorithm.

Algorithm 2 Summary of the heuristic Branch and Price procedure HBP

Inputs: *LowerBound, UpperBound, Incumbent, CGSolution*

Outputs: *LowerBound, UpperBound, Incumbent, NodesExplored*

1: $NodesExplored \leftarrow 0; Pass \leftarrow 0; NodeLimit \leftarrow 100;$ \\ Initializations
2: **Call** $RINS\_Variable\_Fixing(CGSolution, RINSDepth)$ \\ Fix Variables based on the CGSolution values. Store how many variables are fixed in variable RINS-Depth
3: **Call** $Branch\_And\_Price$ \\ Terminates when $NodesExplored = NodeLimit$ or when $LowerBound = UpperBound$
4: **while** $Time < TimeLimit$ **or** $LowerBound < UpperBound$ **do**
5: $Pass = Pass + 1$
6: $NodeLimit = NodeLimit + (Pass - 1) * 50$ \\ Expand the explored space during each pass
7: **Call** $Unfix\_Variables(RINSDepth)$ \\ Free the variables that are fixed at the first RINSDepth levels of the tree
8: **Call** $Update\_Fixed\_Variables$ \\ Keep the remaining fixed variables to the values of the incumbent
9: **Call** $Branch\_And\_Price$ \\ Search again, in a larger neighbourhood
10: **end while**

The exploration of the search space is done in two ways. First, we employ the concept of *relaxation induced neighborhoods search* (RINS) (Danna et al. 2005). Although other MIP-based techniques could be employed, such as local branching (Fischetti and Lodi 2003), most of them destroy the subproblem structure. After solving the root node, a variable fixing phase follows. We fix to 1 (0) the setup variables that are 1 (0) in the incumbent and more (less) than 0.8 (0.2) in the approximate primal solution of the master problem, given by the volume algorithm. Then branching is applied. Following Van Vyve and Wolsey (2006), we branch on the earliest period and on the most fractional item. The intuition behind this choice is that branching decisions in earlier periods are more important than in later periods because they are more likely to influence the subsequent production flow. During the first pass, we branch to 0 if

both the incumbent has the branching variable at 0 and its fractional value at the root note is less than 0.2, else we branch to 1. The bias against 0-branches comes from the fact that when the capacity constraints are tight, it is more likely to produce more than needed in earlier periods, and therefore a setup is necessary.

Notice that the subtree invoked from RINS is guaranteed to have a leaf node that is at least as good as the incumbent, because the incumbent itself is under that subtree. If a better feasible solution exists, it is likely to be found at the early levels of the subtree from the rounding/smoothing heuristic. For this reason, we explore the subtree induced by the RINS heuristic only partially, by imposing a node limit. A potential downside of exhaustively exploring this subtree, is that if some of the fixing decisions are not part of the optimal solution, the algorithm could stall or terminate without a significant improvement. Hence, it is necessary to modify the search space. To do this, we note that at each node the fixed variables are either fixed by the RINS fixing decisions, or by branching. To explore a different but promising part of the setup variable space, we free the variables fixed by RINS and fix the variables that were fixed by branching to the values of the incumbent solution. The previously free variables remain free. In a sense, this scheme sets free the variables previously fixed by the RINS heuristic, while it makes sure that the new search space remains close to the neighborhood of the incumbent. The same idea is applied iteratively: a number of nodes in each subtree is explored, the variables fixed early up in the free become free and the variables fixed from Branch and Price are fixed again to the value of the incumbent.

2.8 Computational Experiments

Computational experiments were performed on a 2.0 GHz / 2 GB machine. The CPU times listed in all tables refer to the time that the incumbent was found. Detailed results of all experiments can be found in the Appendix of this chapter.

2.8.1 Solving the root node: heuristics and lower bounds

To demonstrate the strength of the lower bound of the proposed hybrid scheme, we employ data instances from Trigeiro et al. (1989). These instances are among the hardest ones and have been tested extensively (Belvaux and Wolsey 2000, Van Vyve and Wolsey 2006, Degraeve and Jans 2007, Jans and Degraeve 2004 and Pochet and

Van Vyve 2004). We use 4 different formulations and 3 solution methods. Table 2.1 shows the nomenclature of different combinations of formulations (period decomposition of the Facility Location formulation - FL/P, period decomposition of the Facility Location formulation with Precedence constraints - FLp/P, period decomposition of the Shortest Path formulation - SP/P and period decomposition of the Transformed Shortest Path formulation - SPt/P) and solution methods (Lagrange Relaxation - LR, Approximate Column Generation - ACG and the Hybrid Algorithm- HB). In ACG, the master problem is solved with subgradient optimization. Each subproblem returns one column per iteration, as in standard column generation. ACG is listed to demonstrate how the hybrid scheme performs against a Lagrange-based implementation of column generation.

Table 2.1: Nomenclature of combinations of formulations and solution methods.

	Method Used to Calculate the Lower Bound		
Formulation	**LR**	**ACG**	**HB**
FL/ P	LR	ACG	HB
FLp/P	LRp	ACGp	HBp
SP/P	SPP-LR	SPP-ACG	SPP-HB
SPt/P	SPtP-LR	SPtP-ACG	SPtP-HB

Table 2.2 compares the CPU time in seconds of combinations of formulations and solution methods. For the sake of brevity, we do not present full results for formulations SP/P and SPt/P, since their behavior is very similar to that of FL/P and FLp/P respectively. However, we do show that the hybrid solution method HB is superior to Lagrange Relaxation LR and that LR applied to the transformed formulation SPt/P is much faster than when applied to the standard formulation SP/P. We list the maximum absolute violation of the linking constraints obtained by the volume algorithm in Column 2. The small violations suggest that the approximate primal solution recovered by the volume algorithm is very close to the exact primal solution. Since the difference in lower bound values that each method returned are small, we compare the time needed to calculate that lower bound only, listing the time each method takes relative to the best implementation, which is the hybrid method applied to the FLp/P formulation, called HBp. In Column 3, we report the time in seconds that HBp needs to calculate the lower bound. Columns 4-8 and 11 show the time ratio between each

algorithm and HBp. Columns 9 and 10 refer to the Shortest Path formulation and are therefore compared with the best implementation of the shortest path formulation, SPtP-HB. Both SPtP-HB and HBp use the hybrid scheme and solve the same subproblem. Consequently, the difference in their time performance is small. Finally, the last column, JD, refers to the CPU times obtained by Jans and Degraeve (2004), who utilized decomposition SP/P and implemented a standard simplex-based column generation algorithm. Note that for G57 and G72, we compare to their Lagrange relaxation times (2000 iterations), as they were not able to solve these instances with simplex-based column generation.

Table 2.2: Computational performance of different algorithms for obtaining the lower bound.

Dataset	HBp Max Violation	Time HBp (s)	HB/ HBp	LR/ HBp	LRp/ HBp	ACG/ HBp	ACGp/ HBp	SPP-LR/ SPtP-HB	SPtP-LR/ SPtP-HB	JD (SP/P)/ HBp
G30 (6-15)	0.032	0.18	2.0	8.9	3.1	9.6	9.3	4.89	1.52	8.0
G30b (6-15)	0.049	0.2	1.3	11.9	3.8	7.1	6.7	11.67	3.08	8.2
G53 (12-15)	0.016	1.6	0.9	4.6	1.2	3.5	3.5	6.45	2.90	5.7
G57 (24-15)	0.023	7.60	2.2	5.8	2.3	3.8	3.9	4.70	1.39	3.4
G62 (6-30)	0.029	0.55	1.1	7.2	2.6	15.3	12.6	4.76	1.53	681.0
G69 (12-30)	0.025	2.43	2.2	9.6	2.4	12.4	14.5	7.04	1.77	79.9
G72 (24-30)	0.031	15.77	2.3	6.9	1.6	12.0	11.1	2.73	1.40	18.4
Average	0.029	4.0	1.7	7.8	2.4	9.1	8.8	6.0	1.9	114.9

The comparison of the different approaches suggests some conclusions. The addition of the precedence constraints to the facility location formulation (column 4), and the transformation of the shortest path formulation (column 9 versus column 10) enhance computational performance. We see that on average HB needs 1.7 times the time of HBp to converge (column 4). Further, in columns 5-8 it is evident that the effect of the precedence constraints is much stronger when using LR instead of ACG. When comparing columns 5 and 6, we see that LRp is approximately 3 times faster than LR, whereas columns 7 and 8 reveal that the performance of ACG and ACGp is approximately the same. In addition, columns 9 and 10 reveal that the Lagrange relaxation of the shortest path formulation is approximately 3 times slower, on average, compared to its transformed version. Algorithms SPP-ACG, SPtP-ACG and SPP-HB showed similar behavior as ACG, ACGp and HB respectively and are not reported. The hybrid scheme outperforms all approaches. On average, HBp is 20 times faster than the other algorithms (5.2 times faster if JD is excluded). Finally, it is interesting to note JD, an

implementation of simplex-based column generation. Although the runs are made on a slower computer (Pentium 750 MHz), the CPU times are disproportionately larger, due to the poor performance of simplex-based column generation.

On assessing the lower bound quality obtained by SP/P, Jans and Degraeve (2004) give evidence that for the 7 instances of Table 2.2, the lower bound is stronger than the one obtained by Trigeiro et al. (1989), Belvaux and Wolsey (2000), Miller et al. (2000) and Degraeve and Jans (2007), while the bound from Van Vyve and Wolsey (2006) seems to be stronger for most instances. Note however that there is no ground to support theoretical arguments for which bounds are stronger, with the exception of the bound given by Trigeiro et al. (1989) and Jans and Degraeve (2004). Therefore, the performance of each methodology is data dependent. Intuitively, the period decomposition should perform well when the capacity constraints are tight, because it takes advantage of the extreme points of the single-period polytopes, and when the number of items is small, which is likely to make the subproblems easier.

We compared the performance of our algorithm to the results obtained by Süral et al. (2009). They design a Lagrange-based heuristic for a reformulation of the capacity constrained problem with setup times but without setup costs. The demand constraint is relaxed. They modify the data from Trigeiro et al. (1989) as follows. First, they set all setup costs to zero. Second, they increase zero demands to 2 units. Third, they construct new instances by reducing the number of periods of some existing ones. Finally, they construct instances with unit inventory costs for all items, called *homogeneous* (denoted hom). Instances with the original inventory cost are called *heterogeneous* (denoted het). In total, 100 instances are generated. Since their approach usually terminates in a few seconds, we have chosen to compare it with our hybrid procedure only. Specifically, we use the hybrid process to obtain a lower bound, recover an approximate primal solution with the volume algorithm, fix the setup variables to 0 or 1 (as described in the diving heuristic) and call the successive rounding/smoothing heuristic once. In particular, no branching is performed, and the algorithm is actually the procedure that is performed at the root node of the branch-and-price tree. Table 2.3 displays the average integrality gaps and the average CPU time for the best heuristic approach of Süral et al. (SDW) and our extended hybrid heuristic (EHB). SDW was run on an Intel Pentium 4 machine, and the subproblems were solved with CPLEX 7.0.

Table 2.3: Comparison of EHB with the Lagrange-based heuristic of Süral et al. (2009)

Category	Gap EHB (%)	Gap SDW (%)	CPU EHB (s)	CPU SDW (s)
12 x 10 het	18.43	31.73	0.34	3.06
24 x 10 het	11.14	18.07	1.62	4.79
12 x 15 het	19.25	25.95	1.40	6.07
24 x 15 het	13.44	21.26	6.37	15.33
12 x 30 het	21.92	28.68	3.27	24.01
24 x 30 het	23.44	32.35	19.72	38.93
12 x 10 hom	22.97	42.08	0.53	2.69
24 x 10 hom	14.06	20.88	1.77	4.45
12 x 15 hom	18.44	28.00	1.19	5.64
24 x 15 hom	14.96	20.56	3.46	11.14
12 x 30 hom	21.68	24.33	2.99	19.10
24 x 30 hom	21.35	30.26	7.95	22.91
Average het	17.94	26.34	5.45	15.37
Average hom	18.91	27.69	2.98	10.99

It is interesting to notice the large gaps that result from problems without setup cost. Clearly, EHB outperforms SDW, both in terms of gap quality and CPU time, for both homogeneous and heterogeneous problems.

We also run EHB using the F and G instances from Trigeiro. The average gaps were 2.43% and 2.51% and the average CPU times 0.25 and 2.14 seconds, respectively. Note that Degraeve and Jans (2007) cite an average gap of 2.87% for the F instances, after exploring 2000 nodes in their branch-and-price tree.

We also tested EHB against the best implementation of the iterative production estimate (IPE) heuristic of Pochet and Van Vyve (2004). They run their experiments on a 350 MHz machine and list results on the six instances from the G dataset of Trigeiro et al. (1989) described in Table 2.2 (IPE was not tested on G30). The optimal value is listed to facilitate the comparisons. Table 2.4 presents the results.

Although comparison of CPU times is hard since different machines were used, it is clear that the quality of feasible solutions is much better for EHB. Note that the above IPE results are the best that Pochet and Van Vyve cite, based on the BC-PROD cut generator. Also, despite that EHB seems to find a better feasible solution than IPE, more space for improvement exists, as shown by the optimal solutions. The branch-

Table 2.4: Comparison of EHB against the IPE heuristic

		Best Incumbent		CPU Time (s)	
Dataset	**Optimal**	**IPE**	**EHB**	**IPE**	**EHB**
G30b (6-15)	37721	38976	38162	1.31	0.27
G53 (12-15)	74634	78098	75035	3.53	1.4
G57 (24-15)	136509	137153	136884	6.01	7.21
G62 (6-30)	61746	63073	63018	1.7	0.67
G69 (12-30)	130596	131988	131668	6.03	3.38
G72 (24-30)	287929	290006	288313	21.15	15.5

and-price heuristic performs EHB at the root node and tries to approach the optimal solution. Its computational peformance is described in section 2.8.2.

2.8.2 Comparison of heuristic branch-and-price with other approaches

We compared our approach with the most recent and successful heuristic and exact approaches found in the literature. In order to do this, we employed the 7 instances taken from Trigeiro et al. (1989) described in section 2.8.1. A comparison based on 7 instances is limited. However, these 7 instances are specifically tested in many other papers and therefore it allows us to compare our results to various other results reported in the literature. Table 2.5 presents results of the following studies: Müller et al. (2012) (MS), Degraeve and Jans (2007) (DJ), Belvaux and Wolsey (2000) (BW) and our approach (HB&P). The optimal solution of each instance is also listed, to facilitate the comparisons. MS is a randomized heuristic, so it does not provide any lower bounds and gaps.

Before analyzing the relative performance of each approach, some implementation details are presented. Müller et al. (2012) use a hybrid adaptive large scale neighborhood search strategy. They run their experiments on a 2.66 GHz / 8GB RAM machine and use CPLEX 10.2 to create repair neighborhoods. Since their approach is randomized, the listed values are based on an average of 100 runs, where each run lasts for 60 seconds. Degraeve and Jans (2007) develop an exact branch-and-price algorithm. They pose a limit of 2000 nodes and run their experiments on a 750 MHz processor. Finally, Belvaux and Wolsey (2000) use a relax-and-fix heuristic which they incorporate within BC-PROD, a customized branch-and-cut system for production planning

Table 2.5: Comparison of Branch-and-Price upper bounds, CPU times and integrality gaps with other approaches

		Best Incumbent				CPU Time (s)			Gaps(%)		
Dataset	**Optimal**	MS	DJ	BW	HB&P	DJ	BW	HB&P	DJ	BW	HB&P
G30 (6-15)	37809	-	37809	-	37809	33	-	5	1.90	-	1.00
G30b (6-15)	37721	37776.4	38162	37221	37721	29	-	2	2.51	1.35	0.90
G53 (12-15)	74634	74720.8	75035	74752	74634	66	189	9	1.61	1.33	0.93
G57 (24-15)	136509	130675	136860	136509	136509	44	55	101	0.36	0.10	0.07
G62 (6-30)	61746	61792.2	62644	61746	61746	359	55	140	2.79	1.24	0.88
G69 (12-30)	130596	130675	131234	130599	130599	215	102	131	0.81	0.32	0.20
G72 (24-30)	287929	287966	288383	287950	288016	306	298	46	0.22	0.07	0.07

problems. They use a 200 MHz machine to perform their computations.

In terms of quality of feasible solutions, it seems that the two most competitive approaches are BW and HB&P. It is interesting to notice that, although BW is the oldest approach and runs are made on a slow machine, it seems to provide much better feasible solutions compared to most other approaches. When compared to HB&P, it finds solutions of similar quality. CPU times are not directly comparable. HB&P produces a better gap however, as the lower bound is stronger and the solution quality similar. HB&P gives a stronger bound because most separation routines of BC-PROD use flow-based inequalities that incorporate information mainly from the convex hulls of the single - item uncapacitated polytopes. HB&P however, gives a lower bound that describes the intersection of the capacity and single-item uncapacitated polytopes and therefore it tends to be stronger for tightly constrained problems. Finally, DJ is outperformed both in terms of time and solution quality.

To the best of our knowledge, the best exact approach found in the literature for the above instances is the approximate extended formulation of Van Vyve and Wolsey (2006). Unfortunately, the authors do not cite the CPU time at which they obtain the lower bound at the root node and a comparison with our approach is not possible. However it is expected that a heuristic implementation of their approach would work better for problems with short time horizon, whereas our approach is better for long horizon problems. For example, they solve G62 (6 items, 30 periods) to optimality after 220400 nodes and 1078 seconds on a 1.6GHz machine whereas we need 4212 nodes and 140 seconds to find the optimal value (on a 2GHz machine).

Next, we compared our approach with the best branch-and-price algorithm of those suggested by Pimentel et al. (2010) (PAV). They used the Trigeiro et al. (1989) sets X11117 - X12429. Each X set comprises of 5 instances and there are 30 X sets, giving a total of 150 instances. They applied a per period, per item and simultaneous per period and per item decomposition, solved the subproblems with CPLEX 8.1, and performed their computations on a Pentium 4 machine with 1 GB RAM. Table 2.6 presents results for the 10 hardest sets, i.e. those for which their algorithm gives the largest gaps. The bottom line lists average results for all 150 instances. Detailed results can be found in the Appendix.

Table 2.6: Comparison with Pimentel et al. (2010)

	CPU Time PAV (s)	**CPU Time HB&P (s)**	**Gap PAV (%)**	**Gap B&P (%)**
X11419	3600	96.06	10.367	7.069
X11429	3600	106.30	9.599	4.993
X12429	3600	110.15	7.999	3.650
X12419	3600	84.07	5.967	4.250
X11428	3600	68.42	4.777	0.951
X12428	3600	61.83	3.908	1.563
X12229	3600	29.93	3.236	1.924
X11229	3600	66.01	3.045	2.071
X12219	3600	62.96	2.745	2.507
X11129	3600	63.37	2.525	2.874
Average (150 instances)	3600	74.91	5.417	3.185

Note that HB&P outperforms PAV in all of the above sets except one. Moreover, HB&P terminated within the time limit of 150 seconds in 149 out of 150 instances. Also, the average gap and CPU times are much better for HB&P. An interesting observation is that Pimentel et al. (2010) do not get their best gaps from their simultaneous item/period decomposition, which theoretically gives a stronger bound compared to both their item and period decompositions, but from the item decomposition. This is because the master problems of the simultaneous item/period decomposition are very degenerate and time consuming. Therefore, they may not be able to obtain good feasible solutions and improve their gap within their time limit.

In a final round of trials, we tested our algorithm on some new hard datasets against the CPLEX v12.2 solver (CPX), using the regular formulation CL. The purpose of this comparison is to give evidence on the relative strengths and weaknesses of a decomposition approach against a modern off-the-shelf branch-and-cut software. To this end,

we constructed new harder instances by modifying the Trigeiro dataset. Specifically, we replicated the demand patterns to 60 periods, and reduced the capacity to 95% of its original value. We focused on instances with 6 and 12 items because the integrality gap of the extended formulations improves as the number of items increases, therefore problems with fewer items are more challenging to solve. Finally, we excluded instances that were infeasible without initial inventory because their gap was sensitive to the initial inventory cost.

The process described above led to the creation of 30 new 60-period instances, 21 of which have 6 items and 9 with 12 items. Table 2.7 shows the computational results. Both algorithms used 100 seconds of CPU time.

Table 2.7: Integrality gaps of Heuristic Branch-and-price and CPLEX v12.2

	CPX Gap (%)	**HB&P Gap (%)**	**Gap Closed (%)**
6 - 60 Average	2.29	2.44	17.90
6 - 60 Median	1.86	1.97	9.51
12 - 60 Average	1.79	1.57	12.58
12 - 60 Median	1.71	1.57	9.73
Total Average	2.14	2.18	15.29
Total Median	1.85	1.78	9.57

The computational experiments show that both algorithms achieve good performance in terms of integrality gaps. However, no instance was solved to optimality after 100 seconds. The total median gaps show that heuristic branch-and-price performs better overall, but the average gaps are slightly higher than those of CPLEX. We observed that the lower bound obtained by the period decomposition was always better. To demonstrate the efficiency of the lower bound, we calculated all gaps using the best feasible solution and report the amount of gap that is closed with our branch-and-price algorithm. The amount of gap closed is the difference between the CPX Gap and HB&P Gap, divided by the CPX Gap, where all gaps are calculated using the best feasible solution. The last column indicates the superiority of the lower bound obtained by the period decomposition, as the average gap improvement is about 15%. The lower bound obtained by CPLEX is weak, even after exploring a large part of the branch-and-bound tree. Specifically, CPLEX explores more than 28,000 nodes on average, whereas we explore an average of 644 nodes with our approach. The fact

that CPLEX explores such a large part of the tree allows it to find better feasible solutions in most instances, so its integrality gaps are competitive to branch-and-price. In conclusion, the two approaches give similar results in terms of gap quality, but our approach dominates CPLEX in lower bounds, since it takes advantage of the special structure of the single period polytopes.

The overall conclusion from the computational experiments is that the proposed algorithm and solution approach either outperform or compare favorably with other state-of-the-art approaches, including a commercial implementation of branch-and-cut. Period decompositions deliver strong lower bounds, because their feasible region lies in the intersection of the single-period polytopes of extended formulations, and are therefore stronger than the single-item decompositions of Degraeve and Jans (2007) and the single-period and combined decompositions of Pimentel et al. (2010). The branch-and-price heuristic that is based on period decomposition shows good performance, especially for large instances, namely problems with a large number of periods and items.

2.9 Conclusions and future research

We have presented period decompositions of the facility location and shortest path formulations of the capacity constrained lot size problem with setup times. The subproblems polytopes do not have the integrality property, and therefore an improved lower bound is obtained. Customized branch-and-bound algorithms are developed to solve the single-period subproblems. It is shown that a standard column generation scheme is computationally inefficient, and a novel, subgradient-based algorithm is developed, that combines column generation and Lagrange relaxation. This scheme gives a valid lower bound which is a good approximation of the exact lower bound. Moreover, an approximate primal solution of the restricted master problem is recovered with the volume algorithm. The performance of the hybrid scheme is enhanced further with the proposition of a transformed shortest path formulation and with the addition of a class of valid inequalities in the subproblem. This class improves the quality of the columns that price out, and is equivalent to adding dual optimal inequalities in the dual of the restricted master problem. In addition, the subproblem is still tractable with a fast customized branch-and-bound algorithm. Finally, a branch-and-price based heuristic

is designed, that integrates relaxation induced neighborhoods, selective diving and successive rounding/smoothing within a novel strategy of node exploration. The latter explores promising parts of the branch-and-bound tree based on information obtained by new feasible solutions. Computational results show that the proposed approach outperforms or compares favorably with the most recent and successful approaches found in the literature.

There are several directions that need further research. The implementation of standard stabilization techniques (eg. du Merle et al. 1999) to extended formulations may make them more tractable computationally. To this end, our transformed shortest path formulation can be used as a stabilization method to problems that use similar formulations, such as the multi-commodity network flow problems. In addition, the precedence constraints we introduced for production planning can be used to facility location problems and their extensions. On a different line, the integration of approximate schemes such as the volume algorithm could lead to enhanced MIP-based heuristics that can tackle very large instances efficiently and give a good dual bound, used to assess their performance. Finally, the per-period decomposition could lead to the development of successful approximation algorithms. Recently, Levi et al. (2008) used the simple plant location formulation with added flow cover inequalities to derive the first 2-approximation algorithm for a variant of CLST without setup times. It would be interesting to explore whether a column generation-based relaxation would lead to similar approximation schemes for CLST.

2.10 Appendix

2.10.1 Proof of Theorem 2.1

Theorem 2.1 states that $D' \subseteq D$ where

$$D = \left\{ v_{it} \in \Re \;\middle|\; \begin{array}{ll} v_{it} - v_{i,k+1} \leq cv_{itk} \, \forall i \in I, t, k \in T : t \leq k < m & (A.1) \\ v_{i1} - v_{it+1} \leq ci_{it} \; \forall i \in I, t \in T \backslash \{m\} & (A.2) \\ v_{it} \leq cv_{itm} \; \forall i \in I, t \in T & (A.3) \\ v_{i1} \leq ci_{im} \; \forall i \in I & (A.4) \end{array} \right\};$$

$$D' = \left\{ \bar{v}_{it} \in \Re \quad \left| \begin{array}{l} \sum_{l=1}^{t} \bar{v}_{il} \leq \min\{ci_{it}, cv_{i1t}\} \ \forall i \in I, t \in T \ \ (A.5) \\ \sum_{l=t}^{k} \bar{v}_{il} \leq cv_{itk} \ \forall i \in I, t, k \in T : k \geq t \ \ (A.6) \end{array} \right. \right\}$$

Note that $D = \bigcup_{i=1}^{n} D_i$ and $D' = \bigcup_{i=1}^{n} D_i^{'}$ and $\bigcap_{i=1}^{n} D_i = \bigcap_{i=1}^{n} D_i^{'} = \emptyset$, therefore we can omit the item index in the context of this proof. Also, without loss of generality we can restrict our attention to the positive orthant of both dual spaces because (2.7) can be written as inequality ($\geq$) and also (2.8) can be written as inequality ($\leq$). We will first show that $D' \subseteq D$. For this, consider a point $v_{it} \in D'$. Note that (A.5) and (A.6) imply (A.3) and (A.4). Supposing that (A.6) holds, we show that (A.1) holds as well. We write (A.6) as $v_{it} + \underbrace{\sum_{l=t+1}^{k} v_{il}}_{U} \leq cv_{itk} \Leftrightarrow v_{it} + U \leq cv_{itk}$. Observe that if $v_{it} - v_{i,k+1} > cv_{itk}$, then $v_{it} - v_{i,k+1} > cv_{itk} \geq v_{it} + U \Leftrightarrow U + v_{ik+1} < 0$, which cannot hold because $v_{it} \in \Re_+$ in both spaces. Therefore, (A.6) implies (A.1). Using the same argument, (A.5) implies (A.2). This means that $v_{it} \in D' \Rightarrow v_{it} \in D$ for an arbitrary point $v_{it} \in D'$ and thus $D' \subseteq D$. We show that the inclusion may be strict via an example. Consider the following single-item 2-period uncapacitated system with no initial inventory and its transformed version:

$$\begin{array}{lllll} \min & c_{11}z_{11} + c_{12}z_{12} + c_{22}z_{22} & & [\mathbf{P}] & (A.7) \\ \text{s.t.} & z_{11} + z_{12} & \geq 1 & [v_1] & (A.8) \\ & -z_{11} + z_{22} & \geq 0 & [v_2] & (A.9) \\ & z_{11}, z_{12}, z_{22} & \geq 0 & & (A.10) \end{array}$$

$$\begin{array}{lllll} \min & c_{11}z_{11} + c_{12}z_{12} + c_{22}z_{22} & & [\mathbf{P'}] & (A.11) \\ \text{s.t.} & z_{11} + z_{12} & \geq 1 & [v_1] & (A.12) \\ & z_{12} + z_{22} & \geq 1 & [v_2] & (A.13) \\ & z_{11}, z_{12}, z_{22} & \geq 0 & & (A.14) \end{array}$$

The corresponding dual formulations are:

$$\max \quad v_1 \qquad\qquad [\mathbf{D}] \tag{A.15}$$
$$\text{s.t.} \quad v_1 - v_2 \leq c_{11} \tag{A.16}$$
$$v_1 \leq c_{12} \tag{A.17}$$
$$v_2 \leq c_{22} \tag{A.18}$$
$$v_1, v_2 \geq 0$$
$$\max \quad v_1 + v_2 \qquad [\mathbf{D'}] \tag{A.19}$$
$$v_1 \leq c_{11} \tag{A.20}$$
$$v_1 + v_2 \leq c_{12} \tag{A.21}$$
$$v_2 \leq c_{22} \tag{A.22}$$
$$v_1, v_2 \geq 0 \tag{A.23}$$

Figure 2.3 shows the feasible regions of (D) and (D') . Clearly $(D') \subset (D)$, and the proof is complete.

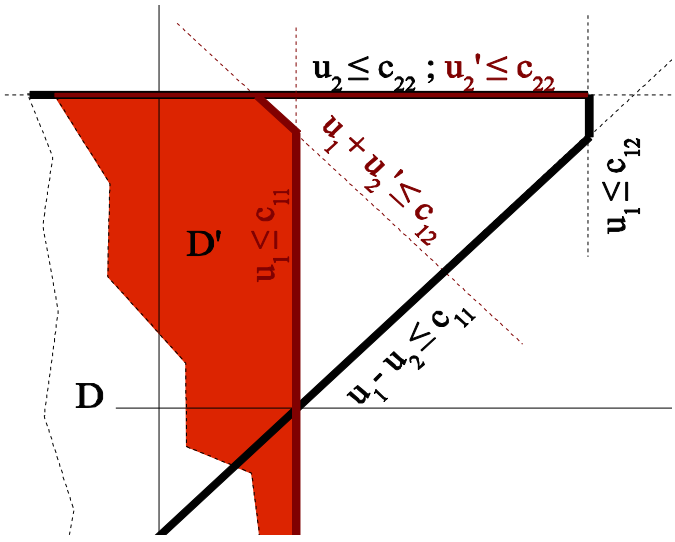


Figure 2.3: Dual space reduction in a small example. -

2.10.2 Master problems SP/P and SPt/P

To describe the period decomposition of the shortest path formulation SP/P, we introduce additional notation. We define S_t as the set of all the extreme production plans of the subproblem defined by $(2.9) - (2.12)$ for each $t \in T$. The variable zt_q is associated with production plan q for period t. The master problem of SP/P is then as follows (Jans and Degraeve, 2004).

$$\min \quad \sum_{t\in T}\sum_{q\in S_t} ct_{tq} zt_{tq} + \sum_{i\in I}\sum_{t\in T} ci_{it} p_{it} \tag{A.24}$$

$$\text{s.t.} \quad \sum_{k=1} (p_{ik} + \sum_{q\in S_1} \alpha_{i1kq} zt_{1q}) = 1 \qquad \forall i \in I \tag{A.25}$$

$$p_{i,t-1} + \sum_{k=1}^{t-1}\sum_{q\in S_k} \alpha_{itkq} zt_{kq} = \sum_{k=t}^{m}\sum_{q\in S_t} \alpha_{itkq} zt_{tq} \quad \forall i \in I, \forall t \in T\backslash\{1\} \tag{A.26}$$

$$\sum_{q\in S_t} zt_{tq} = 1 \qquad \forall t \in T \tag{A.27}$$

$$y_{it} = \sum_{q\in S_t} \bar{y}_{itq} zt_{tq} \qquad \forall i \in I, \forall t \in T \tag{A.28}$$

$$zt_{tq} \geq 0,\ p_{it} \geq 0,\ y_{it} \in \{0,1\} \qquad \forall i \in I, \forall q \in S_t, \forall t \in T \tag{A.29}$$

The constraint coefficients α_{itkq}, the cost parameters ct_{tq} and the setup parameters $\bar{y}_{itq}$ are defined by the extreme point production plans of subproblem $(2.9) - (2.12)$ for each $t \in T$. The master problem SPt/P is derived by SP/P, by replacing, for each $t > 2$, each constraint $(A.26)$ with the sum of constraints $(A.26)$ for all $1 \leq l \leq t$. Using this transformation, constraints $(A.25) - (A.26)$ can be written as $\sum_{k=t}^{m} p_{ik} + \sum_{j=1}^{t}\sum_{k=t}^{m}\sum_{q\in S_t} \alpha_{itkq} zt_{tq} = 1$, for each $i \in I, t \in T$.

2.10.3 Complete computational results

Table 2.8 presents the complete results for the 36 Trigeiro instances X11117 - X12429. For each set, the average CPU time and the average gap is reported. Runs were made on a 2.0GHz / 2GB RAM machine.

Tables 2.9 and 2.10 present results for the EHB heuristic tested on the homogeneous and heterogeneous instances constructed in Süral et al (2009) respectively. MGxx are the Gxx instances with 10 periods and MMG the Gxx instances with 15 periods.

Table 2.8: Computational results for the Trigeiro et al. (1982) X11117 - X12429 datasets

Set	CPU Time (s)	Gap	Set	CPU Time (s)	Gap	Set	CPU Time (s)	Gap
X11419	96.06	7.07%	**X12129**	33.98	0.92%	**X11128**	1.15	0.09%
X11429	106.30	4.99%	**X12418**	31.37	0.86%	**X11217**	12.35	0.07%
X12419	84.07	4.25%	**X12218**	38.67	0.59%	**X11118**	0.40	0.04%
X12429	110.15	3.65%	**X12417**	25.62	0.56%	**X12117**	0.21	0.04%
X12119	80.22	3.46%	**X12128**	1.68	0.38%	**X11227**	1.60	0.03%
X11119	55.94	3.13%	**X12427**	23.05	0.27%	**X12127**	0.10	0.03%
X11129	63.37	2.87%	**X11228**	36.55	0.26%	**X11117**	0.01	0.00%
X11219	72.62	2.51%	**X11427**	60.03	0.25%	**X11127**	0.01	0.00%
X12219	62.96	2.51%	**X11218**	37.93	0.24%	**X11418**	105.00	0.98%
X11229	66.01	2.07%	**X12228**	14.21	0.23%	**X11428**	68.42	0.95%
X12229	29.93	1.92%	**X11417**	57.89	0.22%	**X12217**	4.95	0.17%
X12428	61.83	1.56%	**X12118**	0.53	0.18%	**X12227**	3.64	0.11%

2.10.4 Implementation Details

The hybrid optimization scheme is implemented as follows. First, it is initialized with a vector of dual prices and a lower bound, both coming either from TTM or from the LP relaxation of the SPL formulation. All subgradient algorithms use the standard step length formula with smoothing. In Lagrange relaxation, we perform 20 iterations if the number of hybrid iterations is less than 3 or greater than 20, and 4 iterations otherwise. The step length is 1.05 and if it is not improved at an iteration, it is multiplied by 0.6. We use a 90% smoothing weight in each violation. The upper bound in the subgradient formula is updated as 1.01 times the current restricted master problem value. The restricted master problem is solved similarly. The step length is 0.6 and is multiplied by 0.9 if there is no bound improvement for 10 iterations. The smoothing parameter is 50% initially and then it is halved when the bound improvement is less than 1% in the last 100 iterations.

The volume algorithm is an extension of the classic subgradient method that recovers approximate primal solutions. Barahona and Anbil (2000) show that the primal solution of the lagrange problem, that may violate the constraints that are dualized in the objective function, can be used to recover an approximate primal solution. They suggest an exponential smoothing formula, in which the primal solution in iteration

Table 2.9: The performance of EHB on homogeneous instances

Instance	LB	UB	Gap (%)	CPU Time (s)	Instance	LB	UB	Gap (%)	CPU Time (s)
G51	185.8	218.8	17.74	1.22	**MG56**	66.6	80	20.14	1.66
G52	129.2	159.9	23.7	0.91	**MG57**	169.5	182.2	7.49	2.19
G53	123.3	148	19.98	0.83	**MG58**	194.6	210.5	8.18	1.77
G54	83.7	105	25.39	1.05	**MG59**	190.3	201.5	5.87	2.53
G55	289.9	312.6	7.84	1.51	**MG60**	142.9	151.5	6.01	1.89
G56	247.3	266.2	7.66	3.02	**MG66**	75.5	89.5	18.51	0.38
G57	327.3	356.9	9.03	4.20	**MG67**	41.7	49	17.42	0.43
G58	416.1	441.4	6.09	3.78	**MG68**	78.7	93.2	18.42	0.58
G59	463.9	495.5	6.82	4.03	**MG69**	11.4	18.6	63.16	0.48
G60	369.7	409.0	10.63	3.37	**MG70**	69.4	79.8	14.92	0.51
G66	353.9	445.4	25.87	3.11	**MG71**	62.1	71.3	14.85	1.76
G67	377.9	448.0	18.55	2.73	**MG72**	25.5	36.7	44.20	1.82
G68	738.8	841.9	13.96	3.41	**MG73**	163.2	176.4	8.07	2.17
G69	234.6	298.4	27.20	2.83	**MG74**	76.9	86.6	12.56	1.99
G70	398.9	490	22.85	2.88	**MG75**	145.3	163.8	12.73	1.91
G71	208.5	262	25.65	7.21	**MMG66**	113.8	134.8	18.46	1.04
G72	112.3	162.3	44.48	7.01	**MMG67**	157.3	183.3	16.57	0.82
G73	946.2	1017.2	7.51	8.00	**MMG68**	78.7	93.2	18.42	0.58
G74	398.9	485.6	21.74	6.79	**MMG69**	62.1	82.7	33.15	1.09
G75	1000.5	1074.1	7.35	10.75	**MMG70**	157.9	179.7	13.79	0.98
MG51	96.0	105.3	9.71	0.67	**MMG71**	76.0	89.1	17.27	2.30
MG52	58.8	67.6	14.91	0.37	**MMG72**	45.6	71.5	56.83	2.94
MG53	58.5	65.9	12.59	0.68	**MMG73**	363.3	390.6	7.51	3.61
MG54	24.2	36.3	49.94	0.66	**MMG74**	186.3	221.4	18.83	3.09
MG55	127.1	139.9	10.09	0.53	**MMG75**	345.3	376.2	8.96	4.27

r, x_p^r, is updated as $x_p^r = x + \alpha * x_p^{r-1} + (1 - \alpha) * x_l^r$, where x_l^r is the solution of the lagrange problem in iteration r, and α is a smoothing constant. The volume algorithm uses subgradient updating with a step length of 0.02 and is multiplied by 0.9 if there is no bound improvement for 10 iterations. The *target value* G is initialized at 95% of the lower bound and it is updated as follows. Denote the lower bound by lb. Then whenever $lb > 0.95T$ we set $G = 1.005lb$. Finally, the volume algorithm is short-cutted whenever the step length falls bellow a specified tolerance (0.001) or if the maximum

Table 2.10: The performance of EHB on heterogeneous instances

Instance	LB	UB	Gap (%)	CPU Time (s)	Instance	LB	UB	Gap (%)	CPU Time (s)
G51	354.5	392.8	10.82	1.22	**MG56**	112.4	118.3	5.23	1.27
G52	154.6	177.7	14.95	0.91	**MG57**	204.1	216.4	6.05	2.32
G53	194	233.1	20.15	0.83	**MG58**	281.6	305.7	8.54	1.04
G54	197.3	239	21.12	1.05	**MG59**	313.9	348.2	10.94	1.38
G55	500.9	583.4	16.48	1.51	**MG60**	261.7	288.5	10.24	1.96
G56	536.5	591.1	10.18	3.02	**MG66**	99	109.3	10.4	0.43
G57	427.8	463.6	8.36	4.2	**MG67**	35.8	48.6	35.86	0.17
G58	707.6	743.7	5.1	3.78	**MG68**	136	165.8	21.94	0.2
G59	970.2	1100.6	13.44	4.03	**MG69**	12.4	18.9	52.54	0.25
G60	830	917.6	10.55	3.37	**MG70**	189.3	202.1	6.77	0.41
G66	562.6	660.6	17.41	3.11	**MG71**	77.1	81.5	5.71	3.67
G67	753.2	948.2	25.88	2.73	**MG72**	27.5	37.6	36.73	0.92
G68	1636.9	1889.7	15.45	3.41	**MG73**	262.3	283.4	8.04	3.23
G69	335	427.4	27.57	2.83	**MG74**	89.1	99.2	11.34	1.12
G70	1261.4	1554.8	23.26	2.88	**MG75**	212.9	225.1	5.72	1.59
G71	313.5	403	28.55	7.21	**MMG66**	153.2	175.9	14.82	0.55
G72	114.3	172.4	50.78	7.01	**MMG67**	259	318	22.78	0.79
G73	1947.6	2171.7	11.51	8	**MMG68**	136	165.8	21.94	0.25
G74	507.3	606.3	19.51	6.79	**MMG69**	71	97.3	37.12	0.57
G75	2072.7	2214.2	6.82	10.75	**MMG70**	495.1	601.3	21.44	0.9
MG51	141.1	153	8.4	0.67	**MMG71**	94.5	110.6	16.99	3.44
MG52	69.3	76.3	10.09	0.37	**MMG72**	53.1	74.8	40.92	2.65
MG53	73.8	87.2	18.14	0.68	**MMG73**	673.6	752.3	11.69	4.7
MG54	38.6	42.4	9.82	0.66	**MMG74**	234.9	261.6	11.38	5.22
MG55	185.6	204.8	10.34	0.39	**MMG75**	601.5	636.1	5.75	4.26

absolute violation is less than 3% (2% for the root node) and the gap between the lower bound and the primal solution is less than 3%.

The branch-and-price scheme is implemented in the following sequence. After obtaining a lower bound at the root node, a diving phase follows. The order of variable fixing is the following. We start from the last period. For each item, we calculate a cost to weight ratio, which is the setup cost over the setup time and the demand of some periods ahead. We sort the items and start fixing from the one that has the smallest

ratio *and* complies with the fixing criteria described in the paper. All setup variables that comply with the fixing criteria will be fixed, but the fixing order is important. If the resulting subtree is solved completely, we need to backtrack to one of the nodes that are fixed and it is important to backtrack to an influential fixing decision. Therefore, we fix the variables in reverse period order and within each period we choose to fix the most insignificant items first.

When we examine the second node on the same level, we delete the columns that were generated on the fist node. We also delete columns when backtracking. At each node we first try to find a feasible solution and then we perform 60 subgradient iterations on the restricted master problem. If its objective is higher than the incumbent, we generate new columns. Then, if the lower bound is higher than the incumbent we prune the node, and if it not we call the volume algorithm, with 200 iterations. If the maximum violation is more than 5% and we have not generated columns, we try to generate columns, check the new lower bound against the incumbent and call the volume algorithm again, if needed. If the volume algorithm identifies a feasible solution, we fix the setup variables and solve the resulting extended network problem in the standard (x, y) space. The algorithm terminates in the following situations: a) the time limit is reached (we set 100 seconds for all runs) b) the algorithm backtracks to the root node For the small instances (i.e., 6-15), the algorithm backtracks to the root node, while for the larger ones the time limit is reached.

2.11 References

Alfieri, A., P. Brandimarte, S. D'Orazio. LP-based heuristics for the capacitated lot-sizing problem: the interaction of model formulation and solution algorithm. *International Journal of Production Research,* 40, 441–458, 2002.

Barahona, F., D. Jensen. Plant location with minimum inventory. *Math. Programming,* 83, 101–111, 1998.

Barahona, F., R. Anbil. The volume algorithm: producing primal solutions with a subgradient method. *Math. Programming,* 87, 385–399, 2000.

Barahona, F., R. Anbil. On some difficult linear programs coming from set partitioning. *Disc. App. Mathematics,* 118, 3–11, 2002.

Barany, I., T. Van Roy, L. Wolsey. Strong formulations for multi-item capacitated lot-sizing. *Man. Sci.,* 30, 1255–1261, 1984.

Ben Amor, H., J. Desrosiers, J.M. Valrio de Carvalho. Dual-optimal inequalities for stabilized column generation. *Oper. Res.,* 54, 454–463, 2007.

Belvaux, G., L. Wolsey. Bc-prod: A specialized branch-and-cut system for lot-sizing problems. *Man. Sci.,* 46, 724–738, 2000.

Caserta, M., E.Q. Rico. A cross-entropy lagrangian hybrid algorithm for the multi-item capacitated lot-sizing problem with setup times. *Comp. and Oper. Res.,* 36, 530–548, 2009.

Crowder, H. Computational improvements for subgradient optimization. *Symp. Mathematica* XIX 357–372, 1976.

Danna, E., E. Rothberg, C. Le Pape. Exploring relaxation induced neighborhoods to improve MIP solutions. *Math. Programming Ser. A,* 102, 71–90, 2005.

Degraeve, Z., M. Peeters. Optimal integer solutions to industrial cutting-stock problems: Part 2, Benchmark results. *INFORMS J. Computing,* 15, 58–81, 2003.

Degraeve, Z., R. Jans. A new Danzig-Wolfe reformulation and branch-and-price algorithm for the capacitated lot-sizing problem with setup times. *Oper. Res.,* 55, 909–920, 2007.

Denizel, M., F. T. Altekin, H. Süral, H. Stadtler. Equivalence of the LP relaxation of two strong formulations for the capacitated lot-sizing problem with setup times. *OR Spectrum,* 30, 773–785, 2008.

Denizel, M., H. Süral. On alternative mixed integer programming formulation and LB-based heuristics for lot-sizing with setup times. *Journal of the Operational Research Society,* 57, 389–399, 2006.

Du Merle, O., D. Villeneuve, J. Desrosiers, P. Hansen. Stabilized column generation. *Disc. Mathematics,* 194, 229–237, 1999.

Elhallaui, L., D. Villeneuve, F. Soumis, G. Desaulniers. Dynamic aggregation of set-partitioning constraints in column generation. *Oper. Res.,* 53, 632–645, 2005.

Eppen, G.D., R.K. Martin. Solving multi-item capacitated lot-sizing problems using variable redefinition. *Oper. Res.* 35, 832–848, 1987.

Fischetti, M., A. Lodi. Local branching. *Math. Programming Ser. B,* 98, 23–47, 2003.

Geoffrion, A. M. Lagrangean relaxation for integer programming. *Mathematical Programming Study,* 2, 82–114, 1974.

Gondzio, J., P. G. Brevis, P. Munari. New developments in the primal-dual column generation technique. *Europ. J. Oper. Res.,* 224, 41–51, 2013.

Graham, R. L. An efficient algorithm for determining the convex hull of a finite planar set. *Information Processing Letters,* 1, 132–133, 1972.

Holmberg, K., D. Yuan. A Lagrangean heuristic based branch-and-bound approach for the capacitated network design problem. *Oper. Res.,* 48, 461–481, 2000.

Huisman, D., R. Jans, M. Peeters, A.P.M. Wagelmans. Combining column generation and lagrangian relaxation. G. Desaulniers, J. Desrosiers, M. Solomon, eds. *Column Generation. Springer*, New York, 247–270, 2005.

Jans, R., Z. Degraeve. Improved lower bounds for the capacitated lot-sizing problem with setup times. *Oper. Res. Lett.,* 32, 185–195, 2004.

Jans, R., Z. Degraeve. Meta-heuristics for dynamic lot-sizing: a review and comparison of solution approaches. *Europ. J. Oper. Res.,* 177, 1855–1875, 2007.

Krarup, J., I. Bilde. *Plant location, set covering and economic lot size: an O(mn) algorithm for structural problems.* Numerische methoden bei optimierungsaufgaben, Band 3: optimierung bei graphentheoretischen und ganzzahligen problemen, Vol. 36, Birkhauser Verlag, Basel and Stuttgart, 1977.

Levi, R. A. Lodi, M. Sviridenko. Approximation algorithms for the capacitated multi-item lot-sizing problems via flow-cover inequalities. *Math. Oper. Res.,* 33, 461–474, 2008.

Lübbecke, M., J. Desrosiers. Selected topics in column generation. *Oper. Res.,* 53, 1007–1023, 2005.

Manne, A. S. Programming of economic lot sizes. *Man. Sci.,* 4(2):115–135, 1958.

Miller, A.J., G.L. Nemhauser, M.W. Savelsberg. Solving multi-item capacitated lot-sizing problems with setup times by branch-and-cut. CORE Discussion paper 2000/39, UCL, Louvain-la-Neuve, Belgium, 2000.

Mitchell, J. Cutting plane methods and subgradient methods. *INFORMS TutORials in Operations Research*, INFORMS New Orleans, 2005.

Müller, L.F., S. Spoorendonk, Pisinger, D. A hybrid adaptive large neighborhood search heuristic for lot-sizing with setup times. *Europ. J. Oper. Res.,* 218, 614–623, 2012.

Pimentel, C.M.O., F.P. Alvelos, J.M. Valrio de Carvalho. Comparing danzig-wolfe decompositions and branch-and-price algorithms for the multi-item capacitated lot sizing problem. *Opt. Meth. Soft.,* 25, 299–319, 2010.

Pisinger, D. A minimal algorithm for the multiple-choice knapsack problem. *Europ. J. Oper. Res.,* 83, 394–410, 1995.

Pochet, Y., M. Van Vyve. A generic heuristic for production planning problems. *INFORMS J. Computing,* 16, 316–327, 2004.

Sinha, P., A. Zoltners. The multiple-choice knapsack problem. *Oper. Res.,* 27, 503–515, 1979.

Subramanian, S., H. Sherali. An effective deflective subgradient optimization scheme for implementing column generation for large scale airline crew scheduling problems. *INFORMS J. Computing,* 20, 565–578, 2008.

Süral, H., M. Denizel, L.N. Van Wassenhove. Lagrangean relaxation based heuristics for lot sizing with setup times. *Europ. J. Oper. Res.,* 194, 51–63, 2009.

Trigeiro, W., L.J. Thomas, J.O. McClain. Capacitated lot sizing with set-up times. *Man. Sci.,* 35, 353–366, 1989.

Van Vyve, M., L. Wolsey. Approximate extended formulations. *Math. Programming Ser. B,* 105, 501–522, 2006.

Vanderbeck, F. Lot-sizing with start-up times. *Man. Sci.,* 44, 1409–1425, 1998.

Vanderbeck, F., M. W. P. Savelsbergh. A generic view of Danzig-Wolfe decomposition in mixed integer programming. *Oper. Res. Lett.,* 48, 111–128, 2006.

Wolsey, L. Uncapacitated lot-sizing problems with start-up costs. *Oper. Res.,* 37, 741–747, 1989.

3

A Horizon Decomposition approach for the Capacity Constrained Lot Size Problem with Setup Times

We introduce the *Horizon Decomposition* in the context of Dantzig-Wolfe Decomposition, and apply it to the Capacity Constrained Lot Size Problem with Setup Times. We partition the problem horizon in contiguous overlapping intervals and create subproblems identical to the original problem, but of smaller size. The user has the flexibility to regulate the size of the master problem and the subproblem via two scalar parameters. We investigate empirically which parameter configurations are efficient, and assess their robustness at different problem classes. Our branch-and-price algorithm outperforms state-of-the-art branch-and-cut solvers when tested to a new dataset of challenging instances that we generated. We show how our methodology can be generalized to mathematical programs with generic constraint structure. Finally, we benchmark horizon decomposition with the period decomposition approach described in chapter 2.

3.1 Introduction

Since the seminal work of Dantzig and Wolfe (1960), Dantzig–Wolfe Decomposition has been applied successfully to solving Linear, Integer and Mixed Integer Linear Pro-

gramming problems and a variety of applications. Gilmore and Gomory (1963) were the first to implement a practical algorithm for the cutting stock problem and since then problems of increasing complexity have been solved (Lübbecke and Desrosiers 2005). The implementation of the Dantzig–Wolfe Decomposition principle involves the recognition of a part of the constraint matrix that has block diagonal structure, where each block is associated with a subset of variables. The variables that appear in each block should not appear in other blocks and if so, the corresponding constraints are treated as "complicated". This explains why most research and practical applications are usually problem-specific. In addition, although for certain large-scale problems branch-and-price algorithms may have superior performance against branch-and-cut software, the range of applications is limited by the block diagonal structure that is in place, and by how exploitable this structure is. The competitive advantage of Dantzig–Wolfe reformulations stems from exploiting these substructures to obtain an improved dual bound. This occurs in cases where the subproblem does not have the integrality property (Geoffrion 1974), which means that its linear relaxation does not have all integral extreme points. The backbone of the most successful applications is usually a specialized algorithm that solves the subproblem efficiently.

One of the main contributions of this paper is to show that any generic Mixed Integer Linear Program (MIP) can be reformulated in such a way that it is amenable to Dantzig–Wolfe Decomposition. A distinct characteristic of our approach is that we can regulate the size of the master problem and the subproblem independently, by introducing two scalar parameters. This flexibility suggests that one can experiment with alternative decompositions and address the trade-off between subproblem difficulty and dual bound strength directly. We have performed extensive computational experiments to analyze the efficiency of the horizon decomposition approach, using capacitated lot sizing problems as a testbed. The results indicate that certain decomposition configurations can tackle some particularly hard instances far more efficiently than modern branch-and-cut solvers. We introduce the main idea in the context of the Capacity Constrained Lot Size Problem with Setup Times (CLST) for several reasons. First, CLST constitutes one of the simplest but yet most challenging problem structures. Trigeiro, Thomas and McClain (1989) introduced the problem and constructed a dataset of 540 instances, the hardest of which remained unsolvable until the last decade. Although today all instances can be solved within a few seconds, several researchers

(Araujo et al. 2011, Müller, Spoorendok and Pisinger 2012, Süral et al. 2009) have constructed instances with long horizons, tight capacity constraints or without setup costs which remain intractable. Second, the multi-period nature of lot sizing problems and the complicating structure of the capacity constraints provide an excellent ground to demonstrate the horizon decomposition principle. Based upon this setting, the generalization of our approach comes naturally. Finally, CLST is well-studied in the literature and therefore we can benchmark the efficiency of our approach against other techniques, such as valid inequalities, extended formulations and alternative decomposition schemes. In addition, in some special cases it is possible to establish which approach gives the best bound or draw correspondences across different methodologies.

The principal aim of this work is to illustrate that the application of horizon decomposition to the CLST has at least two important benefits. First, one can exploit the technology of modern solvers in solving subproblems of manipulable size and strength. Since the subproblem size is controlled independently from the size of the master problem, it is possible to find a balance between dual bound quality and subproblem tractability. Second, our computational experiments show that in practice the method shows excellent behavior in perhaps the most challenging class of problems, namely, instances with low ratio of items over periods and tight capacity constraints.

The remainder of this paper is organized as follows. Section 2 gives a brief literature review on column generation methodologies and on CLST-specific research. Section 3 introduces the problem formulation. Section 4 applies the horizon decomposition. A comparison and correspondences with other lower bounds is demonstrated. Section 5 describes a branch-and-price algorithm that uses the horizon decomposition. Section 6 presents computational experiments and section 7 describes how the approach is generalized to generic MIPs. Finally, section 8 concludes the paper with suggestions for future work.

3.2 Literature Review

Column generation was employed by Gilmore and Gomory (1963) to solve the Dantzig-Wolfe reformulation of the cutting stock problem. Since then, many authors have used it either as a stand-alone technique to solve large linear programs (Elhallaoui et al. 2005), or as a bounding technique within branch-and-bound algorithms (Degraeve and

Jans 2007), a scheme also known as *branch-and-price*. From a computational perspective, the most recent advances in branch-and-price algorithms include attempts to combine column generation with cutting planes. In this line, Desauliers (2010) solves the split-delivery vehicle routing problem with time windows by using an efficient label-setting algorithm to solve the resource-constrained shortest path subproblems alongside a cutting plane routine that strengthens the lower bound of column generation. The efficiency of his approach is improved by branching on variables that lead to a balanced branch-and-bound tree. On the theoretical side, there are works that examine the efficient convergence of column generation and the branching rules used in branch-and-price. Ben Amor et al. (2006) show that reducing the feasible dual space of the master problem leads to faster convergence. Degraeve and Jans (2007) demonstrate how the Dantzig-Wolfe decomposition principle is applied to MIPs with an application to the CLST and Vanderbeck and Savelsbergh (2006) develop a theoretical framework. Vanderbeck (2011) explores the issue of branching in branch-and-price when the subproblems are identical, while Villeneuve et al. (2005) construct a compact formulation and use the corresponding variables for branching. The reviews of Lübbecke and Desrosiers (2005) and Barnhart et al. (1998) describe plenty of applications and discuss in detail technical issues of column generation and branch-and-price respectively.

Lagrange relaxation is a related reformulation that, in theory, gives the same dual bound as column generation. Fisher (1981) gives an overview of Lagrange relaxation and describes early applications. The strong dual bound and the relative speed of Lagrange relaxation have led to the development of efficient exact and heuristic methods. Holmberg and Yuan (2000) develop a Lagrange-based heuristic to solve the capacitated network design problem. They use an efficient specialized algorithm to solve the subproblems and their approach gives competitive gaps in large instances, while they are able to solve medium instances to optimality. Likewise, Caprara et al. (1999) present an efficient heuristic for large-scale set covering problems coming from crew scheduling in railways.

Lagrangian decomposition is a generalization of Lagrange relaxation that yields stronger lower bounds. Guignard and Kim (1987) were the first to introduce it in the context of MIPs that have two sets of constraints. The main idea is to introduce "copy" constraints for the original variables and dualize them in the objective function. Our implementation can be seen as a case of Lagrange decomposition since we also introduce

copies of variables and explore the convex hull of the corresponding subproblems. It is more advantageous however in that it is instance-specific, it avoids unnecessary variable copying and performs a systematic reformulation that creates a decomposable structure.

The literature in capacity constrained lot size problems is vast. In their seminal paper, Wagner and Whitin (1958) introduced the single-item uncapacitated version of the problem and solved it using a dynamic programming recursion. Trigeiro, Thomas and McClain (1989) were the first to examine a multi-item problem with capacity constraints and setup times. They showed experimentally that setup times make the problem harder and developed a Lagrange-based smoothing heuristic whose performance remains competitive up to date. An earlier result by Kleindorfer and Newson (1975) proves that the problem is strongly NP-hard. To obtain an improved lower bound, Eppen and Martin (1987) reformulated the problem with shortest-path variables that describe the convex hull of the single-item uncapacitated polyhedron. Similarly, Barany, van Roy and Wolsey (1984) describe the same polyhedron using valid inequalities. In more recent advancements, Degraeve and Jans (2007) develop an exact branch-and-price algorithm using a per-item decomposition and Jans and Degraeve (2004) describe a decomposition of the shortest path formulation that leads to an improved lower bound. The most recent work that applies Dantzig-Wolfe decomposition to the CLST is by Pimentel et al. (2009). They develop three alternative decompositions and branch-and-price algorithms and compare their performance. Finally, another stream of research focuses on finding good feasible solutions with heuristics. Süral et al. (2009) develop a Lagrange-based heuristic for a variant of the CLST without setup costs. They used the subproblem solutions to construct incumbents during the subgradient optimization process and obtained small integrality gaps over a set of hard instances. Similarly, Müller, Spoorendok and Pisinger (2012) use large scale neighborhood search combined with strong formulations and report results on new hard instances. Finally, Akartunali et al. (2013) use column generation to generate cuts from a 2-period relaxation of CLST. The authors use several distance functions, by which they are able to generate valid inequalities that cut-off the linear programming relaxation solution, when this solution has a positive distance from a predefined 2-period lot sizing set.

Our work has contributions in both the Dantzig-Wolfe decomposition and lot sizing literatures. First, we show how Dantzig-Wolfe decomposition can be applied in a novel

way, such that MIPs can be decomposed in subproblems identical to the original, but of smaller size. Second, we demonstrate the applicability of this idea in lot-sizing and investigate under which conditions it is advantageous against competitive methodologies. Third, we show experimentally that a class of CLST instances, namely those with tight capacity constraints and small ratio of items over periods, are time consuming to solve with modern branch-and-cut software. We develop a branch-and-price approach based on horizon decomposition and demonstrate its efficiency against competitive approaches. Finally, we show the extension of our idea to generic MIPs.

3.3 Problem Description and Formulation

3.3.1 Original Formulation

The capacity constrained lot size problem with setup times generalizes the basic single-item uncapacitated lot size problem studied by Wagner and Whitin (1958). Specifically, it models a multi-item setting with one capacity constraint per period and item-specific setup times and production times. It can be used in production planning for determining the production and setup decisions of an MRP system by taking into consideration one bottleneck resource (Pochet and Wolsey 2006). We formulate the problem using the following notation:

Sets

$I = \{1, ..., n\}$: Set of items, indexed by i.
$T = \{1, ..., m\}$: Set of periods, indexed by t.

Parameters

d_{it}: demand of item i in period t, $\forall i \in I, \forall t \in T$.
sd_{itk}: sum of demand of item i from period t till period k, $\forall i \in I, \forall t, k \in T : t \leq k$.
hc_{it}: cost of holding inventory for item i from period $t-1$ to period t, $\forall i \in I, \forall t \in T$.
sc_{it}: setup cost of item i in period t, $\forall i \in I, \forall t \in T$.
vc_{it}: production cost of item i in period t, $\forall i \in I, \forall t \in T$.
st_{it}: setup time of item i in period t, $\forall i \in I, \forall t \in T$.
vt_{it}: variable production time of item i in period t, $\forall i \in I, \forall t \in T$.
M_{it}: big-M quantity, defined as $M_{it} = \min\{sd_{itm}, \frac{cap_t - st_{it}}{vt_{it}}\}, \forall i \in I, \forall t \in T$.

cap_t: time capacity in period t, $\forall t \in T$.

Decision Variables

x_{it}: production quantity of item i in period t, $\forall i \in I, \forall t \in T$.
s_{it}: inventory quantity of item i at the beginning of period t, $\forall i \in I, \forall t \in T \cup \{m+1\}$.
y_{it}: =1 if setup for item i in period t, =0 otherwise, $\forall i \in I, \forall t \in T$.

The mathematical formulation of CLST is then as follows:

$$\min \quad \sum_{i \in I} \sum_{t \in T} sc_{it} y_{it} + \sum_{i \in I} \sum_{t \in T} vc_{it} x_{it} + \sum_{i \in I} \sum_{t \in T} hc_{it} s_{it} \tag{A.1}$$

$$\text{s.t.} \quad s_{it} + x_{it} = d_{it} + s_{i,t+1} \qquad \forall i \in I, \forall t \in T \tag{A.2}$$

$$x_{it} \leq M_{it} y_{it} \qquad \forall i \in I, \forall t \in T \tag{A.3}$$

$$\sum_{i \in I} st_{it} y_{it} + \sum_{i \in I} vt_{it} x_{it} \leq cap_t \qquad \forall t \in T \tag{A.4}$$

$$x_{it}, s_{it} \geq 0, s_{i,m+1} = 0, y_{it} \in \{0,1\} \quad \forall i \in I, \forall t \in T \tag{A.5}$$

The objective function $(A.1)$ minimizes the total cost, that consists of the setup cost, the production cost and the inventory holding cost. To model problems that are infeasible without initial inventory, we allow for initial inventory (Vanderbeck 1998). Constraints $(A.2)$ indicate that demand in each period is covered either by initial inventory or by production, and that the remaining quantity is transferred to the next period. $(A.3)$ links the setup and production decisions and $(A.4)$ describes the per-period capacity constraints. Finally, constraints $(A.5)$ pose non-negativity and integrality restrictions to the problem variables. We use v_{CLST} to denote the optimal objective value of $(A.1)$–$(A.5)$ and $\bar{v}_{CLST}$ to denote its optimal LP relaxation objective value. The next paragraph describes a family of reformulations that allow a generic decomposition scheme for the CLST.

3.3.2 Horizon Reformulation

The fundamental idea of horizon decomposition is to reformulate the problem by using a desired *horizon cover*. A horizon cover $\mathcal{P}$ is a set whose elements are horizons H of the form $H = \{m_0, m_0 + 1, \ldots, m_1\} \subseteq T$, with $m_1 > m_0$. Therefore, each horizon $u \in \mathcal{P}$ consists of a certain number of periods, starting at $m_0(u)$ and ending at $m_1(u)$,

and the horizon cover is the union of possibly overlapping horizons. To characterize the horizon cover set, we introduce the following notation:

$U := \{1, \ldots, p\}$: Index set of $\mathcal{P}$, with $p := |\mathcal{P}|$. Also, we let $\bar{U} := U \backslash \{p\}$.
$H_u := \{m_0(u), m_0(u)+1, \ldots, m_1(u)\}$ where $m_0(u), m_1(u) \in T$ for all $u \in U$. We also define $m_1(p) = m+1$, and $H_0 = H_{p+1} = \emptyset$.
$\bar{H}_u := H_u \backslash \{m_1(u)\}$, for all $u \in U$.
$L_u := H_u \cap H_{u+1}$ and $\bar{L}_u := L_u \backslash \{m_1(u)\}$, for all $u \in U$.

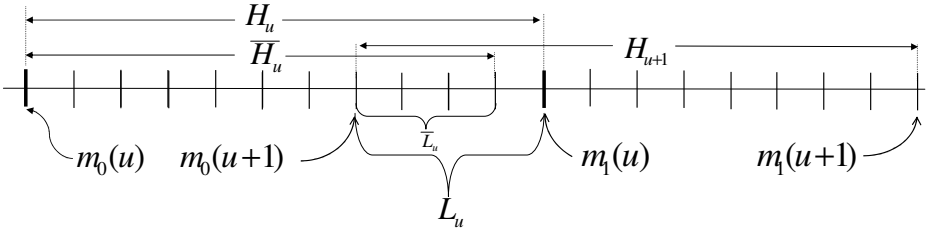


Figure 3.1: Notation used in horizon covering. Each horizontal line segment indicates a discrete time period $t \in T$.

Figure 3.1 demonstrates the notation associated with each horizon. For convenience, we assume throughout the paper that whenever a set that defines a constraint is empty, then the constraint is not defined. Using the above notation, the horizon cover is written as $\mathcal{P} := \{H_1, \ldots, H_p\}$. Note that some periods can be common in two or more horizons. Finally, we assume that two contiguous horizons H_u and H_{u+1} have at least one common period, which implies that $m_1(u) \in H_{u+1}$, for all $u \in \bar{U}$.

Next, we define production, setup and inventory variables for each horizon :

x^u_{it}: production quantity of item i in period t in horizon H_u, $\forall i \in I, \forall t \in \bar{H}_u, \forall u \in U$.
s^u_{it}: starting inventory quantity of item i in period t in horizon H_u, $\forall i \in I, \forall t \in H_u, \forall u \in U$.
y^u_{it}: =1 if setup for item i in period t in horizon H_u, =0 otherwise, $\forall i \in I, \forall t \in \bar{H}_u, \forall u \in U$.

In addition, let $\alpha_{tu} = 1$, if $t \in H_u \backslash L_{u-1}, = 0$ otherwise, for all $t \in H_u, u \in U$. Using the above notation, problem $(A.1)$–$(A.5)$ can be reformulated as follows:

$$\min \quad \sum_{i\in I}\sum_{u\in U}\sum_{t\in \bar{H}_u} \alpha_{tu}(sc_{it}y^u_{it} + vc_{it}x^u_{it}) + \sum_{i\in I}\sum_{u\in U}\sum_{t\in H_u} \alpha_{tu} hc_{it}s^u_{it} \tag{A.6}$$

$$\text{s.t.} \quad s^u_{it} + x^u_{it} = d_{it} + s^u_{i,t+1} \qquad \forall i \in I, \forall t \in \bar{H}_u, \forall u \in U \tag{A.7}$$

$$x^u_{it} \le M_{it} y^u_{it} \qquad \forall i \in I, \forall t \in \bar{H}_u, \forall u \in U \tag{A.8}$$

$$\sum_{i\in I} st_{it}y^u_{it} + \sum_{i\in I} vt_{it}x^u_{it} \le cap_t \quad \forall t \in \bar{H}_u, \forall u \in U \tag{A.9}$$

$$s^u_{it} = s^{u+1}_{it} \qquad \forall i \in I, \forall t \in L_u, \forall u \in \bar{U} \tag{A.10}$$

$$x^u_{it} = x^{u+1}_{it} \qquad \forall i \in I, \forall t \in \bar{L_u}, \forall u \in \bar{U} \tag{A.11}$$

$$y^u_{it} = y^{u+1}_{it} \qquad \forall i \in I, \forall t \in \bar{L_u}, \forall u \in \bar{U} \tag{A.12}$$

$$x^u_{it} \ge 0, y^u_{it} \in \{0,1\} \qquad \forall i \in I, \forall t \in \bar{H}_u, \forall u \in U \tag{A.13}$$

$$s^u_{it} \ge 0 \qquad \forall i \in I, \forall t \in H_u, \forall u \in U \tag{A.14}$$

Constraints $(A.7) - (A.9)$ and $(A.13) - (A.14)$ define a CLST over horizon $\bar{H}_u$. This implies that the corresponding inventory variables s^u_{it} are defined over $H_u = \bar{H}_u \cup \{m_1(u)\}$. Therefore, period $m_1(u)$ is used to associate the ending inventory variables of each CLST defined over $\bar{H}_u$, exactly as period $m+1$ is used to set the ending inventories to zero in formulation $(A.1)$–$(A.5)$. Constraints $(A.10) - (A.12)$ impose that variables indexing the same period in two horizons should attain the same values. Finally, objective function $(A.6)$ considers the setup, inventory and production costs of all horizons. Parameter α_{tu} is an indicator used for the appropriate allocation of costs: if a variable is defined in two horizons, then its cost is allocated to the earliest horizon. Like in Lagrange decomposition (Guignard and Kim 1987), it is straightforward to see that the variables indexed within horizon overlaps can be allocated any fraction of the original cost, without loss of generality.

Note that a benefit of the above reformulation is its flexibility. By selecting the parameters $m_0(u)$ and $m_1(u)$ for each $u \in U$, one can regulate the number of subproblems, subproblem length and periods of overlap. Moreover, the formulation remains valid when no overlap between horizons exists, i.e., when $\bar{H}_u \cap \bar{H}_{u+1} = \emptyset$. In this case $\bar{L_u} = \emptyset$, and there are no linking constraints for the production and setup variables. Finally, the original formulation $(A.1) - (A.5)$ can be considered as a special case of $(A.6) - (A.14)$ where the horizon cover is a singleton with $m_0 = 1$ and $m_1 = m+1$. The next section describes how the above structure is used in Dantzig-Wolfe reformulation.

3.4 Horizon Decomposition

3.4.1 Initial Formulation

Formulation $(A.6)$–$(A.14)$ decomposes per horizon, with the exclusion of constraints $(A.10$–$A.12)$. Let us note with $(\bar{f}_u)$ the subset of a block of constraints (f) that refer to a specific horizon $u \in U$. Also let $(x, y, s)_u = ((x^u_{it}, y^u_{it}) : i \in I, t \in \bar{H}_u, s^u_{it} : i \in I, t \in H_u)$. We then define the *single horizon polyhedron* as $\mathcal{W}_u := \{(x, y, s)_u | (\bar{A.7})_u - (\bar{A.9})_u, (\bar{A.13})_u - (\bar{A.14})_u, s^u_{im_1(u)} \leq sd_{im_1(u),m} \forall i \in I\}$ and let $\mathcal{E}_u$ be the set of extreme points of $conv(\mathcal{W}_u)$, for each $u \in U$. Note that we bound the ending inventory variables, $s^u_{im_1(u)}$, with the remaining item demand, $sd_{im_1(u),m}$. This way, the subproblem space is bounded and no extreme rays are needed for its description. Each extreme point $e = (\bar{x}, \bar{y}, \bar{s})_{ue} \in \mathcal{E}_u$ is associated with the following elements:

c_{ue}: total cost of production, setup and inventory of horizon H_u according to production plan e, =

$\sum_{i \in I} \sum_{t \in \bar{H}_u} \alpha_{tu}(sc_{it}\bar{y}^u_{ite} + vc_{it}\bar{x}^u_{ite}) + \sum_{i \in I} \sum_{t \in H_s} \alpha_{tu} hc_{it}\bar{s}^u_{ite}$

z_{ue}: fraction of production plan e that is used for actual production.

The Dantzig-Wolfe reformulation is then as follows.

$$[\widetilde{\mathbf{DW}}] \qquad \min \sum_{u \in U} \sum_{e \in \mathcal{E}_u} c_{ue} z_{ue} \tag{A.15}$$

$$s.t. \qquad \sum_{e \in \mathcal{E}_u} \bar{s}^u_{ite} z_{ue} = \sum_{e \in \mathcal{E}_{u+1}} \bar{s}^{u+1}_{ite} z_{u+1,e} \forall i \in I, \forall t \in L_u, \forall u \in U \backslash \{p\} \tag{A.16}$$

$$\sum_{e \in \mathcal{E}_u} \bar{x}^u_{ite} z_{ue} = \sum_{e \in \mathcal{E}_{u+1}} \bar{x}^{u+1}_{ite} z_{u+1,e} \forall i \in I, \forall t \in \bar{L}_u, \forall u \in U \backslash \{p\} \tag{A.17}$$

$$\sum_{e \in \mathcal{E}_u} \bar{y}^u_{ite} z_{ue} = \sum_{e \in \mathcal{E}_{u+1}} \bar{y}^{u+1}_{ite} z_{u+1,e} \forall i \in I, \forall t \in \bar{L}_u, \forall u \in U \backslash \{p\} \tag{A.18}$$

$$\sum_{e \in \mathcal{E}_u} z_{ue} = 1 \qquad \forall u \in U \tag{A.19}$$

$$s_{it} = \sum_{e \in \mathcal{E}_u} \bar{s}^u_{ite} z_{ue} \qquad \forall i \in I, \forall (t, u) \in H_u \times U : \alpha_{tu} = 1 \tag{A.20}$$

$$x_{it} = \sum_{e \in \mathcal{E}_u} \bar{x}^u_{ite} z_{ue} \qquad \forall i \in I, \forall (t, u) \in \bar{H}_u \times U : \alpha_{tu} = 1 \tag{A.21}$$

$$y_{it} = \sum_{e \in \mathcal{E}_u} \bar{y}^u_{ite} z_{ue} \qquad \forall i \in I, \forall (t, u) \in \bar{H}_u \times U : \alpha_{tu} = 1 \tag{A.22}$$

$$z_{ue} \geq 0 \qquad \forall e \in \mathcal{E}_u, \forall u \in U \tag{A.23}$$

$$s_{it} \geq 0 \qquad \forall i \in I, t \in T \qquad \text{(A.24)}$$

$$y_{it} \in \{0,1\}, x_{it} \geq 0 \qquad \forall i \in I, t \in T \qquad \text{(A.25)}$$

Formulation [$\widetilde{\mathbf{DW}}$] is equivalent to the original formulation, in the sense that they both attain the same optimal solution. However, the optimal linear programming relaxation objective of [$\widetilde{\mathbf{DW}}$] is always at least as large as that of the original formulation, because the subproblems do not have the integrality property (Geoffrion 1974). Constraints $(A.16)$–$(A.18)$ correspond to $(A.10)$–$(A.12)$ and denote that in any period common to two horizons, the production, setup and inventory quantities should attain the same value in both horizons. Constraints $(A.19)$ together with the non-negativity constraints $(A.23)$ impose that each decision variable is a fraction of an extreme production plan. Equations $(A.20)$–$(A.22)$ define the variables of the original formulation as convex combinations of extreme production plans. Although the number of variables and constraints is large, there are certain reductions that can be performed, which are described in the next section.

3.4.2 Model Reductions

[$\widetilde{\mathbf{DW}}$] is a valid reformulation of the CLST. Without loss of generality, constraints $(A.17)$ and $(A.20) - (A.21)$ can be eliminated. The elimination of the later is straightforward as they only map the solution to the original variable space. To see that $(A.17)$ is redundant, note that $\bar{x}^u_{ite} = d_{it} + \bar{s}^u_{i,t+1,e} - \bar{s}^u_{ite}$ for each $e \in \mathcal{E}_u$. This implies that $\sum_{e\in\mathcal{E}_u} \bar{x}^u_{ite} z_{ue} = d_{it} + \sum_{e\in\mathcal{E}_u} \bar{s}^u_{i,t+1,e} z_{ue} - \sum_{e\in\mathcal{E}_s} \bar{s}^s_{ite} z_{ue} = d_{it} + \sum_{e\in\mathcal{E}_{u+1}} (\bar{s}^{s+1}_{i,t+1,e} - \bar{s}^{u+1}_{ite}) z_{u+1,e} = \sum_{e\in\mathcal{E}_{u+1}} \bar{x}^{u+1}_{ite} z_{u+1,e}$. We have shown the following result.

Corollary Constraints $\sum_{e\in\mathcal{E}_{u+1}} \bar{x}^{u+1}_{ite} z_{u+1,e} = \sum_{e\in\mathcal{E}_u} \bar{x}^u_{ite} z_{ue} \forall i \in I, \forall t \in \bar{L}_u, \forall u \in U\backslash\{p\}$ are redundant.

We denote [**DW**] the model resulting from $(A.15)$–$(A.25)$ with the exclusion of redundant constraints.

Note that one cannot eliminate the setup definition constraints and impose the integrality restrictions on the extreme production plan variables z_{se} (Degraeve and Jans 2007, Vanderbeck and Savelsbergh 2006). A correct reformulation would define, for each extreme point, a binary variable that describes the setup configurations and a continuous variable with the associated production decisions. However, the usability of this

reformulation is restricted, because the resulting branch-and-bound tree is unbalanced (Vanderbeck 2011). In our implementation we branch on the original setup variables by fixing them at the subproblems and by removing the generated columns that do not adhere to the node branching decisions, therefore using $(A.22)$ only implicitly.

3.4.3 Strength of the Lower Bound

In this part we investigate the strength of the lower bound obtained by the horizon decomposition. Since an explicit description of the convex hull of CLST is not known, we can compare the lower bound strength with lower bounds obtained by other approaches. The fact that the subproblems do not have the integrality property implies that the lower bound given by [**DW**], $\bar{v}_{DW}$, is at least as good as that obtained by the LP relaxation of $(A.1)$–$(A.5)$, $\bar{v}_{CLST}$. Hence, $\bar{v}_{CLST} \leq \bar{v}_{DW}$. More interesting is the comparison with the bound obtained when the (l, S) inequalities of Barany, van Roy and Wolsey (1984) are appended to the original formulation $(A.1)$–$(A.5)$. If we denote this bound by $\bar{v}_{lS}$, we can state the following proposition.

Proposition 3.1 *The lower bound $\bar{v}_{DW}$ does not dominate $\bar{v}_{lS}$ or vice versa.*

Proof Consider an instance with $cap_t \geq \sum_{i \in I}(sd_{itm} + st_{it})$ for each $t \in T$. This condition makes the capacity constraints redundant and the problem decomposes in a series of single–item uncapacitated problems. Since the (l, S) inequalities describe the convex hull of the single–item uncapacitated problems, $\bar{v}_{lS} \geq \bar{v}_{DW}$. Moreover, this inequality can be strict. To see this, consider without loss of generality a horizon cover with two subproblems, i.e., $S = \{1, 2\}$ and let L be the index set of overlapping periods. Given this cover, we can construct an instance for which the inequality $s_{ik} + \sum_{t \in \{k,\ldots,l\} \setminus S} x_{it} + \sum_{t \in S} sd_{itl} y_{it} \geq sd_{ikl}$ is binding for some $k \in H_1 \setminus L, l \in H_2 \setminus L$, which implies $\bar{v}_{lS} > \bar{v}_{DW}$. On the other hand, consider a single–item instance with binding capacity constraints, and $s_k = 0$ for some period k at an optimal solution. A horizon decomposition with $H_1 = \{1, \ldots, k-1\}$ and $H_2 = \{k, \ldots, m\}$ will deliver an optimal solution of the original problem, so $\bar{v}_{DW} = v_{CLST}$. However, $\bar{v}_{lS} \leq v_{CLST}$ because the (l, S) inequalities do not suffice to describe the convex hull of the capacitated problem. $\square$.

We can use similar arguments to show that there is no strict dominance between the horizon decomposition and the decompositions considered by Jans and Degraeve (2004) and Araujo et al. (2011). Note that the lower bound of the latter is at least as strong at $\bar{v}_{lS}$, since they apply decomposition to the network reformulation of Eppen and Martin (1987), which describes the same convex polyhedron as the (l, S) inequalities. Finally, $\bar{v}_{DW}$ is at least as strong as the lower bound obtained by the per period decomposition of Pimentel et al. (2009), since their per period decomposition formulation is a special case of a horizon decomposition, where each horizon defines a single-period subproblem for each period.

3.5 A Branch-and-Price Algorithm

Although the relaxation of [**DW**] can give a strong lower bound in most problems, the setup variables, defined by ($A.22$), can be fractional, and therefore a branch-and-price approach is necessary. We first employ a simple heuristic that constructs good quality feasible solutions. Then, we do column generation to find a lower bound for the MIP optimal solution. Finally, we embed column generation in a branch-and-bound scheme, thereby developing a branch-and-price algorithm. This section describes the most important components of our algorithm and outlines the most crucial implementation decisions.

3.5.1 Initialization

The column generation procedure has finite convergence and gives a lower bound provided that the initial restricted master problem is feasible (Lübbecke and Desrosiers 2005). The most common approach to initialize the master problem is to introduce columns with high cost that render it feasible. However, this might result in a large number of iterations, thereby reducing computational efficiency (Vanderbeck 2005). To tackle this issue, we employ the lot elimination heuristic (LEH) utilized by Degraeve and Jans (2007) on top of introducing high cost columns. LEH starts by fixing all setup variables to 1 and progressively eliminates them using some priority rules. LEH terminates when all setup variables are eliminated. Every time LEH finds an improved solution, we add its components as columns to the restricted master problem. These columns, in general, do not correspond to subproblem extreme points, but provide a

good family of points to *warm-start* the column generation process. In addition, LEH outputs an initial upper bound which is used in later stages of column generation.

3.5.2 Hybrid Column Generation and Stabilization

Subproblem Formulation. After initializing the restricted master problem, we start generating columns. Specifically, from each subproblem we add the column that has the minimum reduced cost. The problem of finding the minimum reduced cost problem can be formulated as a CLST defined over each subproblem horizon. We denote by us_{itu}, uy_{itu} and dc_u the dual values of $(A.16)$, $(A.18)$ and $(A.19)$ respectively, and define the indicator variable

$$\delta_{tu} = \begin{cases} 1 & \text{if } t \in L_{u-1} \\ -1 & \text{if } t \in L_u \\ 0 & \text{else} \end{cases}$$

The subproblem is then formulated as follows:

$$\begin{aligned}
[\mathbf{SP_u}] \quad \min v_u = & \sum_{i\in I}\sum_{t\in \bar{H}_u}(\alpha_{tu} sc_{it} + \delta_{tu} uy_{itu}) y^u_{it} + \\
& \sum_{i\in I}\sum_{t\in \bar{H}_u} \alpha_{tu} vc_{it} x^u_{it} + \sum_{i\in I}\sum_{t\in H_u}(\alpha_{tu} hc_{it} + \delta_{tu} us_{itu}) s^u_{it} - dc_u && \text{(A.26)} \\
\text{s.t.} \quad & s^u_{it} + x^u_{it} = d_{it} + s^u_{i,t+1} && \forall i \in I, \forall t \in \bar{H}_u && \text{(A.27)} \\
& x^u_{it} \le M_{it} y^u_{it} && \forall i \in I, \forall t \in \bar{H}_u && \text{(A.28)} \\
& \sum_{i\in I} st_{it} y^u_{it} + \sum_{i\in I} vt_{it} x^u_{it} \le cap_t && \forall t \in \bar{H}_u && \text{(A.29)} \\
& x^u_{it}, s^u_{it} \ge 0, y^u_{it} \in \{0,1\} && \forall i \in I, \forall t \in \bar{H}_u && \text{(A.30)} \\
& 0 \le s^u_{i,m_1(u)} \le sd_{im_1(u),m} && \forall i \in I && \text{(A.31)}
\end{aligned}$$

Although $[\mathbf{SP_u}]$ is a CLST itself, it has smaller dimension than the original CLST (A.1)–(A.5) and it is usually easier to solve efficiently. Despite the fact that a smaller problem dimension does not necessarily imply increased efficiency in general, there are two arguments that justify this claim in the present context. First, given that the problem structure is the same, instances of small dimension will, on average, be solved to optimality faster than larger ones. Second, an early result by Manne (1958) implies that when the number of items is large compared to the number of periods, the

single–item uncapacitated lot sizing convex hull relaxation of CLST gives a lower bound that approaches the integer optimal solution value. The latter convex hull is described by the (l, S) inequalities of Barany, van Roy and Wolsey (1984). Since most modern solvers add the violated (l, S) inequalities as cutting planes (Belvaux and Wolsey 2001), problems of short periods have tight LP relaxations and can be solved efficiently. These observations are confirmed by our computational experiments, where subproblems were solved efficiently by a modern MIP solver.

Column Generation. When the optimal objective function value v_u is negative, we append the corresponding optimal solution vector as a column to the restricted master problem [**DWR**]. Next, we resolve [**DWR**] and use the resulting set of optimal dual values to resolve subproblems [$\mathbf{SP_u}$]. This procedure terminates when no columns price out, i.e. when $\sum_{u \in U} \min(v_s, 0) = 0$. It is worth noticing that a valid lower bound on the original problem objective value is at hand throughout column generation. If v^r_{RMP} is the optimal objective value of the restricted master problem at iteration r, then a valid lower bound is $v^r_{LB} = v^r_{RMP} - \sum_{u \in U} \min(v_s, 0)$.

Stabilization and Algorithmic refinements. It has been observed by many researchers that the primal solutions of the restricted master problem are usually degenerate (Du Merle et al. 1999, Lübbecke and Desrosiers 2005, Vanderbeck 2005). This degeneracy harms the efficiency of column generation: it implies that the dual restricted master problem has multiple optimal solutions and therefore the dual optimal solution at hand might not be an accurate representation of the optimal dual space. If a dual optimal solution of bad quality is used to price out columns in the subproblems, then the generated columns may not be used in the optimal solution of the subsequent restricted master problem. In this case, column generation takes a degenerate step. This phenomenon has severe impact on the algorithmic performance, and it is usually magnified as the final optimal solution is approached, thereby called the *tailing-off* effect (Vanderbeck 2005).

We employ several techniques to stabilize column generation. During early iterations, we use a hybrid column generation–Lagrange relaxation scheme, similar to those described in Degraeve and Peeters (2003) and Barahona and Jensen (1998) and in chapter 2. More specifically, after using the dual values of the restricted master problem to price out new columns, we do not add the new columns to the master problem immediately but generate a new set of dual values via subgradient optimization (Fisher

1981). This updating process is deemed to lead to better quality dual prices, and it has the added benefit that no LP solution is required. It is called whenever column generation takes a degenerate step, i.e., when the optimal master problem objective remains the same in two consecutive iterations. We also adopt a two-phase approach, using both approximate and exact solutions. During phase I we restrict the dual space of the restricted master problem [**DW**] by introducing artificial variables on the primal space, as described in Du Merle et al. (1999). This technique reduces the number of degenerate iterations via reducing the feasible dual space. In addition, during the early stages of column generation the aim is to generate columns that describe progressively more accurate inner representations of the primal space of [**DW**]. Towards this end, we solve the subproblems to feasibility, and we also append all feasible solutions that price out. Throughout phase I, valid lower bounds are calculated using the subproblem lower bounds: $v^r_{LB} = v^r_{RMP} - \sum_{u \in U} \min(\bar{v}_u, 0)$ at iteration r, where $\bar{v}_u$ is a lower bound of [$\mathbf{SP_u}$]. When $|v^r_{RMP} - v^r_{LB}| \leq \epsilon$ for some given $\epsilon > 0$, we switch to phase II, where we apply standard column generation.

3.5.3 Branch-and-Price Strategies

We branch on the original setup variables using (A.18) implicitly. Specifically, we impose the branching restrictions at the subproblem level, and remove existing columns that do not adhere to the branching configuration of each node. We branch on the earliest fractional variable, which is an efficient selection rule for most lot-sizing problems (Van Vyve and Wolsey 2004). Finally, we adopt a best-first approach, i.e., we explore the node with the weakest lower bound first. This strategy is beneficial when the time spent at each node is large, because it minimizes the number of nodes explored in the branch-and-bound tree. In our application, we select horizon decompositions that deliver very strong lower bounds but the solution time of each node is rather large. Therefore, the combination of best-first search and tight lower bounds constitutes an efficient enumeration procedure.

The algorithm consists of three main parts: branching, column generation and pruning. Whenever a node lower bound is lower than the incumbent upper bound, v_{UB}, we branch and apply column generation to its children. If during column generation we calculate a lower bound greater than v_{UB}, we prune the node and delete the generated columns. This is in contrast with columns that do not adhere to branching decisions:

we keep the latter in a pool and add them back when solving nodes in which they are feasible.

3.5.4 Heuristic Solutions

After employing the LEH heuristic that gives a set of progressively better feasible solutions, we exploit the root node optimal solution to construct heuristic solutions using the concept of relaxation induced neighborhoods $RINS$ of Danna et al. (2005). RINS is a versatile procedure that can be embedded easily in our scheme, and when the lower bound is strong, it tends to provide good quality feasible solutions. Specifically, we formulate the problem on the original space (A.1)–(A.5), select some $0.5 < l < 1$ and set $y_{it} = 1$ if $\bar{y}_{it} > l$, $y_{it} = 0$ if $\bar{y}_{it} < 1 - l$ and $y_{it} \in \{0, 1\}$ if $1 - l \leq \bar{y}_{it} \leq l$, where $(\bar{y}_{it})_{i \in I, t \in T}$ are the fractional setup variables obtained by column generation. We search aggressively for a feasible solution for 100 nodes and if we find one we update the incumbent. This is an efficient strategy, but it can be time-consuming if it is applied at every node. To account for this, we use it every 10 nodes, and employ a simple rounding heuristic at every node. The latter rounds the fractional setup variables to the closest integer and solves the resulting extended network flow problem in the continuous variables. It was observed that a strong lower bound at the root node usually leads to a high quality incumbent solution. This is in line with the theory developed in (Larsson and Patriksson 2006) that argues that heuristic solutions constructed by near-optimal Lagrange relaxations are also near-optimal.

3.6 Computational Experiments

The computational section aims to shed light on four aspects. First, it is necessary to investigate the trade-off between solution quality and CPU time by utilizing various combinations of subproblem sizes and horizon overlaps. To this end, we perform a full factorial experiment that delivers empirical insights on which configurations are efficient for which classes of problems. Second, it is interesting to assess a heuristic implementation of horizon decomposition against existing heuristic approaches. We find that a suitable horizon decomposition gives better lower bounds in almost all cases, while in about half of them also delivers equal or better upper bounds. Third, we benchmark an implementation of horizon decomposition against the period decomposition developed

in Chapter 2 and other competitive approaches. Since the Horizon decomposition is an exact rather than a heuristic implementation, we employ a heuristic variant to compare with the period decomposition, which is of heuristic nature. Finally, we construct a new hard dataset and benchmark our implementation against the branch and cut solver CPLEX 12.2. All formulations were coded in C++ and the mixed integer and linear programs were solved with CPLEX 12.2. We use a common subproblem size and horizon overlap, i.e., $H_u = H$ and $L_u = L$ for all $u \in U$ with only possible exception the last subproblem, which consists of the remaining periods till the end of the horizon. Experiments were run on a Quad Core Intel i7 2.70GHz with 8.00 GB RAM.

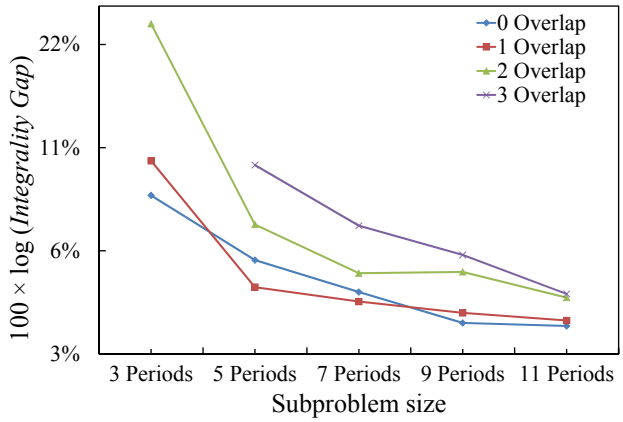
3.6.1 Subproblem Length and Overlap

The usefulness of a horizon decomposition depends heavily on the subproblem size and on the horizon overlap. Long horizon subproblems have the potential to lead to an improved lower bound, but it may be time consuming to solve these to optimality. Likewise, large horizon overlaps can also lead to improved lower bounds, but render the master problems degenerate and amplify the tailing-off effect (Vanderbeck 2005). This section describes the computational experiments we performed to assess which configurations of horizon decompositions are efficient in solving challenging CLST problems. The criterion used to assess efficiency is the integrality gap, while time efficiency is measured by average CPU time.

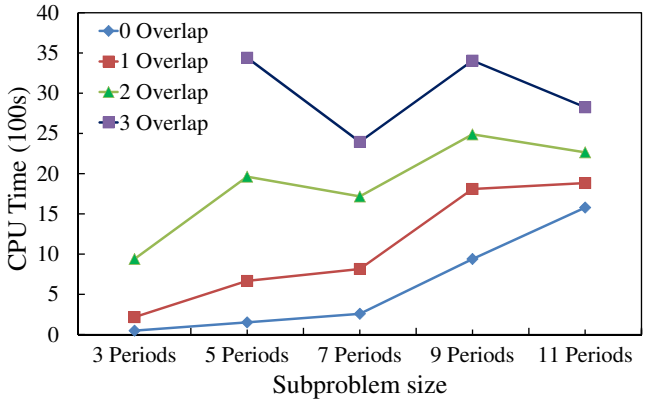
Data Instances. We generated new instances to test our approach. Since the main focus of the paper is problems with few items and long horizons, we generated instances with 2, 6 and 10 items and 15, 30 and 60 periods respectively. We used the problems G30 and G30b from Trigeiro, Thomas and McClain (1989) which have 6 items and 15 periods as follows. First, we created instances with 30 and 60 periods by replicating the demand of each item, and made the capacity constraints harder, so that the average lot for lot capacity utilization was about 120%. Using this utilization level we generated new instances with 2 and 10 items and 15, 30 and 60 periods. Two instances were generated for problems with 15 periods and 4 instances for problems with 30 and 60 periods. This procedure led to the creation of a total of 30 new instances. In addition, we generated 60 more instances with the same average lot for lot utilization and uncorrelated item demands sampled from normal distributions with the same mean and standard deviation. In total, 10 instances for each (item,period) combination were

utilized. Problems with high capacity utilization are usually challenging to solve in practice, and therefore constitute a good testbed for our approach.

Subproblem Length and Overlap. The first round of experiments aimed at identifying the influence of subproblem length and horizon overlap on the integrality gap and on CPU time. To this end, figures 3.2(a) and 3.2(b) show the average integrality gap, calculated using the best upper bound found and the root node lower bound in all trials for each instance, and the average CPU time respectively.



(a) Subproblem Length vs Integrality Gap



(b) Subproblem Length vs CPU Time

Figure 3.2: Sensitivity analysis for subproblem size and overlap length. Each point denotes the average measure obtained from 90 instances.

Some useful preliminary insights can be drawn from these figures. With respect to

the integrality gaps, large subproblems lead to good quality solutions. However, more periods of overlap do not necessarily lead to smaller integrality gaps. This happens because a large number of overlapping periods renders the restricted master problems very degenerate and degrades the algorithmic convergence, therefore column generation may fail to terminate before the imposed time limit. In these cases, the obtained intermediate lower bound can be weaker than that obtained with fewer periods of overlap. The impact of subproblem size seems higher when no overlap exists, and is minimal with three periods of overlap. Interestingly, one period of overlap leads to an important reduction of integrality gaps, whereas a second period of overlap offers no improvement within the given time limit. Therefore, it seems that one period of overlap with a medium subproblem size, such as seven periods, constitutes a good configuration. Considering CPU times, there is an evident interaction between subproblem size and overlap length, which is revealed in configurations with two or three periods of overlap. Specifically, larger overlaps and subproblems lead to higher CPU times in general, but there are cases where small subproblems combined with large overlaps lead to poor column generation convergence and thereby high CPU times. This is the case for five period subproblems combined with two or three periods of overlap. On the one hand, larger subproblems imply fewer linking constraints for a given overlap and therefore better convergence, but on the other hand it may be time consuming to solve them to optimality. This evidence, combined with the marginal gap increase from the inclusion of a third period of overlap, suggests that a third overlapping period may not lead to an efficient computation scheme.

On a more detailed analysis, it is useful to investigate which horizon configuration achieves the best performance in which instances. Table 3.1 displays a breakdown of the configurations that deliver the best performance in each instance category, characterized by the number of items and the number of periods.

Table 3.1 shows a consistent pattern with respect to gap quality and CPU time performance. Non-overlapping horizons induce better convergence behavior of column generation and therefore lead to faster termination. In terms of integrality gaps it is evident that subproblem horizons of 11 periods achieve the best performance in most cases. It is worth noticing however, that as figure 3.2 demonstrates, the gain in gap quality is often marginal. For instances with 10 items and 60 periods, configuration

Table 3.1: Configurations $(|H|, |L|)$ that give the best average Gap and CPU time for each problem category.

$\|I\|$	$\|T\|$=15		$\|T\|$=30		$\|T\|$=60	
	Gap	CPU Time	Gap	CPU Time	Gap	CPU Time
2	(9,3)	(11,0)	(11,1)	(3,0)	(11,3)	(7,0)
6	(11,2)	(3,0)	(11,3)	(3,0)	(11,3)	(3,0)
10	(11,3)	(3,0)	(11,1)	(3,0)	(7,2)	(3,0)

(11,3) induces slow convergence behavior and the average gap after 20,000 seconds of CPU time is not better than that of (7,2).

We use insights from this preliminary experiment to select appropriate horizon configurations in our subsequent experiments. In particular, when the number of items is small, it is beneficial to utilize covers with relatively large subproblems and overlaps. As the number of items increases it is preferable to select smaller subproblems and overlaps. Finally, the horizon decomposition quality seems to be relatively insensitive to the horizon length of the original problem. Specifically, it was observed that small integrality gaps were obtained for problems with long horizons, with a relatively small increase in CPU time.

3.6.2 A Heuristic Implementation

In some production planning environments it is useful to employ heuristics that find solutions of guaranteed quality in a short amount of time. To this end, we utilize our approach to tackle the instances introduced by Süral et al. (2009) and tested by Müller, Spoorendok and Pisinger (2012). The later authors employ an adaptive large scale neighborhood search heuristic (ALNS) which also injects feasible solutions to CPLEX to obtain lower bounds, with a time limit of 300 seconds. They construct problems without setup cost by concatenating horizons of some instances of Trigeiro, Thomas and McClain (1989). Their problems have 12 and 24 items, and although the aim of our approach is to tackle problems with large horizons and few items, it is interesting to investigate its performance on datasets with many items. In our implementation, we stop column generation after 300 or 600 seconds and apply the relaxation induced neighborhood heuristic, which then runs for at most 50 seconds. When column generation is not complete at the root node, we use the best lower

bound obtained by Lagrange relaxation. During preliminary experimentation, it was found that the insights of the previous section carry forward to the instances of Süral et al. (2009). Configurations with no horizon overlaps converge faster, and tend to give better lower bounds compared to similar configurations with overlap in the given time limit. As our objective is to efficiently generate both lower and upper bounds, we do not use overlapping horizons in our heuristic implementation. For instances with 30 periods, we use two subproblems of size 20 and 10 periods respectively. For all other instances, we use a subproblem length of 30 periods, with only possible exception the last subproblem which accommodates the remaining periods. Finally, following Süral et al. (2009), we can distinguish the instances in those with unit holding cost, called homogeneous and those with variable holding costs, called heterogeneous. Table 3.2 shows the integrality gaps obtained by ALNS and horizon decomposition after 300 (HD300) and 600 (HD600) seconds respectively.

Table 3.2: Comparison of Horizon Decomposition and ALNS (Müller, Spoorendok and Pisinger 2012).

Average integrality gap (%)											
			Homogeneous			Heterogeneous			Total		
k	$\|I\|$	$\|T\|$	ALNS	HD300	HD600	ALNS	HD300	HD600	ALNS	HD300	HD600
5	12	30	10.01	1.91	0.59	24.7	11.01	7.49	17.36	6.46	4.04
4	24	30	6.11	8.10	5.75	6.11	7.07	5.56	6.11	7.59	5.66
5	12	45	10.87	5.32	4.29	23.01	12.18	10.45	16.94	8.75	7.37
4	24	45	7.00	7.87	5.23	7.02	6.20	5.94	7.01	7.04	5.58
5	12	60	20.78	13.79	12.64	23.20	20.15	18.16	21.99	16.97	15.40
5	24	60	22.26	24.35	22.87	26.63	29.60	24.67	24.44	26.97	23.77
5	12	90	22.57	17.27	13.15	25.67	20.96	19.87	24.12	19.12	16.51
5	24	90	25.99	25.56	20.39	29.21	31.81	29.02	27.60	28.69	24.70
Average (%)			16.18	13.29	10.88	21.02	17.94	15.64	18.60	15.61	13.6
p-value (%)				0.247	0.001		.526	0.162		0.01	0.000

k denotes the number of instances of size $(|I|, |T|)$ that were utilised. Four instances were excluded from the dataset because their lower and upper bounds reported by Müller, Spoorendok and Pisinger (2012) were inconsistent.

The comparison suggests that HD delivers better integrality gaps, both for homogeneous and heterogeneous instances. Specifically, the paired t-tests indicate that

the average gaps difference is significant at the 1% level. The performance of HD300 is much better for 12 item instances, whereas it remains competitive for 24 item instances. Gaps for the homogeneous 24-60 instances are sometimes better for ALNS because it is able to find better feasible solutions. Detailed analysis reveals that HD300 found a better lower bound in 32/38 homogeneous and 35/38 heterogeneous instances. On upper bounds, HD is competitive, since in 18/38 and 20/38 homogeneous and heterogeneous instances the upper bound is at least as good as that of ALNS. Overall, the integrality gap for HD300 was better in 51/76 instances. It is notable that the HD heuristic improves its performance when the time limit is extended to 600 seconds. Specifically, HD600 found a better lower bound in 72/76 instances (37/38 homogeneous) and a better or equal upper bound in 61/78 instances (24/38 homogeneous).

It is of interest to investigate how the number of items and the horizon length influence the performance of both heuristics. Figure 3.3 shows the integrality gap of ALNS and HD300 in various problem configurations. A first observation is that both algorithms show similar behavior, but HD is much better in 12 item instances, while it remains competitive in 24 item instances. In addition, it is evident that problems with longer horizons are much harder, regardless of the number of items. Interestingly, problems with 24 items have smaller gaps than problems with 12 items with short horizons, but the opposite is true for long horizons. Also, 12 item problems scale better with longer horizons, in the sense that their gaps increase linearly, whereas there is a quadratic increase for 24 item problems. Therefore, a generic conclusion is that extending the problem horizon has a more profound effect to instances with relatively large $|I| / |T|$ ratio.

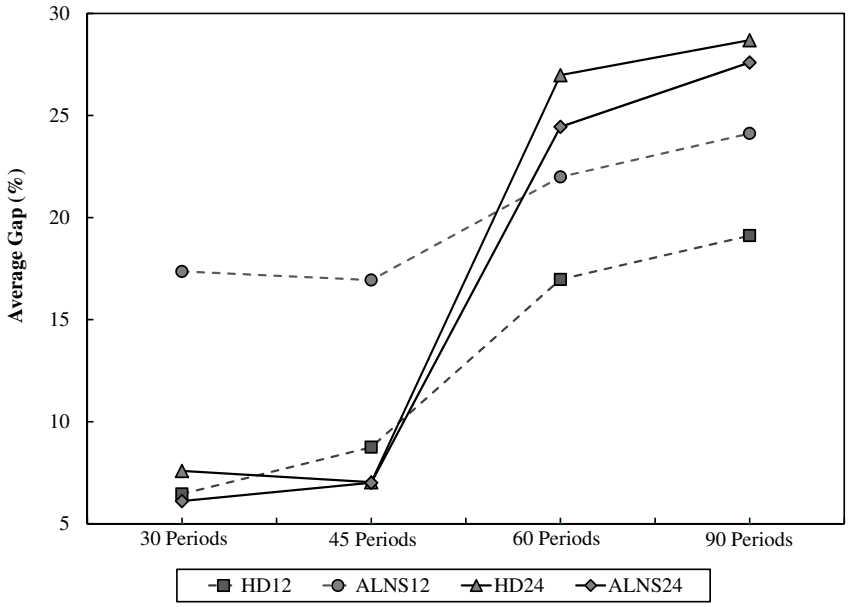


Figure 3.3: Performance sensitivity of ALNS and HD

3.6.3 Comparison with other approaches

Trigeiro. We consider the 7 instances of Trigeiro, Thomas and McClain (1989) that have been used by several other authors to assess the relative strength of the lower and upper bounds of our approach. Table 3.3 compares the lower bound obtained by the horizon (HD), item (DJ) and period decompositions (HB&P), and the approximate extended formulation (AEF) approach of Van Vyve and Wolsey (2004), while table 3.4 compares the upper bound obtained by the HD, DJ, JD and the large-scale neighborhood search approach (MSP) of Müller, Spoorendok and Pisinger (2012).

Table 3.3: Lower bound for the horizon (HD), period (HB&P), item decomposition (DJ) and approximate extended formualtions.

		Lower Bounds			
Instance	$(\|H\|, \|L\|)$	HD	DJ	HB&P	AEF
G30 (6-15)	(10,1)	37,796	37,103	37,431	–
G30b (6-15)	(10,1)	37,705	37,201	37,382	37,469
G53 (12-15)	(8,1)	74,433	73,848	73,945	74,230
G57 (24-15)	(8,0)	134,184	136,366	136,418	136,417
G62 (6-30)	(10,2)	61,708	60,946	61,204	61,294
G69 (12-30)	(10,0)	127,190	130,177	130,338	130,335
G72 (24-30)	(15,0)	284,554	287,753	287,824	287,828

Time limit for the horizon decomposition: 3,600 seconds.

Table 3.4: Upper bound for the horizon (HD), period (HB&P), item decomposition (DJ) and Large-scale variable neighborhood search (MSP).

		Upper Bounds			
Instance	$(\|H\|, \|L\|)$	HD	DJ	HB&P	MSP
G30 (6-15)	(10,1)	37,809	37,809	37,809	–
G30b (6-15)	(10,1)	37,721	38,162	37,721	37,776
G53 (12-15)	(8,1)	74,644	75,035	74,634	74,720
G57 (24-15)	(8,0)	140,897	136,860	136,509	136,675
G62 (6-30)	(10,2)	61,746	62,644	61,746	61,792
G69 (12-30)	(10,0)	134,717	130,675	130,599	130,675
G72 (24-30)	(15,0)	298,656	288,393	287,950	287,966

Time limit for the horizon decomposition: 3,600 seconds.

The comparison of upper and lower bounds confirms the conclusions of our previous experiments. Specifically, the horizon decomposition gives excellent lower bounds for problems with a relatively small number of items. In particular, the decompositions we consider give the strongest known lower bounds for 4 out of 7 instances. For these instances, a strong upper bound is also obtained, because the RINS heuristic is guided in the neighborhood of the optimal solution. For instances with weak lower bound however, the upper bound is equally weak. It is worth noticing that the period decomposition studied in chapter 2 gives consistently strong lower and upper bounds, but it

cannot match the lower bound quality of the horizon decomposition for instances with a small number of items. The next paragraph consider the instances of Süral et al. (2009).

Süral. Table 3.5 shows the CPU time and integrality gap for the horizon decomposition (HD) the branch-and-price heuristic developed in chapter 2 (B&P) and the heuristic of Süral et al. (2009) (SDW). A first conclusion is that although horizon decomposition requires more CPU time, the integrality gaps it delivers are usually a lot better than those of SDW and B&P. An exception to this rule seems to be the homogeneous set of instances with 24 items and 30 periods. For all other categories horizon decomposition gives better gaps, and in some extreme cases, such as the 12 items 30 periods heterogeneous instances, the integrality gap is nine times smaller compared to the best obtained by both SDW and B&P.

Table 3.5: CPU times (s) and integrality gaps (%) for the Horizon Decomposition (HD), the period decomposition (HB&P) and the heuristic of Süral et al. (2009) (SDW).

	CPU Time (s)			Gap (%)		
Instance	HD	B&P	SDW	HD	B&P	SDW
12 x 10 het	17.5	0.3	3.1	18	32	2
24 x 10 het	88.3	1.8	4.8	11	18	2
12 x 15 het	83.3	0.8	6.1	20	26	5
24 x 15 het	142.6	6.4	15.3	13	21	8
12 x 30 het	151.8	3.3	24.0	22	29	19
24 x 30 het	153.3	19.7	38.9	23	32	25
12 x 10 hom	72.3	0.5	2.7	23	42	4
24 x 10 hom	144.8	2.0	4.5	14	21	8
12 x 15 hom	141.0	1.0	5.6	20	28	9
24 x 15 hom	150.9	3.5	11.1	15	21	12
12 x 30 hom	151.8	3.0	19.1	22	24	23
24 x 30 hom	151.1	8.0	22.9	21	30	53
Average het	106.1	5.4	15.4	18	26	10
Average hom	135.3	3.0	11.0	19	28	18
Average	120.7	4.2	13.2	19	27	14

Time limit: 150s. HD uses 0 overlap and 5, 10 and 15 length for 10, 15 and 30 period problems respectively.

Period decomposition heuristic and branch-and-cut. Table 3.6 compares the performance of the commercial solver CPLEX 12.1 (CPLEX), the Period decomposition of chapter 2 (B&P) and the Horizon Decomposition. A time limit of 600 seconds was imposed on all approaches. Columns 2 to 4 show the integrality gap calculated by using the best upper bound of each instance, while columns 5 to 7 shows the integrality gaps calculated by using the best lower bound of each instance. Therefore columns 2 to 4 provide a measure of the strength of the lower bound, and columns 5 to 7 a measure of the strength of the upper bound. It seems that although HD provides a competitive upper bound in most cases, the specialised hybrid procedure of chapter 2 can deliver better results. Also notable is that the performance of CPLEX in terms of lower bounds is competitive with respect to the HD but inferior of the performance of B&P.

Table 3.6: CPLEX, Period decomposition (B&P) and Horizon Decomposition (HD) lower, upper bounds and integrality gaps.

	Gap LB (%)			Gap UB (%)			Gap (%)		
Instance	CPLEX	BnP	HD 600	CPLEX	BnP	HD 600	CPLEX	BnP	HD 600
G7	2.13	1.79	2.94	2.16	3.06	1.79	2.6	3.2	3.0
G11	0.85	0.80	1.61	0.80	0.92	1.02	0.9	0.9	1.9
G16	2.31	2.78	3.04	2.57	3.34	2.31	2.6	4.0	3.1
G17	6.42	4.73	5.45	5.65	6.13	4.73	7.9	6.5	5.8
G21	3.41	3.10	3.55	3.16	3.84	3.10	3.6	4.0	3.7
G22	1.69	1.55	2.78	1.73	1.71	1.55	1.9	1.7	2.9
G24	5.82	5.37	6.67	8.00	8.15	5.37	9.2	8.9	7.2
G26	2.22	1.73	2.58	2.43	3.91	1.73	3.0	4.1	2.7
G27	2.21	2.12	3.17	2.23	3.31	2.12	2.4	3.4	3.3
G28	2.04	1.76	2.05	2.41	3.18	1.76	2.8	3.3	2.1
G30	1.44	1.42	2.73	1.72	1.42	1.79	1.8	1.4	3.2
G33	0.12	0.09	0.88	0.09	0.16	0.09	0.1	0.2	0.9
G34	0.41	0.05	0.90	0.05	0.08	0.46	0.4	0.1	1.3
G36	0.29	0.28	0.78	0.28	0.30	0.53	0.3	0.3	1.0
G37	0.70	0.25	0.12	0.12	0.16	0.12	0.7	0.3	0.1
G40	0.25	0.23	0.81	0.23	0.35	0.28	0.3	0.3	0.9
G51	1.99	1.94	2.81	2.20	1.94	2.41	2.3	2.0	3.4
G52	0.62	0.57	3.13	0.57	0.71	4.73	0.6	0.7	7.7
G54	0.92	0.96	2.49	0.92	1.28	10.71	0.9	1.3	13.8
G55	2.91	2.48	4.14	3.48	2.48	3.61	4.1	2.5	5.5
G61	2.91	2.39	3.25	2.74	3.08	2.39	3.4	3.2	3.4
G62	1.87	1.75	1.90	1.78	1.84	1.75	1.9	1.9	1.9
G63	1.84	1.73	3.20	1.73	1.97	1.86	1.9	2.0	3.4
G64	1.16	0.96	1.59	1.12	2.22	0.96	1.3	2.3	1.6
G65	1.18	0.82	1.67	1.06	1.53	0.82	1.4	1.6	1.7
G66	1.74	1.07	3.76	1.23	1.07	5.02	1.9	1.1	8.2
G67	1.39	1.23	2.95	1.23	1.72	3.76	1.4	1.8	5.7
G68	1.71	1.56	3.65	1.56	1.57	4.45	1.7	1.6	6.9
G69	0.78	0.70	3.64	0.70	1.41	6.96	0.8	1.4	10.8
G70	2.02	1.90	3.16	2.48	1.90	2.24	2.7	1.9	3.6
Average	1.85	1.60	2.71	1.88	2.16	2.68	2.22	2.26	4.02

Subproblem size $|H| = 12$, horizon overlap $|L| = 2$ for i = 6, $|H| = 10, |L| = 0$ for $i = 12$.

Pimentel. At a final round of experiments, we show how horizon decomposition compares to period decomposition and to the simultaneous decomposition approach of Pimentel et al. (2009). To this end, table 3.7 shows the CPU time and integrality gaps of the tree approaches. The gaps show that horizon decomposition is quite competitive in some instances, considering that the imposed time limit was only 120 seconds. In this case, the period decomposition approach seems to be the winner, as it delivers high quality gaps in a small amount of CPU time.

A generic conclusion from the comparison of the horizon decomposition with other approaches is that its performance is superior for instances with a small number of items. In addition, an exact implementation of a branch-and-price scheme might be more promising than a heuristic implementation, because column generation requires a significant amount of time to converge. The next paragraphs shows how a branch-and-price based horizon decomposition compares against branch-and-cut.

3.6.4 Comparison with branch and cut

At a final round of experiments, we compared the horizon decomposition against CPLEX v12.2. The purpose of this comparison is to investigate whether a horizon decomposition approach delivers competitive results against a state-of-the-art commercial solver for certain classes of problems. Since our algorithm uses CPLEX to solve the subproblems, the interpretation of our results should be that in some classes of hard problems, it is more efficient to use branch-and-cut technology within a carefully selected branch-and-price horizon decomposition rather than as a stand-alone solver. Since the suggested methodology delivers a lower bound, the main focus on our experiments is the strength of the lower bound obtained by each approach. However, we also assess the final integrality gap by taking into account the best feasible solution that each method finds.

Data. We focus on problems with small items to periods ratios, since they seem to be the most challenging ones, (Müller, Spoorendok and Pisinger 2012). Specifically, we generated sets of 10 problem instances, each with 2, 4, 6, 8 and 10 items and 100 periods. In total, 50 new problems were constructed. The average capacity utilization was 120%, with some instances that need initial inventory for feasibility. To the best of our knowledge, this is the first dataset that includes instances that need initial inventory. While it is well-known that high capacity utilizations characterize hard problems, it

is usually the case that the resulting dataset is infeasible without initial inventory. Trigeiro, Thomas and McClain (1989), who constructed the most widely used CLST dataset write: "Rather than solve the NP complete feasibility test for each problem, we simply threw out problems for which no feasible solution was found by the heuristic. (...) This results in an unavoidable and unmeasurable bias in problem generation. It occurs mostly for tightly constrained problems". Since then, the assessment of this class of problems has been neglected. Figure 3.4 graphs the average integrality gap of CPLEX and the number of nodes explored in 3600 seconds against the number of items, $|I|$.

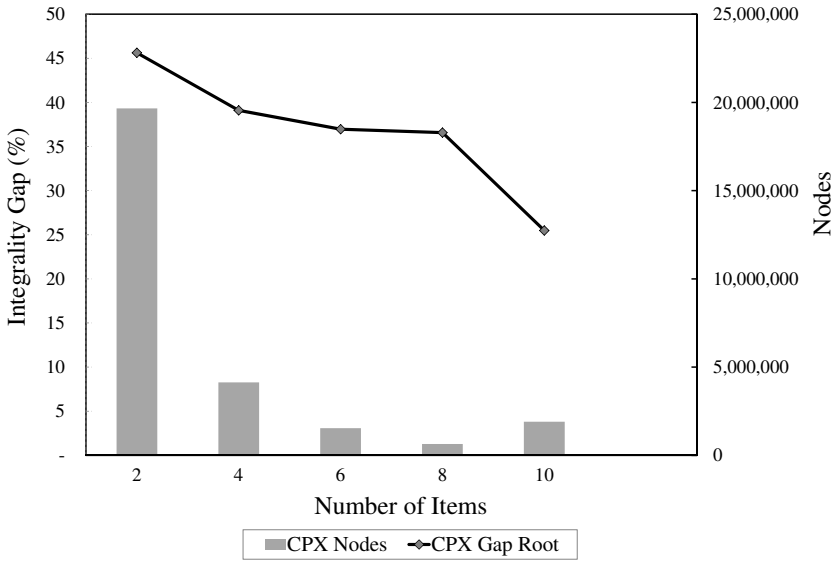


Figure 3.4: Average integrality gap and number of nodes explored by CPLEX. Time limit is 3600 seconds.

Three useful conclusions can be drawn. First, the integrality gaps of the root node are between 25% and 45%, suggesting that solving these instances to within an acceptable tolerance may be challenging. To put these numbers in perspective, for the seven instances from Trigeiro's G dataset that are supposed to be among the hardest (Van Vyve and Wolsey 2004), CPLEX 12.2 has an average integrality gap of 2.54% at the root node, and needs an average 60,000 nodes and 200 seconds to solve them to optimality. Moreover, for the instances examined in Müller, Spoorendok

and Pisinger (2012) and Süral et al. (2009) the average gaps vary between 14% and 20%. A second observation is that the average gaps become smaller as the number of items increases. This is in line with earlier empirical (Trigeiro, Thomas and McClain 1989) and theoretical (Manne 1958) evidence. Finally, the average number of nodes explored generally decreases as the number of items increases, since the linear programs at each node become larger. With these conclusions at hand, it is useful to explore the performance of a horizon decomposition approach.

Horizon Decomposition and Branch and Cut. Table 3.8 shows the relative performance of CPLEX and Horizon Decomposition at the aforementioned problems.

The conclusion of the above result is that horizon decomposition constitutes a promising approach, particularly for problems with only few items. It is worth noticing that although CPLEX explores many more nodes and therefore it is more likely to find good heuristic solutions, the final average gaps are in favor of horizon decomposition. This is explained by the large amount of gap that is closed by our algorithm. Finally, the maximum benefit of horizon decomposition is unfolded in instances with two items. Specifically, we were able to solve to optimality 7 out of 10 instances, 3 of which at the root node, with an average of 660 seconds of CPU time. For the same instances, CPLEX obtained an average gap of 6.6% after one hour of CPU time.

3.7 Generalizations

In this section we demonstrate potential generalizations of the horizon decomposition approach. First, we present an extension that stems naturally from our work in the CLST that fits well to problems with sparse constraint matrices. Then, we consider an alternative approach that is deemed more appropriate for problems with dense constraint matrices. Both methods are applicable to generic mixed integer linear programs, which we consider in the form below.

$$[\mathbf{P}] \qquad \min \quad c^T x \tag{A.32}$$

$$\text{s.t.} \quad Ax = b \tag{A.33}$$

$$x \in X \tag{A.34}$$

The set X describes trivial restrictions such as integralities and range bounds on single variables. We let $I := \{1, \ldots, c\}$ be the variable index set and $R := \{1, \ldots, r\}$ be the

row index set. For notational simplicity, we interpret indexing of a vector or matrix over a set as a reference to the quantities defined over this set. We show that each of the following generalizations takes advantage of different structural characteristics in order to decompose the problem efficiently.

3.7.1 Extension of the Horizon Decomposition Principle: Row Partitioning

The essence of horizon decomposition is about creating copies of variables that are in multiple constraints in such a way that the problem matrix is decomposed. We partition the row index set R into two mutually exclusive and exhaustive sets R^1 and R^2, i.e. $R = R^1 \cup R^2$ and $R^1 \cap R^2 = \emptyset$. The extension to more sets, and also the case with $R^1 \cap R^2 \neq \emptyset$ are straightforward and are omitted to ease the exposition. It is of interest to identify which variable indexes are common in sets R^1 and R^2. We define $\bar{V}_s = \{i \in I : \exists j \in R^s \text{ with } a_{ij} \neq 0\}$ for $s = 1, 2$, $V = \bar{V}_1 \cap \bar{V}_2$ and $V_s = \bar{V}_s \backslash V$ for $s = 1, 2$. Using this notation and selecting a $\lambda \in (0, 1)$, we can recast problem **[P]** as follows:

$$[\mathbf{P_1}] \qquad \min \quad c_{V_1}^T x_{V_1} + c_{V_2}^T x_{V_2} + \lambda c_V^T x_V^1 + (1-\lambda) c_V^T x_V^2 \tag{A.35}$$

$$\text{s.t.} \quad A_{R^1 V_1} x_{V_1} + A_{R^1 V} x_V^1 = b_{R^1} \tag{A.36}$$

$$A_{R^2 V_2} x_{V_2} + A_{R^2 V} x_V^2 = b_{R^2} \tag{A.37}$$

$$x_V^1 - x_V^2 = 0_V \tag{A.38}$$

$$x_{V1} \in X_{V1} \quad x_{V2} \in X_{V2} \quad x_V^1, x_V^2 \in X_V \tag{A.39}$$

[P] has structure that is amenable to Dantzig-Wolfe decomposition, and it constitutes a generalization of the lot-sizing formulation $(A.6 - A.14)$ presented in section 3.2. Specifically, in our application sets R^1 and R^2 capture rows indexed over periods $\{1, \ldots, k\}$ and $\{l, \ldots, m\}$ respectively for some $k, l \in T$ with $l \leq k + 1$. Set V_1 captures the indexes of variables found exclusively in the first subproblem, i.e., $(x_{i1}, y_{i1}, s_{i1}, \ldots, x_{il-1}, y_{il-1}, s_{il-1})$ for each item $i \in I$, and similarly set V_2 those found only in the second subproblem. Set V models the indexes of variables defined over the overlap $(x_{il}, y_{il}, s_{il}, \ldots, x_{ik}, y_{ik}, s_{ik}, s_{ik+1})$ for each item $i \in I$.

In classes of problems where the constraint matrix has an obvious block diagonal substructure, implementing a row partition is relatively straightforward: one has to

regulate the subproblem size and horizon overlap based on empirical data. For problems with sparse matrices but no obvious structure, an issue that arises naturally in formulating $[\mathbf{P_1}]$ is that of row partition selection. There is a stream of literature that considers the problem of rearranging the constraint matrix in such a way that it exhibits a block-triangular substructure. To the best of our knowledge, Martin (1999) was the first to formulate the problem of rearranging a matrix to decomposable format as a MIP. Specifically, he introduced the matrix decomposition problem as that of decomposing a matrix in bordered diagonal form, given the number of blocks and the size of each block. The recent work of Bergner et al. (2011) formulates the same problem as a hypergraph partitioning problem. Moreover, they use the resulting solution within an automatic Dantzig-Wolfe reformulation approach and show experimentally that their approach delivers high quality dual bounds for some challenging MIPLIB 2003 and MIPLIB 2010 instances. A key difference with our approach is that the algorithm they devise tries to identify hidden structures and come up with one decomposition, whereas the horizon decomposition utilizes a family or reformulations, from which the user can select the most suitable one for any particular instance. Although our formulation of $[\mathbf{P_1}]$ does not require any structure, one can use the two aforementioned approaches to construct decomposable index sets and decide on the size of the overlaps accordingly.

3.7.2 An Alternative Generalization: Column Partitioning

The partitioning approach developed in the previous section is well-suited to problems with sparse constraint matrices. This is because the number of linking constraints is small and the column generation process exhibits better numerical properties (Bergner et al. 2011). This section aims to present an alternative formulation that is well-suited to problems with dense constraint matrices, such as set-partitioning problems. Again, we let I be the variable index set, and consider variable index sets H_1, H_2 and L such that $H_1 \cup H_2 = I$ and $H_1 \cap H_2 = L$. The extension to more partitions is straightforward. For ease of notation, let x_i, c_i, A_i and X_i denote the components of each entity that refer to indexes in set H_i, and x_l, c_l, A_l the components of each entity that refer to indexes in set L. Then $[\mathbf{P}]$ can be reformulated such that it is amenable to Dantzig-Wolfe decomposition as follows.

$$[\mathbf{P_2}] \qquad \min \quad c_1^T x_1 + \bar{c}_2^T x_2 \tag{A.40}$$

$$\text{s.t.} \quad A_1 x_1 - \lambda A_l x_l^1 + s_1 = b/2 \tag{A.41}$$

$$A_2 x_2 - (1-\lambda) A_l x_l^2 - s_2 = b/2 \tag{A.42}$$

$$x_l^1 = x_l^2 \tag{A.43}$$

$$s_1 = s_2 \tag{A.44}$$

$$x_1 \in X_1, x_2 \in X_2, x_l^1, x_l^2 \in X_l, s_1, s_2 \in \mathbb{R}^r \tag{A.45}$$

The variables s_1 and s_2 are continuous and their dimension equals the number of rows of matrix A. Also, λ is a fixed scalar and $\bar{c}_{2i} = 0$ if $i \in L$, and $\bar{c}_{2i} = c_{2i}$ otherwise. By dualizing constraints (A.43) and (A.44) [$\mathbf{P_2}$] decomposes in two subproblems, defined over the index sets H_1 and H_2 respectively. This decomposition can be beneficial for problems with large number of variables and relatively few constraints, or problems that exhibit structure over index sets of variables. The issue of selecting a suitable partition is relevant in this formulation as well. One needs to partition the variables in such a way that those with similar row coefficients belong to the same index set H. To the best of our knowledge, this problem has not been tackled in the literature, but variants of the methods of Martin (1999) and Bergner et al. (2011) can also be applied for column partitioning. The effectiveness of such methods remains to be explored and benchmarked against alternative approaches.

3.8 Conclusions and Future Research

We present a horizon decomposition approach and its implementation to the capacity constrained lot size problem with setup times. The problem is decomposed in contiguous horizons of smaller size, and the subproblems are of the same type as the original, but have smaller dimension. The developed methodology suggests a family of reformulations, which offer flexibility to regulate the master problem and subproblem size almost independently. A computational study gives empirical evidence on which configurations lead to the selection of the most efficient reformulation, in the context of capacitated lot sizing problems. Computational results show that the approach delivers strong lower bounds, while it outperforms the best heuristic described in the literature. Further experiments on generated datasets show its competitive performance against the branch and cut solver CPLEX v12.2. Finally, we show how horizon decomposi-

tion can be generalized, and how it can be used to take advantage of certain problem structures.

3.9 References

Araujo, S., B. De Reyck, Z. Degraeve, I. Fragkos, R. Jans. Period Decompositions for the capacity constrained lot-sizing problem with setup times. Working Paper, 2011.

Akartunali, K., A. J. Miller, Ioannis Fragkos, Tao Wu. Local cuts and two-period convex hull closures for big-bucket lot-sizing problems. Working paper, 2013.

Barahona, F., D. Jensen. Plan location with minimum inventory. *Math. Prog.,* 83(1):101–111, 1998.

Barany, I., T. J. van Roy, L. A. Wolsey. Uncapacitated lot-sizing: the convex hull of solutions. *Math. Prog. Stud.,* 22(1):32–43, 1984.

Barnhart, C., E.L. Johnson, G. L. Nemhauser, M. W. P. Savelsbergh. Branch-and-price: column generation for solving huge integer programs. *Op. Res.,* 46(3):316–329, 1998.

Belvaux, G. L. A. Wolsey. Modeling practical lot–sizing problems as mixed–integer programs. *Management Sci.,* 47(7):993–1007, 2001.

Belvaux, G., L. A. Wolsey. Bc-prod: a specialized branch-and-cut system for lot-sizing problems. *Management Sci.,* 46(5):724–738, 2000.

Ben Amor, H., Desrosiers, J., J. M. Valério de Carvalho. Dual optimal inequalities for stabilized column generation. *Op. Res.,* 54(3):454–463, 2006.

Bergner, M., A. Caprara, F. Furini, M. E. Lübbecke, E. Malaguti., E. Traversi. Partial convexification of generic MIPs using Danzig-Wolfe reformulation. *Integer Programming and Combinatorial Optimization (IPCO 2011)* 6655 39–51, 2011.

Bitran, G. R., H. H. Yannasse. Computational complexity of the capacitated lot size problem. *Management Sci.,* 28(10):1174–1186, 1982.

Caprara, A., M. Fischetti, P. Toth. A heuristic method for the set covering problem. *Op. Res.,* 45(5):730–743, 1999.

Cattrysse, D., Maes., J., L. N. Van Wassenhove. A dual ascent and column generation heuristic for the discrete lot-sizing and scheduling problem with setup times. *Management Sci.,* 39(4):477–486, 1993.

Chen, W. H., J. M. Thizzy. Analysis of relaxations of the multi-item capacitated lot-sizing problem. *Annals of Op. Res.,* 26(1):29–72, 1990.

Danna, E., E. Rothberg, C. Le Pape. Exploring relaxation induced neighborhoods to improve MIP solutions. *Math. Prog.,* 102(1):71–90, 2005.

Dantzig, G. B., P. Wolfe. Decomposition principle for linear programs. *Op. Res.,* 8(1):101–111, 1960.

Degraeve, Z. R. Jans. A new danzig-wolfe reformulation and branch-and-price algorithm for the capacitated lot-sizing problem with setup times. *Op. Res.,* 55(5):909–920, 2007.

Degraeve, Z., M. Peeters. Optimal solutions to industrial cutting stock problems: Part 2, benchmark results. INFORMS J. Comput., 15(1):58–81, 2003.

Desaulniers, G. Branch-and-price-and-cut for the split-delivery vehicle routing problem with time windows. *Op. Res.,* 58(1):179–192, 2010.

Du Merle, O., D. Villeneuve, J. Desrosiers, P. Hansen. Stabilized column generation. *Disc. Mathematics,* 194(1–3):229–237, 1999.

Elhallaoui, I., D. Villeneuve, F. Soumis, G. Desaulniers. Agreegation of set-partitioning constraints in column generation. *Op. Res.,* 53(4):632–645, 2005.

Eppen, G., R. K. Martin. Solving multi-item capacitated lot-sizing problems using varibles redefinition. *Op. Res.,* 35(6):832–848, 1987.

Fisher, M. L. The lagrangian relaxation method for solving integer programming problems. *Management Sci.,* 27(1):1–18, 1981.

Geoffrion, A.M. Lagrangean relaxation for integer programming. *Math. Programming. Stud.,* 2(1):82–114, 1974.

Gilmore, P. C., R. E. Gomory. A linear programming approach to the cutting stock problem–Part II. *Op. Res.,* 11(6):863–888, 1963.

Guignard, M., S., Kim. Lagrangean decomposition: a model yielding stronger lower bounds. *Math. Programming,* 39(2):215–228, 1987.

Jans, R., Z. Degraeve. Improved lower bounds for the capacitated lot-sizing problem with setup times. *Op. Res. Letters,* 32(2):185–195, 2004.

Holmberg, K., D. Yuan. A lagrangean heuristic based branch-and-bound approach for the capacitated network design problem. *Op. Res.,* 48(3):461–481, 2000.

Kleindorfer, P. R., E. F. P. Newson. A lower bounding structure for lot-sizing problems. *Op. Res.,* 23(2):299–311, 1975.

Manne, A. S. Programming of economic lot sizes. *Management Sci.,* 4(2):115–135, 1958.

Martin, A. Integer programs with block angular structure. Phd Thesis, Konrad-Zuse-Zentrum für Informationstechnik Berlin, Preprint SC 99-03, 1999.

Michel, S., F. Vanderbeck. Column Generation based Solution for a Tactical Inventory Routing Problem. Research Report INRIA-00169311, Department of Applied Mathematics, University of Bordeaux, 2011.

Müller, L. F., S. Spoorendok, D. Pisinger. A hybrid adaptive large neighborhood search heuristic for lot-sizing with setup times. *Eur. J. Oper. Res.,* 218(3):614–623, 2012.

Larsson, T., M. Patriksson. Global optimality conditions for discrete and nonconvex optimization– with applications to lagrangian heuristics and column generation. *Op. Res.,* 54 (3):436–453, 2006.

Lübbecke, M., J., Desrosiers. Selected topics in column generation. *Op. Res.,* 53(6):1007–1023, 2005.

Pochet, Y., L. A. Wolsey. *Production Planning in Mixed Integer Programming*, Springer Series in Operations Research an Financial Engineering, 2006.

Pimentel, C. M. O., F. P. Alvelos, J. M. Valério de Carvalho. Comparing Danzig-Wolfe decompositions and branch-and-price algorithms for the multi-item capacitated lot sizing problem. *Opt. Meth. Soft.,* 25(2):299–319, 2009.

Süral, H., M. Denizel, L. V. Wassenhove. Lagrangean relaxation based heuristics for lot-sizing with setup times. *Eur. J. Oper. Res.,* 194(1):51–63, 2009.

Trigeiro, W., L. J. Thomas, J. O. McClain. Capacitated lot-sizing with setup times. *Management Sci.,* 35(3):353–366, 1989.

Van Vyve, M., L. A. Wolsey. Approximate extended formulations. *Math. Prog. Ser. B,* 105(2–3):501–522, 2004.

Vanderbeck, F. Branching in branch-and-price: a generic scheme. *Math. Prog. Ser. A,* 130(2):249–294, 2011.

Vanderbeck, F. Implementing Mixed Integer Column Generation. F. Vanderbeck. In G. Desaulniers, Desrosiers, J., and Solomon, M.M. (editors), *Column Generation*, Springer, pp 331–358, 2005.

Vanderbeck, F. Lot-sizing with start-up times. *Management Sci.,* 44(10):1409–1425, 1998.

Vanderbeck, F., M. W. P. Savelsbergh. A generic view of Danzig-Wolfe decomposition in mixed-integer programming. *Op. Res. Letters,* 34(3):296–306, 2006.

Villeneuve, D., J. Desrosiers, M. E. Lübbecke, F. Soumis. On compact formulations for integer programs solved by column generation. *Ann. Op. Res.,* 139(1):375-388, 2005.

Wagner, H. M., T. M. Whitin. Dynamic version of the economic lot size model. *Management Sci.,* 5(1):89–96, 1958.

Table 3.7: Integrality gaps (%) and CPU times (s) for the simultaneous decomposition of Pimentel et al. (2009), the period decomposition of chapter 2 (HB&P) and Horizon decomposition (HD).

	CPU Time (sec)			Gap (%)		
	PAV	HB&P	HD	GAP PAV	GAP HB&P	GAP HD
X11419	3,600	96	128	10.37	7.07	6.66
X11429	3,600	106	132	9.60	4.99	7.36
X12429	3,600	110	123	8.00	3.65	13.76
X12419	3,600	84	128	5.97	4.25	8.72
X11428	3,600	68	128	4.78	0.95	71.05
X12428	3,600	62	128	3.91	1.56	5.88
X12229	3,600	30	121	3.24	1.92	3.54
X11229	3,600	66	124	3.05	2.07	3.36
X12219	3,600	63	122	2.75	2.51	4.27
X11129	3,600	63	122	2.53	2.87	3.34
X11219	3,600	73	123	2.38	2.51	2.81
X11418	3,600	105	132	2.25	0.98	5.85
X12418	3,600	31	125	1.82	0.86	3.54
X12119	3,600	80	123	1.73	3.46	3.34
X11119	3,600	56	123	1.54	3.13	3.15
X12129	3,600	34	111	1.14	0.92	0.72
X12417	2,380	26	124	1.09	0.56	2.97
X11427	2,312	60	133	0.98	0.25	4.46
X12218	2,309	39	119	0.63	0.59	0.71
X12427	1,660	23	123	0.59	0.27	3.81
X11417	2,885	58	129	0.47	0.22	3.86
X12228	1,559	14	110	0.46	0.23	0.59
X11218	3,600	38	120	0.41	0.24	0.13
X11228	3,600	37	105	0.33	0.26	0.27
X12128	11	2	26	0.29	0.38	0.11
X12118	342	1	40	0.20	0.18	0
X12217	27	5	62	0.17	0.17	0.04
X12227	13	4	53	0.15	0.11	0.26
X11128	82	1	21	0.10	0.09	0
X11227	42	2	71	0.09	0.03	0.15
X11217	86	12	66	0.08	0.07	0.11
X12117	0	0	17	0.05	0.04	0
X11118	0	0	6	0.03	0.04	0
X12127	0	0	19	0.02	0.03	0

The horizon decomposition used $|\mathcal{S}| = 1, |H| = 11$ and a time limit of 120 seconds.

Table 3.8: Comparison of Horizon Decomposition and CPLEX.

	CPLEX		Horizon Decomposition			
$\|I\|$	Gap (%)	Nodes	$(\|H\|, \|L\|)$	Final Gap (%)	Gap Closed (%)	Nodes
2	6.73	19,662,721	(12,2)	0.52	93.59	56.7
4	7.77	4,137,254	(12,2)	3.85	58.97	6.2
6	12.60	1,538,035	(10,0)	9.87	22.15	2
8	13.01	644,462	(10,0)	12.33	11.55	0.6
10	9.25	1,903,519	(5,1)	8.47	13.95	0

Time Limit: 3600 seconds.

4

Optimizing Maritime Transshipment Operations for the Noble Group

The Noble Group is a global supply chain manager of agricultural and energy products and metals, minerals and ores. It is headquartered in Hong Kong, and operates in over 140 locations. This paper describes a modeling framework developed for Noble's coal transshipment operations, which include the transportation of coal from several mines to jetties, where it is loaded onto river barges, which then transport the coal to ports where it is transferred onto ocean vessels using floating cranes. Noble incur penalties for delays and late deliveries, costing millions of dollars each month. Additional infrastructure can be hired on a spot basis to minimize the impact of delays, but it comes at a high cost. Our model is designed to minimize the cost of these transshipment operations, including penalities and cost of spot-market resources. The complexity and scale of the model, however, puts it beyond the capabilities of state-of-the-art solvers. Therefore, we develop a column generation procedure that provides strong lower bounds, and a fast local search algorithm that delivers high quality solutions. The modeling framework has been fully implemented in 2013, and the latest results show a significant decrease in Noble's overall shipping costs.

4.1 Introduction

Maritime transport is the driving force behind international trade and a key driver of globalization. According to UNCTAD (2012), more than 80% of global trade by volume is carried by sea. The design of maritime supply chains that operate efficiently under such high volume of demand is the determining factor of success in the maritime logistics businesses. A vital part of any maritime supply chain is the transshipment of goods to downstream suppliers, a complex process that involves multiple suppliers, heterogeneous vessels, external resources with limited capacity and time availability, and multiple delivery locations. From an economic perspective, there are penalties associated with vessel delays and late deliveries, high transportation costs, and external resources provided under contracts with complex cost structures. Moreover, the maritime environment is vulnerable to uncertainties that carry high impact: weather conditions, infrastructure damage, bureaucracy and fluctuating fuel prices are operational risks that can jeopardize supply chain performance.

Despite the inherent complexity of the maritime environment and the vital role of logistics, most companies still resort to manual planning and human judgment to support their decision making processes. The use of optimization methods is limited, despite the tremendous potential of savings they could bring. In this paper we report on the development, evolution, implementation and realized benefits of a hybrid column generation and local search methodology designed for the Noble Group, a global supply chain manager of commodities. The group's operating platform emphasizes seaborne trade with origination in low-cost countries and delivery to high-growth markets. With major activities in energy, agriculture, metals, minerals and ores, and more than 90 billion USD revenue in 2012. Noble's energy segment accounts for 70% of total generated revenue, with coal being the most important traded commodity. Therefore, in this paper we focus on the coal logistics operations.

The coal logistics operations involve the scheduling of a heterogeneous fleet of river boats, whose mission is to carry the cargo from river jetties to large ocean vessels. The economic complexity of the problem is deeply operational, as the managers have to decide on whether they are going to use company-owned resources, or hire spot resources in order to expedite the vessel service and avoid potential latency penalties. The trade-off between the utilization of spot capacity versus the occurrence of penalties from

reduced service quality is a fundamental one in the operations management literature Levin et al. (2012). In a maritime transshipment setting, this trade-off is further complicated by an underlying scheduling and allocation component: the timing of the transshipment operations influences the optimal capacity allocation in a non-trivial way. To the best of our knowledge, this is the first paper that addresses the trade-off between spot capacity utilization and penalties for late deliveries in a maritime environment. Our methodology tackles the underlying scheduling problem using real-time data, taking into account the time availability of scarce resources. Initially, Noble's managers approached the authors seeking advice on when spot capacity should be preferred over the risk of penalties from late deliveries. However, we quickly realized that an operationally viable solution should also take the scheduling part explicitly into account. The output of this project was a modeling framework and decision support system that are currently used on a daily basis in two major ports in Indonesia, and are planned to be rolled out globally in the near future.

This paper focuses on the methodological approach adopted by the authors and its meaningful generalizations. It further reports on the implementation challenges, the realized benefits, and the lessons learned from this project. Last, we discuss potential implementations of the developed framework in logistics environments of similar nature. The academic contributions of the current work are (i) the development of a mathematical model and of a Dantzig-Wolfe Decomposition reformulation that describe the operational complexities of the integrated supply chain (ii) the development of a solution methodology that combines column generation with a local search algorithm (iii) computational experiments that demonstrate the fundamental trade-offs and the efficiency of the developed method and (iv) discussion on extensions to similar problems. Practical contributions include (i) the development of a practical decision making framework based on real-time data for a complex supply chain that has not been studied before and (ii) improved efficiency of operations, resulting in cost savings exceeding $1 million per month.

4.2 Literature Review

Recent years have seen a surge of research related to maritime transportation. Christiansen et al. (2007) give a comprehensive review of advancements in maritime trans-

portation modeling. There are many studies that have addressed problems similar to the one addressed in this paper, but to the best of our knowledge, all studies have significant differences. Perhaps the oldest relevant paper is that of O'Brien and Crane (1959), who investigate the scheduling of a barge line on the Ohio and Mississippi rivers. The authors use a simulation model to allocate tug boats to barges and approximate the optimal number of barges loads on an annual basis. Our study involves allocation, scheduling and spot capacity utilization decisions, and therefore can be considered as an extension of their study, ultimately providing insight into the optimal fleet size. Schwartz (1968) describes a transshipment scheduling model that minimizes barge fleet transportation costs. In order to simplify his formulations, he assumes infinite fleet capacity, identical cost for leased and owned resources and homogeneous resources. Interestingly, Schwartz cannot implement his model and notes "The size of the program generated by the foregoing model for reasonable size problems exceeds the capacity of present solution algorithms. Consequently, the model does not currently appear to be a practical means of solving daily operating problems of bargelines". Our modeling framework can be seen as a natural extension of his work, where we capture penalties for time deliveries, incorporate limited availability of resources and address the allocation of spot capacity. The interesting article by Jaikumar and Solomon (1987) links the tug boat minimization problem to vehicle routing and provides a one-pass algorithm that solves the problem in $\mathcal{O}(n)$ time, where n is the barge fleet size to which tug boats must be allocated. Furthermore, they discuss how their model generalizes to incorporate stochastic demand patterns.

From a practical perspective, the simulation studies of Richetta and Larson (1997) and Taylor et al. (2005) report successful implementations at New York City's refuse marine transport system and at the operations of the American Commercial Barge Line on the Mississippi river. In addition, a recent work by Wagner and Radovilsky (2012) addresses a barge allocation problem at the US Coastal Guard. Our work can be considered a dynamic extension of their approach, as we incorporate real-time data such as actual vessel arrivals and supplier and resource availabilities.

Maritime inventory routing is another related class of maritime problems, which has received a lot of attention the recent years (Al-Khayyal and Hwang 2007, Furman et al. 2011, Persson and Gothe-Lundgren 2005). Most articles apply column generation

and branch-and-price (Engineer et al. 2012) or valid inequalities and extended formulations (Agra et al. 2013) and achieve good performance in terms of integrality gaps on practical size instances. Our model poses an extra level of complexity that comes from the need to explicitly sequence the transshipment operations, and from limited infrastructure resource availability, which makes our problem similar to those found in process scheduling, (Floudas and Lin 2005).

To the best of our knowledge, the modeling framework introduced in this paper is the first after Schwartz (1968) that captures the timing of transshipment operations and, in addition, considers multiple contracting options for resource capacity. Moreover, the dynamic nature of the model makes it versatile enough to cope with unexpected events, such as adverse weather conditions. From a technical viewpoint, the model combines column generation with a local search of feasible solutions, an approach that delivers good quality integrality gaps in reasonable times. Finally, the implemented planning system is intuitive for the schedulers to use on a daily basis.

4.3 Description of Transshipment Operations

Noble's coal supply chain involves the transportation of coal from the mines to river jetties and then to ocean vessels, that ship the coal to downstream customers. Noble's logistics division is responsible for the transportation of coal from the time it reaches the river jetties until it is discharged onto the customers' vessels. First, the coal is loaded onto *barges* which then, after some essential paperwork, sail to the destination port, where they discharge their cargo onto large ocean vessels. As soon as a barge discharges its cargo, it returns to a hub, and then sails again to a jetty to serve either the same or another vessel. Figure 4.1 shows the various operations during a barge voyage. The sailing times between jetties, vessels and the hub can vary between half a day and 4 days.

Loading of Barges. A barge can load at a jetty whenever (i) the jetty is available and (ii) there is sufficient cargo. If either condition is not met, loading is deferred. Typically, suppliers notify Noble in advance about their cargo and jetty availability, and logistics managers send a barge to a jetty only if loading feasibility has been confirmed. Each supplier owns a single jetty, and the duration of loading depends on the quality of the loading infrastructure and on the size of each barge. The jetties are scattered across

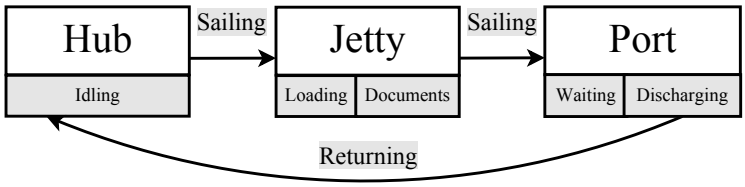


Figure 4.1: Transshipment operations and in-voyage barge states.

different locations, at different distances from the hub. As a result, managers need to take into account the sailing time when they decide when the barges should start sailing to the suppliers. It is important to note that the loading time at the jetties is usually higher than the time it takes to discharge the cargo onto a vessel. Therefore if two barges load cargo from the same supplier in consecutive time slots, they are not able to discharge it back-to-back onto the vessel, as by the time the first barge has discharged, the second barge is still sailing to the port. This can create costly delays, as explained later.

Loading of Ocean Vessels. Each vessel has an *estimated time of arrival* (ETA), which is agreed between Noble and the customer at least 2 weeks in advance of the actual vessel arrival. Also agreed is a *laytime*, a time window starting with the actual arrival of the vessel during which each vessel must be fully loaded. After the end of laytime, *demurrage*, a daily penalty, must be paid to the customer. If vessel loading is completed before the end of the laytime, a bonus based on the unused time is awarded, called *despatch*. However, large amounts of despatch are not achievable because the daily despatch rate is low, and the agreed laytime is typically quite close to the minimum required loading time of each vessel. Demurrage, however, can be very substantial, as high as $45,000 per day per vessel. Total delays of up to 5 days are not uncommon, resulting in demurrage exceeding $225,000 for just one vessel, and in annual multi-million dollar penalties for Noble's Indonesian operations.

The vessel loading operation depends on the vessel type. Certain vessels have geared grabs installed, which grab the cargo from the barge surface and discharge it into the storage hatch. The geared grabs can load cargo from either side of the vessel and therefore can discharge two barges simultaneously. For vessels that do not have geared grabs, their loading requires *floating cranes*. The loading speed of floating cranes is much higher compared to those of geared grabs, but only one floating crane

can be used per vessel, and it can discharge only one barge at a time. The allocation of floating cranes to vessels is made by a separate devision, out of direct control of Noble's supply chain management division. This is because the floating cranes are not only a supporting service, but a stand-alone business unit of the group. Figure 4.2 illustrates the terminology introduced so far, using an example with 3 barge voyages. In this example, demurrage is incurred because the third barge started its voyage too late, due to it being committed to another vessel beforehand.

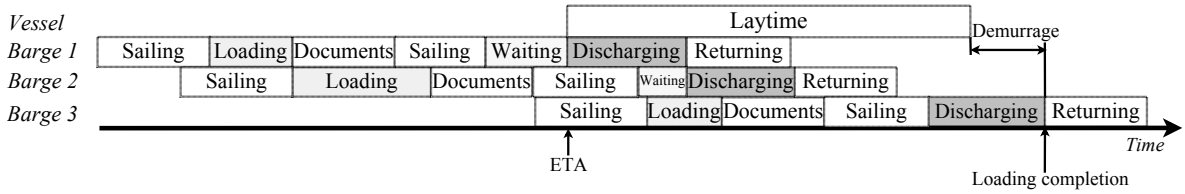


Figure 4.2: Barge Operations and Vessel Loading.

Barge Attributes. Barges can be fully characterized by their size and their ownership. Noble own a fleet of barges of various sizes, and additional barges are hired on an annual basis under a leasing contract. Noble can use the leased barges on demand, up to a maximum number per vessel, specified in a contract. The number of leased barges depends on the annual demand forecast and is decided at the beginning of each year on a rolling basis. There also exists a *spot market* for barges, consisting of companies that offer their barges on a one-off basis. The spot barges are expensive to hire, and are used only when an economic benefit is anticipated, i.e., when excessive penalties can be avoided.

Operational Costs and Trade-Offs. Every barge voyage has an associated transportation cost. For barges owned by Noble, the predominant cost component is the fuel cost, which depends on the size of the barge and on the distance it covers to reach the supplier jetty. The cost of voyages by leased and spot barges is a flat amount per carried tonne, which also depends on the supplier location and on the barge size. For any given size and location, Noble-owned barges are cheaper compared to leased barges, and spot barges are the most expensive. In addition, for leased and spot barges Noble pay *detention*, a daily fee for every extra day that the voyage duration exceeds an agreed number of days.

The need to utilize extra barges comes from the requirement to promptly load the customer vessels: late completion of loading carries heavy demurrage and undermines the group's reputation as a reliable commodities trader. The fundamental trade-off that the barge rotation model resolves is the joint minimization of demurrage and transportation costs: demurrage can be avoided if extra barges are utilized, but the utilization of extra barges carries a high cost. Therefore, the objective is to determine the *appropriate* number of barges such that the total cost from vessel delays and barge transportation is minimized. Section 4.3.1 follows with the formulation of the Barge Rotation model.

4.3.1 Model Formulation

Building blocks

The proposed model indicates for each Noble-owned, or *regular*, barge the sequence of voyages that this barge is going to make. Decision variables are noted with capital letters. We prepend I for binary variables, and append S for binary variables that express sequencing decisions. For example, X is a continuous variable, IX a binary variable, and $IXS_{ab} = 1$ if a precedes b and 0 otherwise. Parameters are denoted with small letters and sets with large calligraphic letters. Finally, we use ϵ and M to denote an arbitrarily small and large quantity, respectively.

The *Barge Rotation Model* uses the following notation:

Sets

Vessels:	$v \in \mathcal{V}$
Suppliers:	$s \in \mathcal{S}$
Regular (Owned) Barges:	$b \in \mathcal{B}$
Barge Types:	$\tau \in \mathcal{BT}$
Time Horizon:	$t \in \mathcal{T}$

Subsets and Indexed Sets

Suppliers that serve Vessel v:	$\mathcal{S}_v \subseteq \mathcal{S}$,	$\forall v \in \mathcal{V}$
Vessels served by Supplier s:	$\mathcal{V}_s \subseteq \mathcal{V}$,	$\forall s \in \mathcal{S}$
Barge Types Allowed at Supplier s:	$\mathcal{BT}_s \subseteq \mathcal{BT}$,	$\forall s \in \mathcal{S}$
Regular Barge Types:	$\mathcal{R} \subseteq \mathcal{BT}$,	
Regular Barge Types Allowed at Supplier s:	$\mathcal{R}_s \subseteq \mathcal{BT}_s$,	$\forall s \in \mathcal{S}$

Barge Types are defined as the cross product of barge sizes with contract types.

Parameters

Processing Documents Duration:	$tdocs \in \mathbb{R}^+$	
Return to Hub Duration:	$tret \in \mathbb{R}^+$	
Barge Type of Barge b:	$\tau_b \in \mathcal{R}$	$\forall b \in \mathcal{B}$
Sailing Duration:	$tsail_s \in \mathbb{R}^+$,	$\forall s \in \mathcal{S}$
Loading Duration:	$tload_{\tau s} \in \mathbb{R}^+$,	$\forall \tau \in \mathcal{BT}_s, s \in \mathcal{S}$
Discharging Duration:	$tdisch_{\tau v} \in \mathbb{R}^+$,	$\tau \in \mathcal{BT}, \forall v \in \mathcal{V}$
Tonnage Cost:	$tonc_{\tau s} \in \mathbb{R}^+$,	$\forall \tau \in \mathcal{BT}_s, s \in \mathcal{S}$
Detention Cost:	$detc_{\tau s} \in \mathbb{R}^+$,	$\forall \tau \in \mathcal{BT}_s, s \in \mathcal{S}$
Detention Time Window:	$tdet_{\tau s} \in \mathbb{R}^+$,	$\forall \tau \in \mathcal{BT}_s, s \in \mathcal{S}$
Estimated Time of Arrival:	$eta_v \in \mathbb{R}^+$,	$\forall v \in \mathcal{V}$
Laytime:	$lt_v \in \mathbb{R}^+$,	$\forall v \in \mathcal{V}$
Demurrage Rate:	$rdem_v \in \mathbb{R}^+$,	$\forall v \in \mathcal{V}$
Despatch Rate:	$rdes_v \in \mathbb{R}^+$,	$\forall v \in \mathcal{V}$
Floating crane for vessel v is blocked at time t:	$fb_{vt} \in \{0,1\}$,	$\forall v \in \mathcal{V}, t \in \mathcal{T}$
Jetty of supplier s is blocked at time t:	$jb_{st} \in \{0,1\}$,	$\forall s \in \mathcal{S}, t \in \mathcal{T}$
Vessel has Floating Crane:	$f_v \in \{0,1\}$,	$\forall v \in \mathcal{V}$
Max Number of Barges:	$nb_{\tau v} \in \mathbb{Z}^+$,	$\forall \tau \in \mathcal{BT}, v \in \mathcal{V}$
Quantity Loaded:	$q_{sv} \in \mathbb{R}^+$,	$\forall s \in \mathcal{S}_v, v \in \mathcal{V}$
Barge Type Capacity:	$cap_\tau \in \mathbb{R}^+$,	$\forall \tau \in \mathcal{BT}$

We define the set of voyages associated with each supplier $s \in \mathcal{S}_v$ and vessel $v \in \mathcal{V}$:

$$\mathcal{W}_{sv} := \left\{1, \ldots, \lceil \frac{q_{sv}}{\min_\tau cap_\tau} \rceil \right\}, \forall s \in \mathcal{S}_v, v \in \mathcal{V},$$

where $\lceil \frac{q_{sv}}{\min_\tau cap_\tau} \rceil$ indicates the maximum number of barges needed to carry q_{sv} tonnes to vessel v. Note that the actual number of voyages depends on the size of the allocated barges, and can be less than the maximum. In particular, if barges larger than the minimum size are allocated, fewer voyages can be needed. We call each chosen voyage *active*, and assign a binary indicator showing when a voyage is active, as explained below.

Decision Variables

Amount of vessel demurrage:	Dm_v,
Indicator of demurrage:	IDm_v,
Amount of vessel despatch:	Dp_v,
Completion time of the vessel loading operation:	Com_v,
Barges of type τ allocated to supplier s for vessel v:	$B_{\tau sv}$,
Indicator activated when voyage w serves supplier s using a type τ barge for vessel v:	$IV_{ws\tau v}$,
Indicator activated when voyage w to supplier s of vessel v is allocated to regular barge b:	IB_{wsbv},
Amount of detention in days for voyage w:	De_{wsv},
Start of loading for each voyage:	L_{wsv},
Loading indicator activated when loading starts on t:	IL_{wstv},
Loading sequence indicator between w_1 and w_2:	$ILS^{w_1v_1}_{w_2v_2s}$,
Start of Discharge for each voyage:	Di_{wsv},
Discharge indicator activated when discharge starts on t:	IDi_{wstv},
Discharge sequence indicator between w_1 and w_2:	$IDiS^{w_1s_1}_{w_2s_2v}$,
Indicator activated when w_1 precedes w_2:	$IVS^{w_1s_1v_1}_{w_2s_2v_2}$,

The Barge Rotation Model

The barge rotation model can be formulated as follows.

$$\min \quad \sum_{v\in\mathcal{V}}(Dm_v - Dp_v) + + \sum_{v\in\mathcal{V}}\sum_{s\in\mathcal{S}_v}\sum_{\tau\in\mathcal{BT}_s}\sum_{w\in\mathcal{W}_{sv}} detc_{\tau s}De_{ws\tau v} \tag{A.1}$$

$$\text{s.t.} \quad Dm_v \le M * IDm_v, \quad \forall v \in \mathcal{V} \tag{A.2}$$

$$Dm_v \le (Com_v - eta_v - lt_v)rdem_v + M(1 - IDm_v), \quad \forall v \in \mathcal{V} \tag{A.3}$$

$$Dm_v \ge (Com_v - eta_v - lt_v)rdem_v - M(1 - IDm_v), \quad \forall v \in \mathcal{V} \tag{A.4}$$

$$Dp_v \le M(1 - IDm_v), \quad \forall v \in \mathcal{V} \tag{A.5}$$

$$Dp_v \le (eta_v + lt_v - Com_v)rdes_v + M * IDm_v, \quad \forall v \in \mathcal{V} \tag{A.6}$$

$$Dp_v \ge (eta_v + lt_v - Com_v)rdes_v - M * IDm_v, \quad \forall v \in \mathcal{V} \tag{A.7}$$

$$Com_v - eta_v - lt_v \le M * IDm_v, \quad \forall v \in \mathcal{V} \tag{A.8}$$

$$Com_v - eta_v - lt_v \ge M(IDm_v - 1), \quad \forall v \in \mathcal{V} \tag{A.9}$$

$$\sum_{s\in\mathcal{S}_v:\tau\in\mathcal{BT}_s} B_{\tau sv} \le nb_{\tau v}, \quad \forall \tau \in \mathcal{BT}, v \in \mathcal{V} \tag{A.10}$$

$$\sum_{\tau\in\mathcal{BT}_s} cap_\tau B_{\tau sv} \ge q_{sv}, \quad \forall s \in \mathcal{S}_v, v \in \mathcal{V} \tag{A.11}$$

$$\sum_{w \in \mathcal{W}_{sv}} IV_{ws\tau v} = B_{\tau sv}, \quad \forall \tau \in \mathcal{BT}_s, s \in \mathcal{S}_v, v \in \mathcal{V} \tag{A.12}$$

$$\sum_{\tau \in \mathcal{BT}_s} IV_{ws\tau v} \leq 1, \quad \forall w \in \mathcal{W}_{sv}, s \in \mathcal{S}_v, v \in \mathcal{V} \tag{A.13}$$

$$\sum_{b:\tau_b = \tau} IB_{wsbv} = IV_{ws\tau v}, \quad \forall w \in \mathcal{W}_{sv}, \tau \in \mathcal{R}_s, s \in \mathcal{S}_v, v \in \mathcal{V} \tag{A.14}$$

$$\sum_{w \in \mathcal{W}_{sv}} \sum_{t \in \mathcal{T}} IL_{wstv} = \sum_{\tau \in \mathcal{BT}} B_{\tau sv}, \quad \forall s \in \mathcal{S}_v, v \in \mathcal{V} \tag{A.15}$$

$$tIL_{wstv} \leq L_{wsv} \leq (t+1-\epsilon)IL_{wstv} + M(1 - IL_{wstv}),$$
$$\forall t \in \mathcal{T}, w \in \mathcal{W}_{sv}, s \in \mathcal{S}_v, v \in \mathcal{V} \tag{A.16}$$

$$IL_{wst-uv} \leq 2 - jb_{st} - IV_{ws\tau v},$$
$$\forall w \in \mathcal{W}_{sv}, \tau \in \mathcal{BT}_s, s \in \mathcal{S}_v, v \in \mathcal{V}, t \in \mathcal{T},$$
$$u \in \{0, \ldots, \min(t-1, \lceil tload_{\tau s} \rceil - 1)\} \tag{A.17}$$

$$L_{wsv} \leq t - tload_{\tau s} IL_{wst-\lceil tload_{\tau s} \rceil v} + M * (2 - IL_{wst-\lceil tload_{\tau s} \rceil v} - IV_{ws\tau v}),$$
$$\forall w \in \mathcal{W}_{sv}, \tau \in \mathcal{BT}_s, s \in \mathcal{S}_v, v \in \mathcal{V}, t \in \mathcal{T} : jb_{st} = 1 \tag{A.18}$$

$$L_{w_1 s v_1} + \sum_{\tau \in \mathcal{BT}_s} tload_{\tau s} IV_{w_1 s \tau v_1} \leq L_{w_2 s v_2} + M(1 - ILS^{w_1 v_1}_{w_2 v_2 s}),$$
$$\forall w_i \in \mathcal{W}_{sv_i}, v_i \in \mathcal{V}_s, i = 1,2 : v_1 \neq v_2, s \in \mathcal{S} \tag{A.19}$$

$$L_{w_1 s v} + \sum_{\tau \in \mathcal{BT}_s} tload_{\tau s} IV_{w_1 s \tau v} \leq L_{w_2 s v} + M(1 - ILS^{w_1 v}_{w_2 v s}),$$
$$\forall w_1 \neq w_2 \in \mathcal{W}_{sv}, s \in \mathcal{S}_v, v \in \mathcal{V} \tag{A.20}$$

$$ILS^{w_1 v_1}_{w_2 v_2 s} + ILS^{w_2 v_2}_{w_1 v_1 s} = 1, \forall w_i \in \mathcal{W}_{sv_i}, v_i \in \mathcal{V}_s, i = 1,2 : v_1 \neq v_2, s \in \mathcal{S} \tag{A.21}$$

$$ILS^{w_1 v}_{w_2 v s} + ILS^{w_2 v}_{w_1 v s} = 1, \quad \forall w_1, w_2 \in \mathcal{W}_{sv} : w_1 \neq w_2, s \in \mathcal{S}_v, v \in \mathcal{V} \tag{A.22}$$

$$1 - (ILS_{w_1 v_1 w_v s} + ILS_{w_v w_1 v_1 s} + ILS_{w_2 v_2 w_v s} + ILS_{w_v w_2 v_2 s}) \leq$$
$$ILS_{w_1 v_1 w_2 v_2 s} + ILS_{w_2 v_2 w_1 v_1 s}, \forall w, w_i \in \mathcal{W}_{s_i v_i}, s, s_i \in \mathcal{S}_{v_i}, v, v_i \in \mathcal{V}, i = 1,2 \tag{A.23}$$

$$\sum_{w \in \mathcal{W}_{sv}} \sum_{t \in \mathcal{T}} IDi_{wstv} = \sum_{\tau \in \mathcal{BT}_s} B_{\tau sv}, \quad \forall s \in \mathcal{S}_v, v \in \mathcal{V} \tag{A.24}$$

$$tIDi_{wstv} \leq Di_{wsv} \leq (t+1-\epsilon)IDi_{wstv} + M(1 - IDi_{wstv}),$$
$$\forall t \in \mathcal{T}, w \in \mathcal{W}_{sv}, s \in \mathcal{S}_v, v \in \mathcal{V} \tag{A.25}$$

$$Di_{w_1 s_1 v} + \sum_{\tau \in \mathcal{BT}_{s_1}} tdisch_{s_1 v} IV_{w_1 s_1 \tau v} \leq Di_{w_2 s_2 v} + M(1 - IDiS^{w_1 s_1}_{w_2 s_2 v}),$$
$$\forall w_i \in \mathcal{W}_{s_i v}, s_i \in \mathcal{S}_v, i = 1,2 : s_1 \neq s_2, v \in \mathcal{V} \tag{A.26}$$

$$Di_{w_1sv} + \sum_{\tau \in \mathcal{BT}_s} tdisch_{sv} IV_{w_1s\tau v} \leq Di_{w_2sv} + M(1 - IDiS^{w_1s}_{w_2sv}),$$
$$\forall w_i \in \mathcal{W}_{sv}, s \in \mathcal{S}_v, v \in \mathcal{V} \quad (A.27)$$

$$IDiS_{w_1s_1w_2s_2v} + IDiS^{w_2s_2}_{w_1s_1v} \leq 1, \geq f_v,$$
$$\forall w_i \in \mathcal{W}_{s_iv}, s_i \in \mathcal{S}_v, i = 1,2 : s_1 \neq s_2, v \in \mathcal{V} \quad (A.28)$$

$$IDiS_{w_1sw_2sv} + IDiS^{w_2s}_{w_1sv} \leq 1, \geq f_v, \ \forall w_1 \neq w_2 \in \mathcal{W}_{sv}, s \in \mathcal{S}_v, v \in \mathcal{V} \quad (A.29)$$

$$1 \leq \sum_{j=1}^{3}\sum_{k=1}^{3} IDiS^{w_js_j}_{w_ks_kv},$$
$$\forall w_i \in \mathcal{W}_{s_iv}, s_i \in \mathcal{S}_v, i = 1,2,3 : s_1 < s_2 < s_3, v \in \mathcal{V} \quad (A.30)$$

$$1 \leq \sum_{j=1}^{3}\sum_{k=1}^{3} IDiS^{w_js_j}_{w_ks_kv},$$
$$\forall w_i \in \mathcal{W}_{s_iv}, s_i \in \mathcal{S}_v, i = 1,2,3 : w_1 \neq w_2, s_1 = s_2 < s_3, v \in \mathcal{V} \quad (A.31)$$

$$1 \leq \sum_{j=1}^{3}\sum_{k=1}^{3} IDiS^{w_js_j}_{w_ks_kv},$$
$$\forall w_i \in \mathcal{W}_{s_iv}, s_i \in \mathcal{S}_v, i = 1,2,3 : w_1 \neq w_3, s_1 = s_3 < s_2, v \in \mathcal{V} \quad (A.32)$$

$$1 \leq \sum_{j=1}^{3}\sum_{k=1}^{3} IDiS^{w_js_j}_{w_ks_kv},$$
$$\forall w_i \in \mathcal{W}_{s_iv}, s_i \in \mathcal{S}_v, i = 1,2,3 : w_2 \neq w_3, s_1 < s_2 = s_3, v \in \mathcal{V} \quad (A.33)$$

$$1 \leq \sum_{j=1}^{3}\sum_{k=1}^{3} IDiS^{w_js}_{w_ksv}, \ \forall w_1 \neq w_2 \neq w_3 \in \mathcal{W}_{s_iv}, s \in \mathcal{S}_v, v \in \mathcal{V} \quad (A.34)$$

$$Di_{wsv} + \sum_{\tau \in \mathcal{BT}_s} tdisch_{\tau v} IV_{ws\tau v} \leq Com_v, \ \forall w \in \mathcal{W}_{sv}, s \in \mathcal{S}_v, v \in \mathcal{V} \quad (A.35)$$

$$Di_{wsv} \geq L_{wsv} + \sum_{\tau \in \mathcal{BT}_s} tload_{\tau s} IV_{ws\tau v} + tdocs + tsail_s,$$
$$\forall w \in \mathcal{W}_{sv}, s \in \mathcal{S}_v, v \in \mathcal{V} \quad (A.36)$$

$$eta_v * \sum_{\tau \in \mathcal{BT}_s} IV_{ws\tau v} \leq Di_{wsv}, \ \forall w \in \mathcal{W}_{sv}, s \in \mathcal{S}_v, v \in \mathcal{V} \quad (A.37)$$

$$IDi_{wst-uv} \leq 2 - IV_{ws\tau v} - fb_{vt}, \ \forall w \in \mathcal{W}_{sv}, \tau \in \mathcal{BT}_s, s \in \mathcal{S}_v, v \in \mathcal{V},$$
$$t \in \mathcal{T}, u \in \{0, \ldots, \min(t-1, \lceil tdisch_{\tau v} \rceil - 1\}) \quad (A.38)$$

$$Di_{wsv} \leq t - tdisch_{\tau v} IDi_{wst-\lceil tdisch_{\tau v} \rceil v} + M(2 - IDi_{wst-\lceil tdisch_{\tau v} \rceil v} - IV_{ws\tau v}),$$
$$\forall w \in \mathcal{W}_{sv}, \tau \in \mathcal{BT}_s, s \in \mathcal{S}_v, v \in \mathcal{V}, t \in \mathcal{T} : fb_{vt} = 1 \quad (A.39)$$

$$Di_{wsv} + (tdisch_{\tau v} - tdet_{\tau s}) IV_{ws\tau v} - L_{wsv} - M(1 - IV_{ws\tau v})$$

$$\leq De_{wsv} \leq M \sum_{\tau \in \mathcal{BT}_s} IV_{ws\tau v}, \ \forall w \in \mathcal{W}_{sv}, \tau \in \mathcal{BT}_s, s \in \mathcal{S}_v, v \in \mathcal{V} \quad \text{(A.40)}$$

$$Di_{w_1 s_1 v_1} + \sum_{\tau \in \mathcal{BT}_{s_1}} tdisch_{\tau v_1} IV_{w_1 s_1 \tau v_1} + tret - M(1 - IVS^{w_1 s_1 v_1}_{w_2 s_2 v_2}) \leq L_{w_2 s_2 v_2} - tsail_{s_2},$$

$$\forall w_i \in \mathcal{W}_{s_i v_i}, s_i \in \mathcal{S}_{v_i}, v_i \in \mathcal{V}, i = 1, 2 : v_1 \neq v_2 \quad \text{(A.41)}$$

$$Di_{w_1 s_1 v} + \sum_{\tau \in \mathcal{BT}_{s_1}} tdisch_{\tau v} IV_{w_1 s_1 \tau v} + tret - M(1 - IVS^{w_1 s_1 v}_{w_2 s_2 v}) \leq L_{w_2 s_2 v} - tsail_{s_2},$$

$$\forall w_i \in \mathcal{W}_{s_i v}, s_i \in \mathcal{S}_v, i = 1, 2 : s_1 \neq s_2, v \in \mathcal{V} \quad \text{(A.42)}$$

$$Di_{w_1 s v} + \sum_{\tau \in \mathcal{BT}_s} tdisch_{\tau v} IV_{w_1 s \tau v} + tret - M(1 - IVS^{w_1 s v}_{w_2 s v}) \leq L_{w_2 s v} - tsail_{s_2},$$

$$\forall w_1 \neq w_2 \neq w_3, \in \mathcal{W}_{sv}, s \in \mathcal{S}_v, v \in \mathcal{V} \quad \text{(A.43)}$$

$$IVS^{w_1 s_1 v_1}_{w_2 s_2 v_2} + IVS^{w_2 s_2 v_2}_{w_1 s_1 v_1} \leq 1, \geq IB_{w_1 s_1 b v_1} + IB_{w_2 s_2 b v_2} - 1,$$

$$\forall w_i \in \mathcal{W}_{s_i v_i}, s_i \in \mathcal{S}_{v_i}, v_i \in \mathcal{V}, i = 1, 2 : v_1 < v_2, b \in \mathcal{B} \quad \text{(A.44)}$$

$$IVS^{w_1 s_1 v}_{w_2 s_2 v} + IVS^{w_2 s_2 v}_{w_1 s_1 v} \leq 1, \geq IB_{w_1 s_1 b v} + IB_{w_2 s_2 b v} - 1,$$

$$\forall w_i \in \mathcal{W}_{s_i v}, s_i \in \mathcal{S}_v i = 1, 2 : s_1 < s_2, v \in \mathcal{V}, b \in \mathcal{B} \quad \text{(A.45)}$$

$$IVS^{w_1 s v}_{w_2 s v} + IVS^{w_2 s v}_{w_1 s v} \leq 1, \geq IB_{w_1 s b v} + IB_{w_2 s b v} - 1,$$

$$\forall w_1 < w_2 \in \mathcal{W}_{sv}, s \in \mathcal{S}_v, v \in \mathcal{V}, b \in \mathcal{B} \quad \text{(A.46)}$$

$$Dm_v \geq 0, \ IDm_v \in \{0, 1\}, \ Dp_v \geq 0, \ Com_v \geq 0, \ \forall v \in \mathcal{V} \quad \text{(A.47)}$$

$$B_{\tau s v} \in \mathbb{N}, \quad \forall \tau \in \mathcal{BT}_s, s \in \mathcal{S}_v, v \in \mathcal{V} \quad \text{(A.48)}$$

$$IV_{ws\tau v} \in \{0, 1\}, \quad \forall w \in \mathcal{W}_{sv}, \tau \in \mathcal{BT}_s, s \in \mathcal{S}_v, v \in \mathcal{V} \quad \text{(A.49)}$$

$$IB_{wsbv} \in \{0, 1\}, \quad \forall w \in \mathcal{W}_{sv}, s \in \mathcal{S}_v, v \in \mathcal{V}, b \in \mathcal{B} \quad \text{(A.50)}$$

$$L_{wsv} \geq 0, \ Di_{wsv} \geq 0, \ De_{wsv} \geq 0, \quad \forall w \in \mathcal{W}_{sv}, s \in \mathcal{S}_v, v \in \mathcal{V} \quad \text{(A.51)}$$

$$L_{wsvt} \in \{0, 1\}, \ Di_{wsvt} \in \{0, 1\}, \quad \forall w \in \mathcal{W}_{sv}, s \in \mathcal{S}_v, v \in \mathcal{V}, t \in \mathcal{T} \quad \text{(A.52)}$$

$$ILS^{w_1 v_1}_{w_2 v_2 s} \in \{0, 1\}, \quad \forall w_i \in \mathcal{W}_{sv_i}, v_i \in \mathcal{V}_s, i = 1, 2, s \in \mathcal{S} \quad \text{(A.53)}$$

$$IDiS^{w_1 s_1}_{w_2 s_2 v} \in \{0, 1\}, \quad \forall w_i \in \mathcal{W}_{s_i v}, s_i \in \mathcal{S}_v, i = 1, 2, v \in \mathcal{V} \quad \text{(A.54)}$$

$$IVS^{w_1 s_1 v_1}_{w_2 s_2 v_2} \in \{0, 1\}, \quad \forall w_i \in \mathcal{W}_{s_i v_i}, s_i \in \mathcal{S}_{v_i}, v_{\in} \mathcal{V}, i = 1, 2 \quad \text{(A.55)}$$

The objective function takes into consideration three cost components, namely the joint demurrage cost or despatch bonus for all vessels, the total transportation cost, which depends on the barge type, i.e., its size and its contract structure, and the penalty detention that occurs if the voyage transshipment operations exceed a predefined time window. Constraints (A.2)–(A.9) model the penalties incurred when the completion of

loading exceeds the ETA by more than the laytime, as well as the bonuses received for early completions.

Constraints (A.10)–(A.14) model the allocation of barge types to vessels and voyages. Specifically, (A.10) pose an upper limit on the maximum number of barges for each barge type and vessel. For owned barges, this upper limit is simply the number of owned barges of each size. For other barges, it is specified by the corresponding contract. Constraints (A.11) pose that the barges allocated to each supplier should carry the agreed quantity for each vessel, and (A.12) indicate the total number of voyages taken from each barge type to the suppliers of each vessel. Further, (A.13) express that at most one barge type should be used for any active voyage. Also, (A.14) poses that if a voyage is served by a regular barge type, then there must be exactly one regular barge of that type that serves it. Note that since these (A.14) are restricted to regular barges, different barge types denote a difference in size only.

Constraints (A.15)–(A.22) describe the in-voyage barge loading operations. Constraints (A.15) interface the loading and allocation parts, expressing that all allocated barges should have a loading operation. Constraints (A.16) link the continuous timing of loading with the time-indexed indicator. Note that the binary indicator variable of loading would not have been necessary if all suppliers were fully available. In practice, key suppliers may not have available cargo on time, therefore we have to incorporate their availability explicitly. To this end, (A.17) and (A.18) model supplier availability: (A.17) prevents the start of loading at days where there is not enough time left for it to be completed, and (A.18) indicates the maximum time that loading can start within a day, such that the total loading operation remains within the permitted range. Finally, (A.19)–(A.22) impose that the loading operations of two voyages cannot overlap. Note that this set of constraints involves voyages of different vessels, as long as they are allocated to the same supplier. (A.19) impose the sequencing restrictions to voyages of different vessels, and (A.20) to voyages of the same vessel. Finally, (A.21) and (A.22) impose a sequence to all pairs of voyages.

Constraints (A.24)–(A.27) describe restrictions similar to those of the loading operations and, for the sake of brevity, their description is omitted. The case of vessels with geared grabs has an interesting peculiarity, as two barges can discharge simultaneously. (A.28) and (A.29) allow pairs of barges to discharge simultaneously for vessels with floating cranes, and (A.30)–(A.34) impose that if a discharge operation is allowed to

overlap with two others, then those two others are not allowed to overlap each other. (A.35)–(A.37) link the timing of discharge with the vessel loading time, the loading time and the ETA respectively. Finally, (A.40) define the amount of voyage detention in days for each active voyage.

The last part of the model, (A.41)–(A.46), imposes sequencing restrictions across pairs of voyages that use the same owned barge. The restriction is that any owned barge can be allocated to non-overlapping voyages only. Further, safety time can be added if necessary between the end of a voyage and the beginning of the next one. Finally (A.47)–(A.55) denote the non-negativity and integrality restrictions of the model's decisions.

4.3.2 Problem Complexity

Theorem 4.1 *The barge rotation model is strongly $\mathcal{NP}$-Hard (Atallah and Blanton 2010), even in the special case of a single vessel with no voyage costs and two suppliers.*

Proof We define a reduction from the Hybrid Flow Shop problem (HFS). A recent review on HFS is given by Ruiz and Vazquez-Rodriguez (2010), and a proof of its complexity can be found in Garey and Johnson (1990). In HFS we are given a set of n jobs that are to be processed in a series of m stages. Each stage k has $M_k \geq 1$ identical machines in parallel, which can be used to process the job. There must be at least one stage with at least two machines, therefore $\max_k M_k > 1$. The processing time of job j in stage k is given as p_{jk}. The objective is to create a feasible schedule that minimizes the maximum completion time of jobs across all machines. The decision variables indicate in which machine each job is processed for each stage and how the jobs are sequenced in each stage. Specifically, $Y_{jkl} = 1$, if job j on stage k is scheduled in machine l, 0 otherwise; $X_{jrk} = 1$, if job j precedes job $r \neq j$ on stage k, 0 otherwise.

Given an instance of the HFS problem, we create the following instance of a single-vessel barge rotation problem. The vessel is assumed to be served by a floating crane. Jobs of the HFS problem correspond to barge types of the barge rotation problem. Further, stages correspond to the loading and discharging operations and machines in each stage correspond to the suppliers and to the floating crane, respectively. Let us consider a family of HFS instances with input $\mathfrak{I}_F := \{n \in \mathbb{N}, m = 2, M_1 = 2, M_2 = 1, p_{jk} > 0\}$. We show that any feasible solution of an instance of this family is mapped

to a feasible solution of a properly constructed instance of our problem. For notational brevity, we drop the indexes of single cardinality sets from the variables. Given $\mathcal{I} \in \mathcal{I}_F$, a barge rotation instance can be constructed with $\mathcal{I}^* = \{|\mathcal{S}| = 2, \mathcal{B} = \emptyset; |\mathcal{BT}| = n, \mathcal{R} = \emptyset, tload_{\tau_j,1} = p_{j1}, tload_{\tau_j,2} = p_{j1}, tdisch_{\tau_j} = p_{j2}, rdem = 1, nb_{\tau_j} = 1, lt = 0, q_j = \lfloor n/2 \rfloor (cap_{\tau_j})\}$. We also assume zero sailing time, zero documents processing time, zero transportation costs, and complete time availability of suppliers and of the floating crane. Other data, such as the vessel ETA and the length of the detention time window, can take any value without affecting the feasibility and optimality of this instance.

We consider any feasible solution (Y^F_{jkl}, X^F_{jrk}) of $\mathcal{I}$. A feasible solution for the barge rotation instance $\mathcal{I}^*$ can be constructed as follows. First, we observe that the set of voyages to suppliers 1 and 2 can be expressed as $\mathcal{W}_1 = \{1, \ldots, \sum_j Y^F_{j11}\}; \mathcal{W}_2 = \{1, \ldots, \sum_j Y^F_{j12}\}$. Also, each job assignment to a machine determines the type of barge that is allocated to each voyage. This is because jobs have different processing times in each stage, and barge types have different sizes, and thereby different durations for loading and for discharging. Thus, the above assignment maps the allocation of jobs to machines on stage 1 to an allocation of barge types to suppliers:$Y^F_{jkl} = 1 \longrightarrow$ the j-th voyage receives the l-th barge type. Hence, this mapping indicates the values of $IV_{ws\tau}$. Under the same reasoning, X_{jrk} indicate the sequence between voyages w_j and w_r, for the loading and discharging stage. As a result, the exact sequences of loading and discharging can be recovered, and the exact timing of operations is then straightforward: each barge type starts an operation as soon as all the preceding barge types have completed that operation. L_{ws} and Di_{ws} are constructed under this argument. From L_{ws} and Di_{ws} the completion time can be derived, which equals the completion time objective of (Y^F_{jkl}, X^F_{jrk}), because we assumed unit cost of demurrage. The same argument can be extended to optimal solutions.■

4.4 Dantzig-Wolfe Decomposition

4.4.1 Motivation

Preliminary experiments with commercial branch-and-cut solvers showed that problems of realistic size are intractable. In smaller instances that could be solved to optimality, it was observed that the root node lower bound is very weak. For a case with 2 vessels and 3 barge voyages per vessel, the root node gap is 98.5% after the application of solver cuts

and heuristics, and 55% when the optimal solution was found. Such a weak lower bound makes the problem notoriously hard to solve when it comes to instances of larger size. In addition, a strong lower bound is useful from a practical viewpoint, as it provides an indication of the quality of the proposed solution. Given that much of the problem complexity lies at the in-vessel operations, a vessel decomposition approach could give promising results. This is because the polytopes associated with each vessel do not have the integrality property, and therefore an improved lower bound can be obtained (Barnhart et al. 1998, Fisher 2004). This section presents in detail the application of Dantzig-Wolfe decomposition on the original model.

Dantzig and Wolfe (1960) were the first to observe that systems which exhibit a rich structure in several separate modules are amenable to decomposition. In our context, the cross-vessel constraints (A.19), (A.21), (A.41) and (A.44) are used to coordinate the in-vessel restrictions, which generate feasible *service plans* for each vessel. Specifically, given a vessel $v \in \mathcal{V}$, we define $\mathcal{E}_v$ as the convex hull of the set of extreme points of the constraints associated with v only. Note that this set is bounded, as upper bounds for all continuous variables can be inferred from the size of the time horizon. Each extreme point p of $\mathcal{E}_v$ is associated with a vessel service plan and is defined by the values that the decision variables have at this particular plan. We introduce z_{pv} as the fraction of service plan p of vessel v that is used in an optimal solution. The variables of each subproblem appear in the formulation of the master as column coefficients of each service plan. To differentiate them from the master problem variables, we add a bar to them, and append a superscript p that indicates to which service plan they refer to. Further, we denote the cost of service plan p of vessel v as c_{pv}. The master problem reformulation is then as follows.

$$\min \sum_{v \in \mathcal{V}} \sum_{p \in \mathcal{E}_v} c_{pv} z_{pv} \tag{A.56}$$

$$\text{s.t.} \sum_{p \in \mathcal{E}_{v_1}} \bar{L}^p_{w_1 s v_1} z_{pv_1} + \sum_{\tau \in \mathcal{BT}_s} \sum_{p \in \mathcal{E}_{v_1}} tload_{\tau s} \bar{IV}^p_{w_1 s \tau v_1} z_{pv_1} \leq \sum_{p \in \mathcal{E}_{v_2}} \bar{L}^p_{w_2 s v_2} z_{pv_2} + M(1 - ILS^{w_1 v_1}_{w_2 v_2 s}),$$
$$\forall w_i \in \mathcal{W}_{sv_i}, v_i \in \mathcal{V}_s, i = 1,2 : v_1 \neq v_2, s \in \mathcal{S} \tag{A.57}$$

$$ILS^{w_2 s_2 v_2}_{w_1 s_1 v_1} + ILS^{w_1 s_1 v_1}_{w_2 s_2 v_2} = 1, \quad \forall w_i \in \mathcal{W}_{sv_i}, v_i \in \mathcal{V}_s, i = 1,2 : v_1 < v_2, s \in \mathcal{S} \tag{A.58}$$

$$\sum_{p\in\mathcal{E}_{v_1}} \bar{Di}^p_{w_1s_1v_1} z_{pv_1} + \sum_{\tau\in\mathcal{BT}_{s_1}} \sum_{p\in\mathcal{E}_{v_1}} tdisch_{\tau v_1} \bar{IV}^p_{w_1s_1\tau v_1} z_{pv_1} + tret - M(1 - IVS^{w_2s_2v_2}_{w_1s_1v_1})$$
$$\leq \sum_{p\in\mathcal{E}_{v_2}} \bar{L}^p_{w_2s_2v_2} z_{pv_2}, \forall w_i \in \mathcal{W}_{s_iv_i}, s_i \in \mathcal{S}_{v_i}, v_i \in \mathcal{V}_{s_i}, i = 1,2 : v_1 \neq v_2 \quad (A.59)$$

$$IVS^{w_2s_2v_2}_{w_1s_1v_1} + IVS^{w_1s_1v_1}_{w_2s_2v_2} \leq 1,$$
$$\forall w_i \in \mathcal{W}_{s_iv_i}, s_i \in \mathcal{S}_{v_i}, v_i \in \mathcal{V}_{s_i}, i = 1,2 : v_1 \leq v_2 \quad (A.60)$$

$$IVS^{w_2s_2v_2}_{w_1s_1v_1} + IVS^{w_1s_1v_1}_{w_2s_2v_2} \geq \sum_{p\in\mathcal{E}_{v_1}} IB^p_{w_1s_1bv_1} z_{pv_1} + \sum_{p\in\mathcal{E}_{v_2}} IB^p_{w_2s_2bv_2} z_{pv_2} - 1, \quad (A.61)$$
$$\forall w_i \in \mathcal{W}_{s_iv_i}, s_i \in \mathcal{S}_{v_i}, v_i \in \mathcal{V}_{s_i}, i = 1,2 : v_1 \neq v_2, b \in \mathcal{B} \quad (A.62)$$

$$\sum_{p\in\mathcal{E}_v} z_{pv} = 1, \quad \forall v \in \mathcal{V} \quad (A.63)$$

$$IDm_v = \sum_{p\in\mathcal{E}_v} I\bar{D}m^p_v z_{pv} \in \{0,1\}, \quad \forall v \in \mathcal{V} \quad (A.64)$$

$$IV_{ws\tau v} = \sum_{p\in\mathcal{E}_v} \bar{IV}^p_{ws\tau v} z_{pv} \in \{0,1\}, \quad \forall w \in \mathcal{W}_{sv}, \tau \in \mathcal{BT}_s, s \in \mathcal{S}_v, v \in \mathcal{V} \quad (A.65)$$

$$IL_{wstv} = \sum_{p\in\mathcal{E}_v} \bar{IL}^p_{wstv} z_{pv} \in \{0,1\}, \quad \forall w \in \mathcal{W}_{sv}, s \in \mathcal{S}_v, v \in \mathcal{V}, t \in \mathcal{T} \quad (A.66)$$

$$ID_{wstv} = \sum_{p\in\mathcal{E}_v} \bar{ID}^p_{wstv} z_{pv} \in \{0,1\}, \quad \forall w \in \mathcal{W}_{sv}, s \in \mathcal{S}_v, v \in \mathcal{V}, t \in \mathcal{T} \quad (A.67)$$

$$IDiS^{w1s_1}_{w_2s_2v} = \sum_{p\in\mathcal{E}_v} I\bar{D}iS^{w_1s_1,p}_{w_2s_2v} z_{pv} \in \{0,1\},$$
$$\forall w_i \in \mathcal{W}_{s_iv}, s_i \in \mathcal{S}_v, i = 1,2, v \in \mathcal{V} \quad (A.68)$$

$$ILS^{w_2v_2s}_{w_1v_1} \in \{0,1\}, \quad \forall w_i \in \mathcal{W}_{sv_i}, v_i \in \mathcal{V}_s, i = 1,2, s \in \mathcal{S} \quad (A.69)$$

$$IVS^{w_2s_2v_2}_{w_1s_1v_1} \in \{0,1\}, \quad \forall w_i \in \mathcal{W}_{s_iv_i}, s_i \in \mathcal{S}_{v_i}, v_i \in \mathcal{V}, i = 1,2, s \in \mathcal{S} \quad (A.70)$$

$$z_{pv} \geq 0, \quad \forall v \in \mathcal{V} \quad (A.71)$$

The objective function (A.56) minimizes the convex combination of the costs of the selected service plans. Constraints (A.57) and (A.58) make sure that the service plans that are used do not result in overlapping loading operations. It should be noted that the time availability restrictions of each supplier need not be taken into account, as the generated columns satisfy the supplier availability constraints. In fact, convex combinations of service plans could violate the time availability restrictions. For example, if loading at day $[t, t+1)$ is not allowed, and a feasible plan p_1 starts loading at time $t-1$, and another feasible plan p_2 starts loading at time $t+1$ then the convex plan $z_{p_1v_1} = z_{p_2v_2} = 0.5$ violates the supplier availability. However, the integrality condi-

tions that we pose later allow combinations of plans that start loading at the same day only, thereby recovering feasibility. The next set of constraints, (A.59)–(A.62), makes sure that a company-owned barge is not allocated to overlapping voyages. Constraints (A.63) enforce a convex combination of service plans, and (A.64)–(A.68) impose integrality restrictions on the in-vessel binary variables. Notice that imposing integrality restrictions on the service plan variables z_{pv} may not lead to an optimal solution (Degraeve and Jans 2007, Vanderbeck and Savelsbergh 2006). However, the integrality of the voyage indicators $IV_{ws\tau v}$ implies the integrality of the number of barges $B_{\tau sv}$ via (A.14), and therefore the integrality restrictions on the latter are omitted.

Solving the mixed integer program (A.56)–(A.71) involves the solution of its linear programming relaxation (LP), and then the implicit enumeration of the alternative configurations of the binary variables. An advantage of the LP relaxation is the small number of constraints: only the fact that loading operations, as well as voyages of the same barge, should not overlap needs to be considered. However the formulation is based on a combinatorial number of variables, which make it impractical in its current form. Formulations with such structure are solved with *column generation*: a few service plans are generated initially, and new service plans are generated as needed. Lübbecke and Desrosiers (2004) give an excellent up-to-date review of the method. Intuitively, column generation takes advantage of the fact that an optimal solution is formed by only a small subset of the service plans of each vessel. The method can be initialized with a set of variables that render the master problem feasible at a high cost, and then in each iteration the service plan that has the minimum reduced cost can be added. On termination no more service plans need to be added, and the linear programming relaxation of (A.56)–(A.71) is solved. The problem of finding the minimum reduced cost column is an integer problem itself, formulated in section 4.4.2.

4.4.2 Subproblem Formulation

We introduce the following notation:

$\gamma^{w_1 v_1}_{w_2 v_2 s} \leq 0$: dual price of constraints (A.57),

$\zeta^{w_1 s_1 v_1}_{w_2 s_2 v_2} \leq 0$: dual price of constraints (A.59),

$\theta^{w_1 s_1 v_1}_{w_2 s_2 v_2 b} \geq 0$: dual price of constraints (A.62), and

$\lambda_v \geq 0$: dual price of the convexity constraints (A.63). Each dual price is defined over the domain of the corresponding constraint. To facilitate exposition, we express each objective function component in isolation.

$$
\begin{aligned}
rc_{v1} =& Dm_v - Dp_v + \sum_{s\in\mathcal{S}_v}\sum_{\tau\in\mathcal{BT}_s} tonc_{\tau s}B_{\tau sv} + \sum_{s\in\mathcal{S}_v}\sum_{\tau\in\mathcal{BT}_s}\sum_{w\in\mathcal{W}_{sv}} detc_{\tau s}De_{ws\tau v}\\
rc_{v2} =& \sum_{s\in\mathcal{S}_v}\sum_{\tau\in\mathcal{BT}_s}\sum_{w\in\mathcal{W}_{sv}} IV_{ws\tau v}\sum_{v_1\neq v}\left\{tload_{\tau s}\sum_{w_1\in\mathcal{W}_{sv_1}}\gamma^{wv}_{w_1v_1s} + tdisch_{\tau v_1}\sum_{s\in\mathcal{S}_{v_1}}\sum_{w_1\in\mathcal{W}_{s_1v_1}}\zeta^{wsv}_{w_1s_1v_1}\right\}\\
rc_{v3} =& \sum_{s\in\mathcal{S}_v}\sum_{w\in\mathcal{W}_{sv}} L_{wsv}\sum_{v_1\neq v}\left\{\sum_{w_1\in\mathcal{W}_{sv_1}}(\gamma^{wv}_{w_1v_1s} - \gamma^{w_1v_1}_{wvs}) - \sum_{s_1\in\mathcal{S}_{v_1}}\sum_{w_1\in\mathcal{W}_{s_1v_1}}\zeta^{w_1s_1v_1}_{wsv}\right\}\\
rc_{v4} =& \sum_{s\in\mathcal{S}_v}\sum_{w\in\mathcal{W}_{sv}} Di_{wsv}\sum_{v_1\neq v}\sum_{s_1\in\mathcal{S}_{v_1}}\sum_{w_1\in\mathcal{W}_{s_1v_1}}\zeta^{wsv}_{w_1s_1v_1}\\
rc_{v5} =& \sum_{s\in\mathcal{S}_v}\sum_{w\in\mathcal{W}_{sv}}\sum_{b\in\mathcal{B}} IB_{wsvb}\left\{\sum_{v1>v}\sum_{s_1\in S_{v_1}}\sum_{w_1\in W_{s_1v_1}}\theta^{wsv}_{w_1s_1v_1b} + \sum_{v1<v}\sum_{s_1\in S_{v_1}}\sum_{w_1\in W_{s_1v_1}}\theta^{w_1s_1v_1}_{wsvb}\right\}\\
rc_v =& rc_{v1} - (rc_{v2} + rc_{v3} + rc_{v4} + rc_{v5} + \lambda_v)
\end{aligned}
\tag{A.72}
$$

The subproblem constraints are all in-vessel constraints of the original problem, i.e., all single-vessel constraints except from (A.19), (A.21),(A.41), (A.44) and (A.46). A new column is added whenever $rc_v < 0$, and the algorithm terminates when $\sum_v \min\{rc_v, 0\} = 0$. It should be noted that through the course of the algorithm a valid lower bound is at hand, calculated as $lb = \sum_{v\in\mathcal{V}}\sum_{p\in\mathcal{E}_v} c_{pv}z_{pv} - \sum_v \min\{rc_v, 0\}$ (Lübbecke and Desrosiers 2004). Therefore, if the subproblems cannot be solved to optimality, we can use feasible solutions to approximate the lower bound. This was necessary in our implementation because some subproblems could not be solved to optimality for medium and large vessels within the given time limit. Nevertheless, the approximate lower bound was still superior to the LP bound, as demonstrated by the computational experiments of section 4.7.

The downside of column generation algorithms is that there is no guarantee that a feasible solution will be recovered. Imposing the integrality restrictions when the column generation terminates could lead to an infeasible problem (Barnhart et al. 1998). In some applications recovery of a feasible solution from the lower bound solution is possible. In our application, a recovery strategy would require shifting the voyages in

such a way that there are no loading clashes and no overlaps of regular barges. These operations can be quite involved, but more importantly the quality of the feasible solution will depend on the quality of the lower bound solution, which may be far from optimality, especially when the algorithm terminates with an approximate lower bound only. Therefore, instead of relying on a lower bound to recover a feasible solution, we decided to design a local search procedure to deliver high quality solutions. The main building blocks of the local search procedure are described in section 4.5.

4.5 A Local Search Procedure

4.5.1 Initial Barge Allocation using Branch-and-Bound

Our approach decomposes the problem into a voyage *allocation* and a voyage *scheduling* part. The allocation part decomposes in a series of single vessel problems and focuses on minimizing the transportation cost per vessel, namely the fuel cost of regular barges and the fixed cost of leased and spot barges: $ac_v = \sum_{s \in \mathcal{S}_v} \sum_{\tau \in \mathcal{BT}_s} tonc_{\tau s} B_{\tau sv}$. The corresponding demurrage and despatch are disregarded in this phase and considered later during the scheduling phase. For each vessel, the decision variables, $B_{\tau sv}$, indicate the number of voyages from each barge type allocated to each supplier. Each voyage of a non-regular barge type will be allocated a non-regular barge. For regular barge types, decision variables do not reveal which particular barge will be allocated, but rather how many barges of each size will be allocated. We impose that the number of allocated voyages from each barge type across all suppliers should not exceed the maximum number of barges of that barge type, and that the cargo quantity to be loaded from each supplier is covered: $\min_{B_{\tau sv}} \{ac_v \mid \sum_{s \in \mathcal{S}_v : \tau \in \mathcal{BT}_s} B_{\tau sv} \leq nb_{\tau v}, \tau \in \mathcal{BT};\ \sum_{\tau \in \mathcal{BT}_s} cap_\tau B_{\tau sv} \geq q_{sv}, s \in \mathcal{S}_v\}$. Note that this is a multi-dimensional knapsack problem (Kellerer et al. 2004), with knapsack constraints for each supplier s, and upper bounds imposed on the disjoint sets of items $\mathcal{BT}_s$. We are able to solve these problems to optimality fast by branch-and-bound.

4.5.2 Schedule Generation using Greedy Sequencing

When the scheduling process is invoked, the barge-to-voyage allocations are fixed. Therefore, the transportation costs are fixed, and the aim is to minimize the total cost of demurrage and the detention of non-regular barges. We create a priority list of

vessels based on ascending ETA order and use a scoring criterion to break ties. The criterion utilized is the demurrage rate over the *laytime slack*, the later defined as the difference between the end of laytime and the minimum possible loading time. This score gives priority to vessels with higher demurrage costs per unit of safety time and was found to perform well in practice.

Scheduling of Current Barge Voyages. The scheduling module is designed so that it takes into consideration the current barge operations. For each vessel, voyages that are already in progress are scheduled first. We start by scheduling the voyages that are discharging onto the vessel. This way it becomes apparent when the vessel loading resources, i.e., floating cranes or geared grabs, are not available. Second we schedule voyages with barges waiting to discharge and third voyages with barges that sail to the port or process documents. Then voyages with barges that are loading are scheduled next, and the availability of each supplier's jetty is updated accordingly. Last we schedule voyages with barges sailing towards a supplier's jetty and with barges that return to the hub. After scheduling we record the time that each regular barge is released, as it poses a lower bound to the start of subsequent voyages.

Scheduling of Future Barge Voyages. The scheduling of future voyages is more involved than that of current voyages, because it takes into account the time availability of regular barges. If a voyage has to be allocated a regular barge, we allocate the regular barge that is first released from previous voyages. Voyage scheduling starts by determining the earliest feasible start of discharge for this barge, by taking into account the availability of the discharging equipment. Based on the derived discharging interval, we schedule the sailing and loading operations back-to-back with discharging, and check if the resulting loading interval at the supplier's jetty is feasible. Loading can be infeasible either because (i) the barge is not released promptly from previous voyages and cannot make it to the jetty on time or because (ii) the jetty is blocked at the proposed time of loading. If (ii) is true, we shift the loading backwards to the first feasible loading interval. This operation can fail, if either the jetty is occupied all the previous days, or if (i) is true, i.e., the barge is serving another voyage during that loading interval. In case backward shifting fails, we apply forward shifting. That is, we select the first feasible loading interval going forwards, taking also the jetty blockages and the barge availability. In case we resort to forward scheduling of loading, we forward schedule the discharging operation as well, by finding the earliest feasible

discharge time, given the end of loading. In case of vessels with geared grabs, we schedule the start of discharge as soon as a grab is released. If the earliest discharge interval implied by end of loading interval is infeasible, we shift discharging forward to the first feasible interval. Note that it is not possible to shift discharging backwards because this will result in infeasibility of loading. Also, for non-regular barges we perform both backward and forward shifting, and if both are feasible we compare the potential barge detention to the demurrage implied by the time lost due to forward shifting, and select the least expensive action. Finally, we record when each voyage is completed and update the corresponding barge availability of the regular barges.

Barge Swapping and elimination of non-regular barges. As soon as the current and future voyages have been scheduled, we apply a swapping procedure that poses a first-load first-discharge sequence across voyages. Given the scheduling procedure as described so far, two barges originating from the same supplier will not discharge their cargo in the sequence they load it. This is because the first barge will occupy the first available discharging and loading intervals, and the second barge will need to load before, if feasible, and discharge after the first barge. To avoid this situation, we swap the loading times of voyages A and B that originate from the same supplier whenever A loads before and discharges after B. In addition, we swap barges alongside the voyages, in order to guarantee that the voyage that loads first is allocated a barge that is not occupied then by another voyage. This procedure is implemented across all pairs of voyages that load at the same supplier.

At a final phase, we check for each voyage taken by a non-regular barge if there exists a regular barge that is idle during that voyage. In case such a regular barge exists, we replace the non-regular barge with that regular barge. This situation may occur when a non-regular barge is allocated to a voyage of the latest arriving vessel of a schedule. If that vessel has high demurrage it may be profitable to allocate a non-regular barge, but if later we allocate non-regular barges to voyages of previous vessels, then some regular barges may become available promptly for the voyages of the last vessel and therefore render non-regular barges redundant.

4.5.3 Improving the Initial Schedule by Local Search

The initial allocation takes into account the transportation cost only and, given that the regular barges are cheaper, it results in their maximum utilization. Therefore, the

schedule that is generated based on the initial allocation is likely to have high demurrage when the regular barges fail to start each of their allocated voyages on time. The local search algorithm checks, in a greedy manner, if replacing regular barges with leased or spot barges improves the joint demurrage and transportation costs. The intuition behind our strategy was derived by observing actual barge operations. In particular, most delays can be attributed to supply blockages, or late vessel arrivals. In both cases, barges are tied up and subsequently released in batches. Our local search procedure checks the improvement from replacing incrementally bigger batches of over utilized barge types.

First, we find the vessel with highest realized demurrage, and the type, τ_b, of the last-discharging regular barge, b, on this vessel. Then, we reoptimize the initial allocation problem amended by $\sum_{s \in \mathcal{S}_v : \tau_b \in \mathcal{R}_s} B_{\tau_b s v} \leq nb_{\tau_b v} - 1$. The goal of this modified allocation problem is to minimize the transportation cost as before, but also to replace the last barge of type τ_b with a barge of the next least expensive type. This way the transportation cost increases, but the vessel demurrage may decrease when the replacing barge can be readily available. In the context of our application, the modified allocation problem could be solved easily by enumeration because only three barge sizes are used. If the barge to be replaced is of the largest size, we replace it with a same-sized barge of the next cheapest available type. If it is of medium size, we select the least expensive of a large regular or a medium leased, if available, or a medium spot barge. Small size barges are used only for voyages that medium or large barges cannot take because of the low water level of the rivers in certain supplier locations, therefore they can be replaced only with small leased or small spot barges. A new schedule is generated using the modified barge allocation. If the schedule has an improved cost, the algorithm iterates by reselecting the vessel with highest realized demurrage and the type of the last discharging regular barge.

If the new schedule fails to improve the cost, we attempt to replace the types of the two last-discharging regular barges by modifying the initial allocation problem accordingly. We iterate by increasing the number we attempt to replace whenever a failure occurs, till we attempt to replace all regular barges. Every time a batch replacement fails to return an improved cost, we keep increasing the number of replaced barges till we reach the maximum number of regular barges allocated to that vessel. If still no improvement is made, we consider the vessel with the next highest realized

demurrage. The vessel in which we apply the replacement checks is the one with the highest realized demurrage that has not been fully checked till this point. This strategy qualifies as a greedy local search procedure because (i) there is an incremental movement at the solution space in each iteration and (ii) it progressively improves the current best schedule. The procedure terminates when all regular barge allocations across all vessels have been explored.

Algorithm 3 describes the main components of the local search procedure.

Algorithm 3: *Local Search*

Step 0. Solve the InitialAllocation problem for each vessel.
Step 1. Generate initial schedule: $\mathcal{S} \leftarrow$ GenerateSchedule(); $\mathcal{S}^* \leftarrow \mathcal{S}$.
Step 2. Initializations: $chkV \leftarrow \arg\max_v\{Dm_v\}$; $CheckedVessels \leftarrow \emptyset$; $r \leftarrow 1$.
Step 3. Find the last discharging r regular barges of $chkV$: $\{b_1, b_2, \ldots, b_r\}$.
Step 4. For each regular type $\tau \in \mathcal{R}$ find how many of $\{b_1, b_2, \ldots, b_r\}$ are of that type τ: λ_τ.
Step 5. Solve ModifiedAllocation$(chkV, \lambda_\tau)$ with constraints $\sum_{s \in \mathcal{S}_v} B_{\tau sv} \leq nb_\tau - \lambda_\tau$.
Step 6. Reschedule: $\mathcal{S} \leftarrow$ GenerateSchedule().
Step 7. If all regular barges of $chkV$ are checked put $chkV$ in $CheckedVessels$.
Step 8. **If** $chkV$ has still the highest demurrage across non-checked vessels **then**
If new schedule did not improve cost **then** increment checked barges: $r \leftarrow r + 1$.
ElseIf all vessels are checked **then** save best schedule: $\mathcal{S}^* \leftarrow Best(\mathcal{S}^*, \mathcal{S})$.
Terminate.
else Find next vessel with highest demurrage:
$chkV \leftarrow \arg\max_{v \notin CheckedVessels}\{Dm_v\}$.
Reset checked barges: $r \leftarrow 1$.
End If
Step 9. Save best schedule: $\mathcal{S}^* \leftarrow Best(\mathcal{S}^*, \mathcal{S})$. **Go to** Step 3.

4.6 Implementation and Impact

The barge rotation model was first implemented and used to optimize Noble's shipping operations in Indonesia, first in the South Kalimantan region and then the East Kalimantan region. In the near future, the model will be rolled out to Sumatra, with the

rest of the world following later. Below we report on the impact the model has already had on Noble's Indonesian operations since its first implementation in August 2012.

4.6.1 Decision Making prior to System Implementation

Prior to the implementation of the barge rotation model, logistics decisions were relying heavily on the experience and intuition of the logistics planners. Decision making was decentralized, in the sense that the product supply division were suggesting loading slots based on each supplier's availability, and the marketing division were negotiating the vessel arrival dates without considering the impact on logistics costs. The logistics division had to incorporate this information in order to construct economically sound rotation plans. As this information was updated at least twice per day, the workload of the schedulers was daunting.

To cope with this complexity, schedulers were using a basic spreadsheet that captured the short-term evolution of the operations. Updating of this spreadsheet was a tedious process which involved extracting information from multiple sources, such as supplier and floating crane availability and would require about two hours to update. Moreover, changes had to be made often as soon as new information would become available. For example, if a supplier was not able to deliver an agreed cargo at a certain date, then all loading intervals of this supplier had to be updated accordingly. This would lead to a tedious and error-prone redesign process. From a functional perspective, the planning spreadsheet had some important limitations. First, no costs were displayed or calculated, let alone minimized. Second, only the loading intervals were displayed in each voyage: the sailing times to and from the jetties, the documents processing time and the discharging intervals were not displayed, but rather estimated implicitly. Finally, it was not obvious that the barge allocations to subsequent vessels were feasible, as the time that each barge was released from each voyage was unknown.

The primary concern of the planning spreadsheet was to allocate barges to supplier jetties, in order to make sure that the total required quantities from each supplier would be transferred from the jetties to each vessel. However, there was no way of checking whether each barge would make it to its allocated jetty on time. As a result, delays on current operations were propagating to subsequent operations, generating significant demurrage. On an annual basis, penalties due to delays typically amounted to several

million dollars, and senior management believed that much of this could be avoided by using a more intelligent, optimized approach to barging operations.

4.6.2 Challenges and Adoption during Implementation

The logistics managers and schedulers were open to adopting a different approach, as they well understood the shortcomings of their manual process. Nevertheless, there were several challenges during adoption and implementation. First, the model requires a complete description of the current operations of the supply chain. Acquiring this data was not straightforward, and involved requesting information from other departments. For example, fuel prices and spot barge cost are kept by finance, whereas supplier availability and vessel arrival dates required input from marketing. These divisions were never asked to share their data before, and at first they were reluctant to do so. Second, the data entry itself is an error-prone process. It requires on a daily basis (i) vessel information, such as ETA, loading rates, cargo quantities from each supplier, demurrage rates, number of leased barges that can be used, (ii) barge information, such as size and contract type, current operation and estimated end of current operation and (iii) time availability of suppliers. The complexity of the data required the addition of many error-trapping tools into the model.

4.6.3 Implementation

Senior management requested that the model be built with a user interface in Excel. The original model and column generation algorithm were implemented in AIMMS using CPLEX v12.5 and interfaced with Excel. Experiments with Gurobi and alternative settings did not change the quality of the produced results. The local search procedure uses Excel for the initial allocation, while all other subroutines are coded in VBA. Figure 4.3 shows an example output as displayed by the barge rotation model. Laytime bars appear in green. Red bars denote demurrage and blue bars despatch. Each line that starts with a yellow bar corresponds to a barge voyage. Light blue bars indicate the slack time between consecutive voyages of the same regular barge. Barge loading and discharging operations are denoted with deep blue and gray respectively.

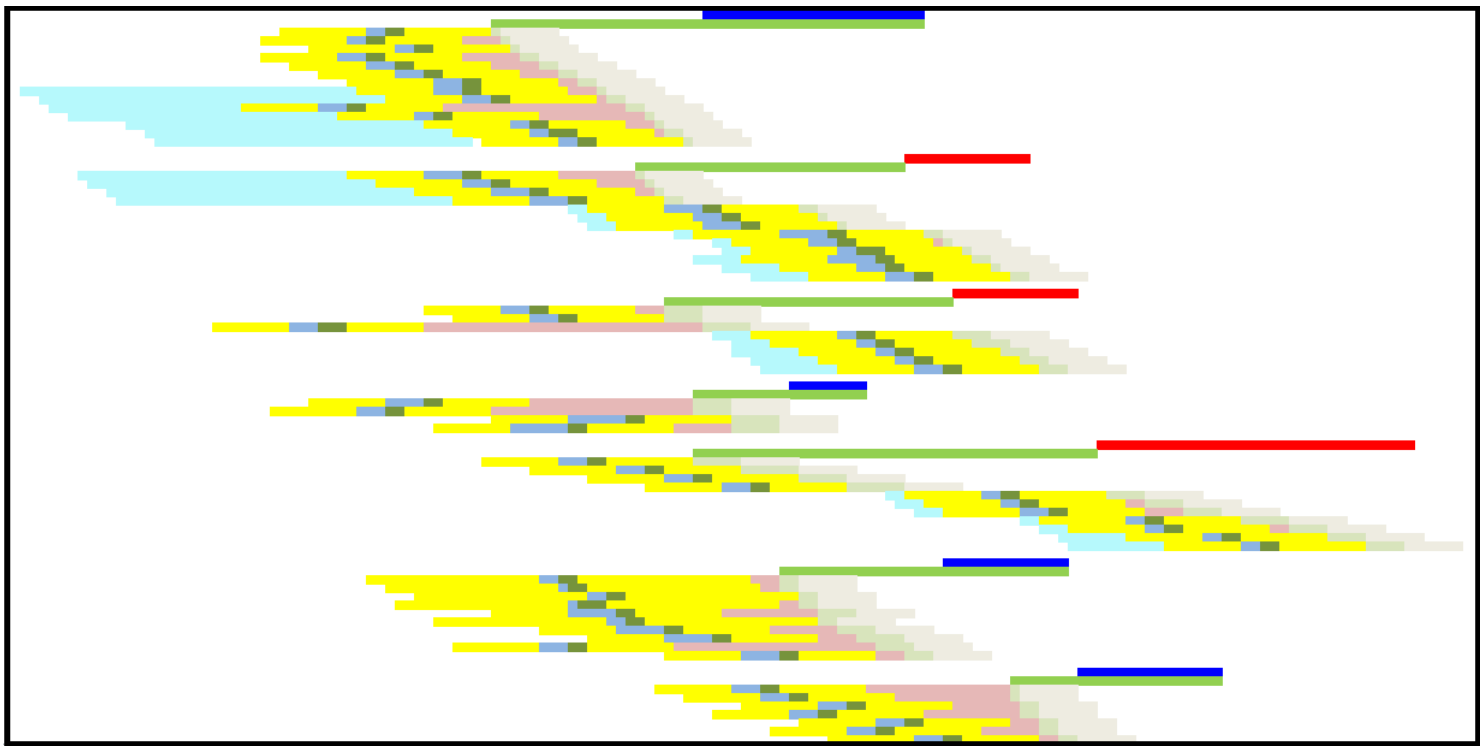


Figure 4.3: The Barge Rotation Model output.

4.6.4 Impact

Results collected so far indicate that the system has resulted in significant cost savings of more than $1 million per month. In particular, we look at the average unit cost, namely the total demurrage, despatch, spot, lease and fuel cost, per tonne carried. Figure 4.4 shows the evolution of the monthly average cost before and after the implementation of the barge rotation model in the two ports. The dotted lines show the monthly average cost per tonne during the entire observation period, whereas the bold lines show a six-month moving average, starting from the month that the model was rolled out in each port. The arrows indicate the implementation dates for each port.

Figure 4.4 reveals that the average cost per tonne is reduced from $3.8/tonne to $2.0/tonne in South Kalimantan, and from $3.7/tonne to $2.2/tonne in East Kalimantan, a reduction of $1.8/tonne and $ 1.5/tonne respectively. Given the actual number of carried tonnes during the observation period, the amount of savings for both ports combined is $1.3m per month, an estimated $15 million per year. A t-test indicates, for both ports, that the average cost per tonne before and after implementation are

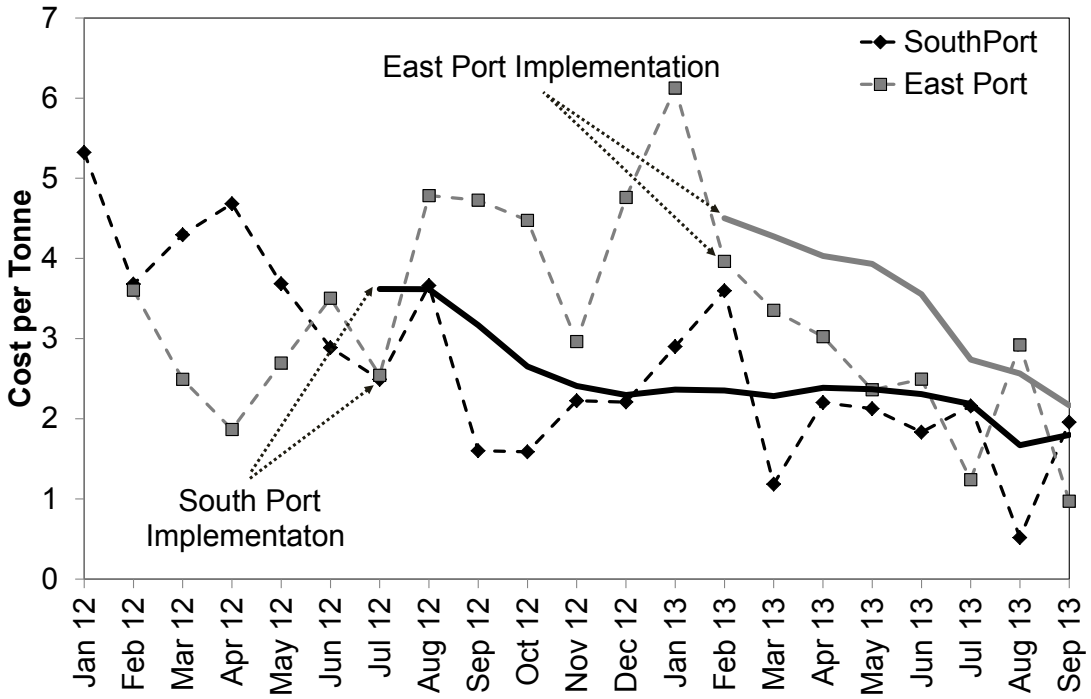


Figure 4.4: Evolution of cost per tonne and average cost per tonne at the two ports.

significantly different at a 5% level. We note, however, that the implementation of the barge rotation model may not be the only factor that explains costs variations. In particular, supplier availability, the amount of vessel traffic intensity, the availability of floating cranes and the oil price can have a direct impact on the cost per tonne. For example, the increase of cost per tonne in both ports between November 2012 and February 2013 might be attributed to a 17% increase of the oil price, which occurred during that period. In an attempt to isolate the effect of the model implementation, we carried out a regression analysis. We used an aggregate measure of supplier availability to control for cargo supply disruptions, and we used the carried tonnage per month as a proxy for vessel traffic intensity. As delays can propagate on following months, we also tested a version of carried tonnage lagged by one month. Further, we combined supplier availability with carried tonnage and lagged carried tonnage to test if heavy vessel traffic has an impact only when it is combined with bad supplier performance. Finally, we controlled for the cost of oil and for floating crane availability. We ran 4 variable selection methods, namely backward, forward, stepwise and best subsets regression,

in order to see which combination of explanatory variables yielded the best outcome, as measured by the adjusted R^2 of each model. The analysis was carried out for the South Kalimantan dataset, for which there is a balanced number of observations with and without the model implementation. Table 4.1 reports the results of the regression models.

Table 4.1: Regression analysis results for the South Kalimantan port.

Explanatory variable	**Best subset selection**	**Forward selection**
Supplier performance	2.34	–
	(1.34)	
System implementation	-2.47	-2.04
	(0.60)**	(0.45)**
Tonnes carried	–	–
Tonnes carried lagged	-4.61	-1.78
	(2.45)	(1.63)
(Tonnes carried)*(supplier performance)	2.95	–
	(2.60)	
(Tonnes carried lagged)*(supplier performance)	4.72	–
	(3.36)	
Oil price	0.06	0.05
	(0.02)*	(0.02)
Adjusted R^2	0.62	0.61

Notes. Standard errors of regression coefficients appear in parenthesis. Stepwise regression and backward regression selected the same model as best subset regression. $^{*}p < 0.05$; $^{**}p < 0.01$.

The best subset regression selects the subset of predictors that maximize the adjusted R^2 value. However, the resulting model suffers from multicollinearity, as *Supplier performance* and *(Tonnes carried lagged)*(Supplier performance)* have large variance inflation factors. This may be attributed to the limited number of observations, that can create coincidental correlations across explanatory variables. On the contrary, the forward selection model does not suffer from multicollinearity, autocorrelation, or overfitting, and has a high explanatory power. Using this model, the model implementation has a coefficient value of -2.04, significant at 1% level, suggesting that its impact is

actually more profound than that depicted by the difference of the average cost per tonne, namely $1.80, resulting in estimated savings of $9 million per year for South Kalimantan only. It is worth reporting that analysis of the East Kalimantan dataset also shows a consistent improvement after the model implementation, and Figure 4.4 shows a clear decline in average costs too; however the limited number of observations does not allow for any robust conclusions at this point in time.

4.7 Computational Demonstrations

The purpose of this section is twofold. First, we benchmark the performance of the barge rotation model against an off-the-shelf implementation. Second, we analyze the fundamental issue underlying the barge rotation problem, namely the trade-off between hiring extra infrastructure, i.e., spot barges, versus delaying the service of vessels. More specifically, we investigate how congestion caused by short vessel interarrival times and the price of spot barges impact the economic viability of hiring spot barges.

4.7.1 Efficiency Analysis

We demonstrate the efficiency of our approach using six artificially generated instances and one instance with real data. Table 4.2 describes the characteristics of the dataset.

Table 4.2: Characteristics of the test instances.

Instance	Vessels	Suppliers	Voyages	Barge Sizes	Contract Types	Regular Barges
Small A	2	1	3	1	1	3
Small B	2	2	19	2	1	7
Medium A	4	1	3	1	1	3
Medium B	4	2	19	2	1	7
Large A	6	1	3	1	1	3
Large B	6	2	19	1	1	7
Real	8	9	26	3	2	14

Instances vary from small, medium to large, based on the number of vessels. Moreover, instances labeled A are of the simplest possible structure, namely those with 1 supplier, a fixed number of 3 voyages per vessel, barges of one size, with regular and

spot categories, and with a fleet of 3 regular barges. Instances labeled B have a richer structure, as they include vessels with 2 suppliers, barges of 2 sizes, and therefore a varying number of voyages to each supplier, and with a fleet size of 7 regular barges. The real instance is based on the situation in South Kalimantan on 3 August 2013. In table 4.3, we compare the performance of the barge rotation model against CPLEX v12.5 with default settings.

Table 4.3: Numerical results for the comparison of CPLEX and the barge rotation model (BRM).

	Lower Bound		**Upper Bound**		
Instance	**CPLEX**	**BRM**	**CPLEX**	**BRM**	**BRM UB Time (s)**
$SmallA$	22,500	13,500	22,500	25,100	2
$SmallB$	8,660	158,900	339,000	257,266	6
$MediumA$	19,953	34,139	92,750	92,750	6
$MediumB$	-249,199	323,700	N/A	324,243	20
$LargeA$	19,712	64,245	1,525,000	136,250	10
$LargeB$	-183,348	543,000	N/A	799,900	18
$Real$	539,169	581,576	N/A	713,902	120

Notes. Time limit is 3 hours for all instances.

Table 4.3 clearly shows that obtaining good lower and upper bounds is hard from a computational perspective. CPLEX with default settings failed to find a feasible solution for 3 of our 7 instances, when using formulation $(A.1) - (A.55)$. In addition, the lower bound that CPLEX finds is, in the majority of cases, very weak. Note that negative values are possible, because despatch is awarded when a vessel is fully loaded within its laytime. The lower bound generated by the Barge Rotation Model was always superior to that found by CPLEX, with $SmallA$ being the only exception. Note that in theory the column generation lower bound dominates the root node lower bound of the original model, but in practice CPLEX may improve the lower bound of the linear programming relaxation by generating cuts. Perhaps surprisingly, the column generation lower bound quality improves as the problem structure becomes richer. The bigger the in-vessel complexity, the better the column generation lower bound quality, because it captures that complexity explicitly in the subproblems. This makes our

approach especially useful for problems of realistic size, and the last instance indeed shows good performance of column generation. On a practical note, both lower bounds are sensitive to the magnitude of the utilized big-M values (Codato and Fischetti 2006). Column generation however obviates this issue because it includes the vessel-specific big-M constraints to the subproblem level, which is more tractable. Therefore only the big-M constraints of the master may affect the lower bound quality. In terms of computation time, our barge rotation model requires only a limited amount of time, and when the model was tested on real data with as many as 16 vessels, computation times increased almost linearly, never taking more than 10 minutes, thereby ensuring a time-robust performance.

4.7.2 Economic Analysis

In this section, we use a stylized case to derive insights on the impact that spot barge cost and vessel interarrival times have on the structure of the optimal solution. Specifically, we investigate how the cost of spot barges, vessel demurrage and interarrival times influence the optimal number of spot barges, via an example case. Table 4.4 and figure 4.5 present the example data.

Table 4.4: Stylized example data

Attribute	**Value**
Vessels	2
Suppliers	1
Barge Sizes	1
Spot Contracts	1
Regular Barges	4
Voyages per vessel	3
$\frac{Demurrage\ rate}{Despatch\ rate}$	5

All vessel characteristics are identical except for their time of arrival. Both vessels use a floating crane each, and source coal from the same supplier. Since there are 4 regular barges and 6 voyages in total, there will be at most 2 spot barges used. Figure 4.6 shows the number of total barges utilized in the optimal solution, when we vary the vessel interarrival times and the cost of spot barges.

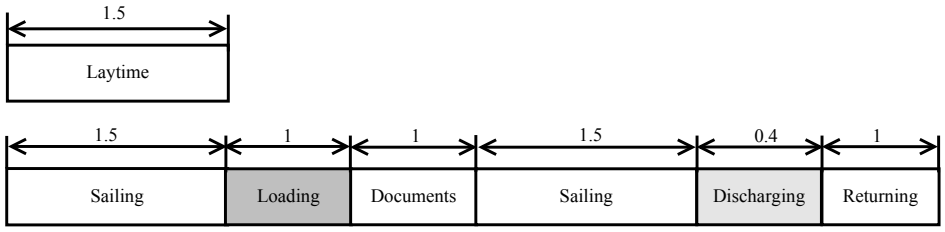


Figure 4.5: Process durations for the stylized example

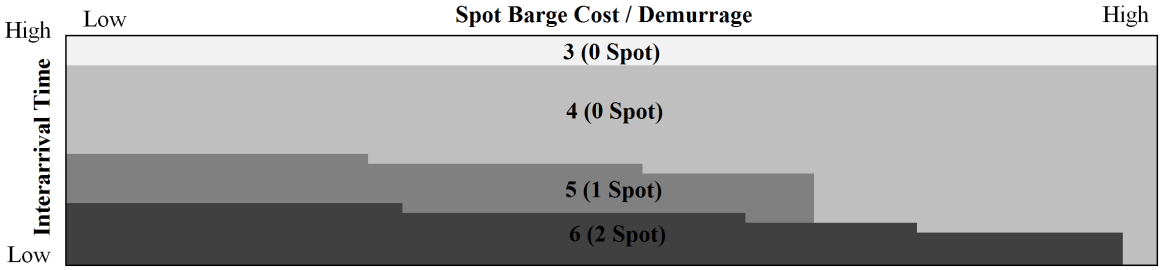
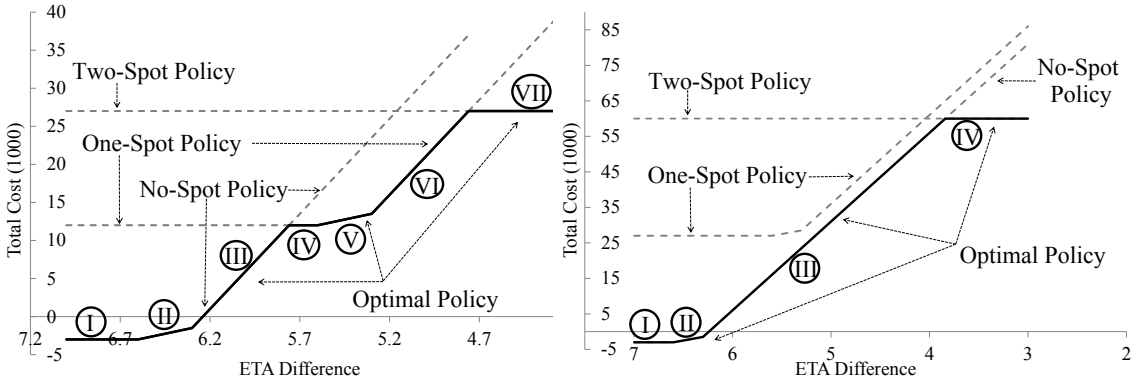


Figure 4.6: Number of barges used at optimality for different configurations of interarrival time and spot barge cost.

Each pair of spot barge cost and interarrival times corresponds to a fully characterized data instance. We created 1,722 such pairs that were solved to optimality. We observe that both the spot barge cost and the interarrival time influence how many spot barges should be hired. When the interarrival time is very large, it is possible to utilize only 3 regular barges and serve both vessels without any delay. As the interarrival times become smaller, the optimal solution utilizes all 4 regular barges. When vessels arrive even closer to each other, it is optimal to hire 0, 1 or 2 spot barges, depending on their relative cost versus the demurrage that can be avoided by hiring them. To further investigate this behavior, figure 4.7 shows the total cost when 0, 1 and 2 spot barges are hired, for small and large values of spot barge cost respectively.

In both graphs of figure 4.7, the bold lines show the optimal barge hiring policy, which is the lower envelope of the total cost for the 0, 1 and 2 spot barge policies. In region I of both graphs vessels arrive far from each other, and not hiring spot barges is optimal. As the interarrival times become smaller in region II, the no-spot hiring policy remains optimal, but the despatch obtained from the late-arriving vessel



(a) Case 1: low barge cost

(b) Case 2: high barge cost

Figure 4.7: Optimal cost and the structure of optimal policies

decreases, and in region III, despatch turns into demurrage, resulting in a faster increase of total cost. When spot barges have a low cost, as in case 1 of figure 4.7, then there is a threshold after which hiring 1 spot barge yields a lower cost than no hiring spot barges, shown in regions IV, V and VI. Interestingly, when spot barges are expensive relative to demurrage, as in case 2, it ultimately becomes optimal to hire 2 spot barges immediately, as in region IV of case 2.

In the context of our application, spot barges have high cost when compared to vessel demurrage. Therefore, the managerial insight we derive is that in periods where vessel congestion is high, the optimal replacement policy has an all-or-nothing structure. A similar result was given recently by Chen et al. (2013) for an inventory management problem, where the authors show that when procuring from a regular supplier becomes expensive, the buyer will source solely from the spot market. We also exploit this insight in our algorithm, namely in the local search procedure described in section 4.5, where we simultaneously replace sets of regular barges by spot barges, taking into account that the optimal region switches from hiring no spot barges to hiring the maximum possible.

4.8 Discussion and Learnings

Maritime transportation is an area with relatively few applications of mathematical programming models. At present, many problems arising in transshipment operations are

beyond the capabilities of modern solvers, both because of their structure and of their large size. To the best of our knowledge, this is the first practical work that describes the development of a decision support system that allows (i) multiple contracting options in a (ii) complex maritime scheduling setting, that is (iii) fed by real-time data and (iv) with high impact on the efficiency of the transshipment operations.

The application we report on this paper is interesting from two different perspectives. First, its mathematical formulation is beyond the capabilities of modern integer programming software, and as such it offers fruitful ground for theoretical studies. The formulation can be conceptualized as a series of generalized hybrid jobshop flow scheduling problems appended by linking constraints, that prohibit the overlap of loading operations and of voyages taken by the same barge. Second, the model constitutes a good testbed for improving generic heuristic techniques. As mathematical programming technology develops at a fast pace, we envisage that the usage of optimization models in maritime environments will increase in the future, leading ultimately to more efficient operations and greater integration of maritime practitioners with the operations research community.

4.9 References

Agostinho Agra, Marielle Christiansen, and Alexandrino Delgado. Mixed integer formulations for a short sea fuel oil distribution problem. *Transportation Science*, 47:108–124, 2013.

F. Al-Khayyal and S. J. Hwang. Inventory constrained maritime routing and scheduling for multi-commodity liquid bulk, part i: Applications and model. *European Journal of Operational Research*, 176(1):106–130, 2007.

Mikhail J. Atallah and Marina Blanton, editors. *Algorithms and Theory of Computation Handbook: General Concepts and Techniques*. Chapman & Hall/CRC, 2 edition, 2010. ISBN 978-1-58488-822-2.

Cynthia Barnhart, Ellis L Johnson, George L Nemhauser, Martin WP Savelsbergh, and Pamela H Vance. Branch-and-price: Column generation for solving huge integer programs. *Operations research*, 46(3):316–329, 1998.

Youhua Frank Chen, Weili Xue, and Jian Yang. Optimal inventory policy in the presence of a long-term supplier and a spot market. *Operations Research*, 61(1):88–97, 2013.

M. Christiansen, K. Fagerholt, B. Nygreen, and D. Ronen. *Handbooks in Operations Research and Management Science 14, Transportation.* Elsevier B.V., 2007.

Gianni Codato and Matteo Fischetti. Combinatorial benders cuts for mixed-integer linear programming. *Operations Research*, 54, 2006.

George B. Dantzig and Philip Wolfe. Decomposition principle for linear programs. *Operations Research*, 8(1):pp. 101–111, 1960.

Zeger Degraeve and Raf Jans. A new dantzig-wolfe reformulation and branch-and-price algorithm for the capacitated lot-sizing problem with setup times. *Operations Research*, 55 (5):909–920, 2007.

Faramroze G. Engineer, Kevin C. Furman, George L. Nemhauser, Martin W. P. Savelsbergh, and Jin-Hwa Song. A branch-price-and-cut algorithm for single-product maritime inventory routing. *Operations Research*, 60(1):106–122, 2012.

Marshall L Fisher. The lagrangian relaxation method for solving integer programming problems. *Management science*, 50(12 supplement):1861–1871, 2004.

Christodoulos A. Floudas and Xiaoxia Lin. Mixed integer linear programming in process scheduling: Modeling, algorithms, and applications. *Annals of Operations Research*, 139 (1):131–162, 2005.

Kevin C. Furman, Jin-Hwa Song, Gary R. Kocis, Michael K. McDonald, and Philip H. Warrick. Feedstock routing in the exxonmobil downstream sector. *Interfaces*, 41(2):149–163, 2011.

Michael R. Garey and David S. Johnson. *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1990.

R. Jaikumar and M. Solomon. The tug fleet size problem for barge line operations: A polynomial algorithm. *Transportation Science*, 21(4):pp. 264–272, 1987.

Hans Kellerer, Ulrich Pferschy, and David Pisinger. *Multidimensional Knapsack Problems*. Springer, 2004.

Yuri Levin, Mikhail Nediak, and Huseyin Topaloglu. Cargo capacity management with allotments and spot market demand. *Operations Research*, 60(2):351–365, 2012.

Marco Lübbecke and Jacques Desrosiers. Selected topics in column generation. *Operations Research*, 53:1007–1023, 2004.

George G. O'Brien and Roger R. Crane. The scheduling of a barge line. *Operations Research*, 7(5):pp. 561–570, 1959.

J. A. Persson and M. Gothe-Lundgren. Shipment planning at oil refineries using column generation and valid inequalities. *European Journal of Operational Research*, 163(3):631–652, 2005.

Octavio Richetta and Richard C. Larson. Modeling the increase complexity of new york city's refuse marine transport system. *Transportation Science*, 31(3):272, 1997.

Ruben Ruiz and Jose Antonio Vazquez-Rodriguez. The hybrid flow shop scheduling problem. *European Journal of Operational Research*, 205(1):1 – 18, 2010.

N. L. Schwartz. Discrete programs for moving known cargos from origins to destinations on time at minimum bargeline fleet cost. *Transportation Science*, 2(2):pp. 134–145, 1968.

G. Don Taylor, Todd C. Whyte, Gail W. DePuy, and D.J. Drosos. A simulation-based software system for barge dispatching and boat assignment in inland waterways. *Simulation Modelling Practice and Theory*, 13(7):550–565, 2005.

UNCTAD. Review of maritime transport 2012. Technical report, UNCTAD, 2012.

Franois Vanderbeck and Martin W.P. Savelsbergh. A generic view of dantzig-wolfe decomposition in mixed integer programming. *Operations Research Letters*, 34(3):296–306, 2006.

Michael R. Wagner and Zinovy Radovilsky. Optimizing boat resources at the U.S. coast guard: Deterministic and stochastic models. *Operations Research*, 60(5):1035–1049, 2012.

5

Technical Appendix

This technical appendix that explains the most important methods that are utilized throughout the thesis. Specifically, the basic concepts underlying Dantzig-Wolfe decomposition, column generation, Lagrange relaxation, subgradient optimisation and branch-and-price are described. The goal of this chapter is to serve as a quick reference to the methods that are used throughout the thesis and facilitate their exposition to the reader. For this reason, it draws material from the two most comprehensive references in this area, namely Barnhart et al. (1996) and Lübbecke and Desrosiers (2004).

5.1 Dantzig-Wolfe Decomposition

Before we explain the concept of Dantzig-Wolfe Decomposition we give some useful definitions.

Definition 5.1 (Convex Set) *A set S is called convex iff for each s_1, s_2 in S and λ in $[0,1]$ the point $s = \lambda s_1 + (1-\lambda)s_2$ is in S.*

Definition 5.2 (Convex Hull) *We define the convex hull of a set S as $conv(S) = \{s \mid \exists s_1, s_2 \in S, \lambda \in [0,1] : s = \lambda s_1 + (1-\lambda)s_2\}$.*

Collorary 5.3 *If S is convex then $conv(S) \equiv S$. If S is not convex then $S \subseteq conv(S)$.*

Definition 5.4 (Relaxation of a set S) *We relaxation of a set S any set R with $S \subseteq R$.*

From the definition of relaxation it holds that $conv(S)$ is a relaxation of S. Moreover, $conv(S)$ is the *minimal* convex relaxation of S in the sense that it is a subset of any other convex set that is also a relaxation of S.

Definition 5.5 (Extreme point) *A point p of a convex set S is an extreme point of S when for each p_1, p_2 in S and λ in $[0,1]$ $p = \lambda p_1 + (1-\lambda)p_2 \iff p = p_1 = p_2$.*

Definition 5.6 (Extreme ray) *A ray r of a convex set S is a point such that for each λ in $\Re_+$, λr is in S. A ray is extreme if for any two rays r_1, r_2 and λ in $\Re_+$ it holds that $r = \lambda r_1 + (1-\lambda)r_2 \iff \frac{r}{\|r\|} = \frac{r_1}{\|r_1\|} = \frac{r_2}{\|r_2\|}$*

We consider a mathematical program P of the generic form

$$P \quad \min c^T x \tag{A.1}$$
$$\text{s.t.} \quad Ax \geq b \tag{A.2}$$
$$x \in X \tag{A.3}$$
$$x \text{ integer} \tag{A.4}$$

Dantzig-Wolfe Decomposition uses the fact that the set $X^* = \{x \in X : x \text{ integer}\}$ can be represented by a finite set of vectors, to derive an alternative definition of the variables of P. Specifically, when X is bounded then X^* is a finite set of points and when X is not bounded it can be represented as a convex combination of a finite set of extreme points and a linear combination of a finite set of extreme rays (Schrijver 1986). Therefore, if $P = \{p_1, \ldots, p_n\}$ are the extreme points and $R = \{r_1, \ldots, r_m\}$ the extreme rays of X^*, any $x \in X^*$ can be written as

$$x = \sum_{p_i \in P} \lambda_i p_i + \sum_{r_j \in R} \kappa_j r_j \in \text{integer}$$

for some $\lambda_i \geq 0, i = 1, \ldots, n$ with $\sum_{i=1}^{n} \lambda_i = 1$ and $\kappa_j \in \Re_+, j = 1, \ldots, m$. Substituting this expression in P yields the Dantzig-Wolfe form of P, DWP:

$$DWP \quad \min c^T \sum_{p_i \in P} \lambda_i p_i + c^T \sum_{r_j \in R} \kappa_j r_j \tag{A.5}$$
$$\text{s.t.} \quad A \sum_{p_i \in P} \lambda_i p_i + A \sum_{r_j \in R} \kappa_j r_j \geq b \tag{A.6}$$
$$\sum_{p_i \in P} \lambda_i p_i + \sum_{r_j \in R} \kappa_j r_j \in \text{integer} \tag{A.7}$$

$$\sum_{i=1}^{n} \lambda_i = 1 \tag{A.8}$$

$$\lambda_i \geq 0, i = 1, \ldots, n \tag{A.9}$$

$$\kappa_j \in \Re_+, j = 1, \ldots, m \tag{A.10}$$

For the remainder of the chapter we make the assumption that the set X^* is finite and bounded, and therefore it can be represented by extreme points only. This is the most common case in practice, and the only relevant case for the decompositions developed in this thesis. Problem DWP is a reformulation of P and as such, it attains the same optimal objective value. Moreover, every solution of P can be mapped to a solution of DWP and vice versa. The potential advantage of DWP compared to P is that, whenever the convex hull of X^* does not have integer extreme points, its linear programming relaxation can yield a better lower bound than the linear programming relaxation of P (Geoffrion 1974). To see this, we note that the feasible set of solutions of the linear programming relaxation of P is $P_{LP} = \{x|Ax \geq b\} \cap \{x \in X\}$, while the corresponding feasible set for DWP is $DWP_{LP} = \{x|Ax \geq b\} \cap \{x|x \in conv(X^*)\}$, and $DWP_{LP} \subseteq P_{LP}$ because $conv(X^*) \subseteq X$, since $conv(X^*)$ is the minimal convex relaxation of X^* and X is an arbitrary convex relaxation of X^*. This is important because a weak lower bound leads to a larger enumeration tree in a branch-and-bound algorithm that aims to find an optimal solution. The potential disadvantage of DWP is that it may have a large number of variables. Column generation is an algorithmic procedure that finds an optimal solution of the linear programming relaxation of DWP without explicitly considering all of its variables. The next section describes column generation in detail.

5.2 Column Generation

Each extreme point of X^* is associated with a variable in DWP. In the majority of applications, X^* has too many extreme points, and considering them explicitly is usually not an efficient option. However, most of these points and their corresponding variables will be zero at an optimal solution of DWP, and therefore it is not necessary to consider them explicitly. The concept of column generation is to identify the subset of variables that are non-zero at an optimal solution. The process starts with a restricted set of variables and iteratively identifies the most promising variable that

is not considered so far and adds it to DWP. The next steps describes in more detail the column generation process. The term column is used to describe the vector of coefficients of a variable in DWP.

Step 1. *Generate an initial set of columns.* An initial set of columns can be generated in several ways. The most trivial way is to generate artificial columns, one for each constraint of (A.2), and penalize them by a high cost at the objective function (Chvátal 1983). A strategy that is usually more efficient is to identify a set of feasible solutions of P and use them as initial columns. This is also easy from a practical viewpoint, as feasible solutions can be generated readily by MIP solvers or heuristics.

Step 2. *Find the variable that maximally improves the current objective value.* This task can be recast as a linear program in which the reduced cost of each variable is used as a proxy for maximal improvement. The reduced cost of a variable that is not used in the current solution, is the amount by which the objective function will change if this variable is set at unit level. For variable x_i, it is defined as $rc_i = c_i - \pi^T A_i$, where π is the dual values of (A.2) in the current optimal solution, c_i is the objective coefficient of variable x_i and A_i the column of A that corresponds to the coefficients of x_i.

Step 3. *Solve the linear programming relaxation of DWP or terminate.* From step 2, we have calculated which variable maximally improves the current objective value of DWP by solving the *pricing problem* $\min\{rc = (c^T - \pi^T A)x \mid x \in conv(X^*)\}$. If $rc > 0$ then no column can further improve the objective function and the algorithm terminates. If $rc < 0$, the column that returned the minimum reduced cost is entered into the relaxation of DWP and the procedure iterates: a new dual solution π is obtained, and the algorithm returns to step 2.

5.3 Lagrange Relaxation

An alterative approach to solving P is *Lagrange relaxation* (Fisher 2004). The basic idea is to relax constraints $Ax \geq b$ and penalize their violation $v = Ax - b$ in the objective function, using a set of weights $u \geq 0$. The problem can then be formulated as follows:

$$L(u) := \min_{x \in X^*} c^T x - u^T (Ax - b)$$

The above problem is known as *Lagrange subproblem*, and weights u are the *Lagrange multipliers*. It is easy to see that any solution is a valid lower bound of P for each $u \geq 0$. The problem of selecting the w that maximizes the Lagrange subproblem is known as the *Lagrange dual*, and is formulated as:

$$\text{LRP} : \mathcal{L} := max_{u \geq 0} L(u) = max_{u \geq 0} \{ \min_{x \in X^*} c^T x - u^T (Ax - b) \}$$

The Lagrange function $L(u), u \geq 0$ is the lower envelope of a family of functions linear in u and as such it can be shown to be concave (Lübbecke and Desrosiers 2004). Then it follows that it is differentiable everywhere on its domain other than in its breakpoints, namely the points $u \geq 0$ in which the minimum value $L(u)$ is attained by two or more members of the family $f_x = c^T x - u^T (Ax - b),\ x \in X^*$. A nice result by Geoffrion (1974) shows that the lower bound $\mathcal{L}$ is the same as the optimal objective value of DWP. The proof uses duality arguments, and it shows that the dual of DWP can be recast as a Lagrange dual.

The fact that LRP is concave means that it has a unique maximizer and therefore convex optimisation methods can be utilised. However, LRP is only subdifferentiable in its breakpoints, in which a gradient function is not defined. In each breakpoint u, the optimisation problem $\min_{x \in X} c^T x - u^T (Ax - b)$ has multiple solutions, and for each solution x there exists a gradient direction $d_x = Ax - b$. A *subgradient* at a point u_b is a direction $d_x = Ax - b$ for which it holds that $L(u) - L(u_b) \geq d_x (u - u_b)$ for any $u \geq 0$. Subgradient optimization is a method that solves LRP approximately, but updating the set of multipliers $u \geq 0$ iteratively till a convergence criterion is met. Specifically, given an initial point $u^0 \geq 0$ subgradient optimisation is a procedure that produces a sequence of estimates $u^1, \ldots, u^m$ of u by applying the formula

$$u^{k+1} = \max\{0, u^k - v_k (b - Ax^k)\}$$

where u^k is the estimate of u in step k, v_k is a scalar parameter called *step size* and x^k the solution of $\min_{x \in X^*} \{c^T x - u^{kT} (Ax - b)\}$. The stepsize v_k is an important parameters that can influence the convergence of subgradient optimisation. A popular formula used in practice is

$$v_k = \frac{\lambda_k (L(u_k) - Z^F)}{\|Ax_k - b\|^2}$$

where Z^F is the best known feasible solution and λ_k is a number between 0 and 2 which is adjusted dynamically depending on the convergence of u_k. Specifically, $\lambda_k = 2$ is considered a good starting value, that leads to large steps in promising directions of improvement. As the sequence $\{u^0, \ldots, u^k\}$ approaches the vicinity of the optimal solution λ_k is reduced to almost zero, in order to allow for a more refined search strategy.

When the subgradient algorithm terminates, a lower bound of P is at hand. This procedure can be used as a bounding subroutine in a branch-and-bound algorithm, which is an enumeration scheme that eliminates options that cannot be optimal. However, finding a good branching direction can be a challenge, as the optimal solution returned by the Lagrange subproblem is integral but typically violates the constraints $Ax \geq b$. It is desirable to obtain a solution that does not violate $Ax \geq b$ but violates the integrality restrictions $x \in Integer$, because then standard branching schemes can be adopted. Unfortunately, there is no generic way to obtain such a structure with subgradient optimisation, so most methods resort to column generation. Specifically, the solution $x \in X^*$ that lead to the optimal $u \geq 0$ can be used a column of DWP and then column generation will generate the other alternative optima $x \in X^*$, and their convex combination ensures that $Ax \geq 0$ is not violated.

Decomposable structures. In the vast majority of applications the set X has a decomposable structure, namely it can be written as $X = \bigcup_{i=1}^{k} X_i$ with $X_i \cap X_j = \emptyset$, for all $i, j \in \{1, \ldots, k\}$ with $i \neq j$. This implies that the subproblem decomposes in a series of independent suproblems of smaller size. This makes the solution process more efficient computationally, as subproblems can be solved fast and in parallel. This statements holds not only for Lagrange relaxation, but also for column generation.

5.4 Branch-and-Price and Branch-and-Cut

After a lower bound of of the objective value function of P is at hand, an enumeration scheme can be employed. In particular, the bounding process is applied iteratively to variants of P in which some integer variables are fixed into some integer values. The operation of fixing variables in certain values is called *branching*, because it creates problems of the form of P. A lower bound is calculated for each problem that is created, and whenever it is higher than the best known objective value of P, the corresponding problem is rejected. Specifically, this is a proof that fixing the integer variables in this

particular fashion will lead to an objective value of P that is no better than the existing one, and therefore this fixing configuration can be rejected. The search stops when all possible combinations of integer variables have been considered, either explicitly, or implicitly.

This technique has been given many names in the literature, such as *branch-and-bound, implicit enumeration* and *divide and conquer*. When the lower bound is obtained using column generation, it is usually called *branch-and-price*. When inequalities that cutoff non-feasible solutions of P are added in its linear programming relaxation, branch-and-bound is called *branch-and-cut*. Branch-and-cut is the approach adopted by state-of-the-art software that solves mathematical programs, such as CPLEX and Gurobi. The implementation of branch-and-price requires structural information about each problem, and this hinders its adoption as a standard method in commercial software.

5.5 References

Cynthia Barnhart, Ellis L. Johnson, George L. Nemhauser, Martin W. P. Savelsbergh, and Pamela H. Vance. Branch-and-price: Column generation for solving huge integer programs. *Operations Research*, 46:316–329, 1996.

Vaŝek Chvátal. *Linear Programming*. Freeman, 1983.

Marshall L. Fisher. The lagrangian relaxation method for solving integer programming problems. *Manage. Sci.*, 50(12 Supplement):1861–1871, December 2004. ISSN 0025-1909. doi: 10.1287/mnsc.1040.0263. URL `http://dx.doi.org/10.1287/mnsc.1040.0263`.

A.M. Geoffrion. Lagrangian relaxation for integer programming. 2:82–114, 1974.

Marco Lübbecke and Jacques Desrosiers. Selected topics in column generation. *Operations Research*, 53:1007–1023, 2004.

Alexander Schrijver. *Theory of Linear and Integer Programming*. John Wiley & Sons, Inc., New York, NY, USA, 1986. ISBN 0-471-90854-1.

6

Conclusions

With an ever-growing amount of data readily available in many areas of operations management, optimisation methodologies face new opportunities and challenges. Efficient solution procedures for large-scale models, such as this presented in the third essay, have yet to be developed. This thesis contributes to the advancement of large-scale solution methodologies in three different aspects. First, it shows how algorithmic techniques, such as Lagrange relaxation and column generation can be made more efficient by amending a formulation such that its dual space is reduced. This leads to important efficiency gains, and ultimately paves the way to considering applications of larger-scale than what can be considered to date. Second, the development of a horizon decomposition approach demonstrates how problems that expand over a large horizon can be decomposed and solved efficiently. Lastly, it demonstrates how Dantzig-Wolfe decomposition can be used as a lower bounding technique in a complex practical problem, and presents a dedicated algorithm that generates high-quality feasible solutions.

There are many avenues for further research related to the current thesis. An emerging stream of literature investigates stabilisation techniques in column generation and Lagrange relaxation. The quest for efficient solution approaches that arise in areas such as transportation, advertising and revenue management has boomed in recent years, and problems considered intractable have been tackled by insightful applications of column generation and Lagrange relaxation. Problems that span over a multitude of periods pose important computational challenges, and the horizon-decomposition approach developed in this thesis constitutes a well-suited approach to multi-period problems. Future research is needed to demonstrate in which classes of problems horizon

decomposition approach is mostly efficient. Finally, the practical application presented in the third essay is beyond the capabilities of modern solvers, even those that encapsulate the latest advancements in integer programming theory. Theoretical studies are needed to investigate the structures of the underlying polytopes and suggest families of inequalities that improve the formulation's lower bound. In addition, it provides fruitful ground for research on heuristics based on mixed integer programming, as solvers failed to find any feasible solutions for medium and large scale instances.

It is the author's belief that theoretical advancements on mathematical programming theory and on heuristics can ultimately have a large impact on practical operations, and thereby increase the visibility of the academic operations management community. As globalised operations environments become increasingly more complex, and as data availability is better than before, large-scale optimisation has the opportunity to become a ubiquitous research area of increased importance. The current thesis, in addition to offering methodological advancements, constitutes a step towards demonstrating the practical applicability of large-scale optimisation methods to the ever-changing landscape of operations management.

Declaration

I herewith declare that I have produced this paper without the prohibited assistance of third parties and without making use of aids other than those specified; notions taken over directly or indirectly from other sources have been identified as such. This paper has not previously been presented in identical or similar form to any other English or foreign examination board.

The thesis work was conducted from 2009 to 2013 under the supervision of Bert De Reyck at University College London.

London, 2013