A Combinatorial Benders' decomposition for the lock scheduling problem

J. Verstichel[a,], J. Kinable[a,c], P. De Causmaecker[b], G. Vanden Berghe[a]

[a]*KU Leuven Department of Computer Science, CODeS, Gebroeders De Smetstraat 1, 9000 Gent, Belgium*
[b]*KU Leuven Department of Computer Science, iMinds-ITEC, Etienne Sabbelaan 53, 8500 Kortrijk, Belgium*
[c]*KU Leuven Faculty of Economics and Business, ORSTAT, Naamsestraat 69, 3000 Leuven, Belgium*

Abstract

The Lock Scheduling Problem (LSP) is a combinatorial optimization problem that represents a real challenge for many harbours and waterway operators. The LSP consists of three strongly interconnected sub problems: scheduling lockages, assigning ships to chambers, and positioning the ships inside the chambers. These should be interpreted respectively as a scheduling, an assignment, and a packing problem. By combining the first two problems into a master problem and using the packing problem as a sub problem, a decomposition is achieved that can be solved efficiently by a Combinatorial Benders' approach. The master problem is solved first, thereby sequencing the ships into a number of lockages. Next, for each lockage, a packing sub problem is checked for feasibility, possibly returning a number of combinatorial inequalities (cuts) to the master problem. The result is an exact approach to the LSP. Experiments are conducted on a set of instances that were generated in correspondence with real world data. The results indicate that the decomposition approach significantly outperforms other exact approaches presented in the literature, in terms of solution quality and computation time.

Keywords: Lock Scheduling Problem, Combinatorial Benders' Decomposition

Email address: jannes.verstichel@cs.kuleuven.be (J. Verstichel)

1. Introduction

The Port of Antwerp (Belgium), one of the largest harbours in Europe, processed more than 180 MTE (Million Tonnes Equivalent) of cargo and 70000 ships in 2012, with an average of almost 200 ships a day (Port of Antwerp, 2012). It is a major hub for both inland and intercontinental cargo traffic. The harbour is situated at the river Scheldt with tidal differences averaging five meters. In order to ensure a persistent water level within the harbour, locks separate the docks from the main water way. These locks entail a complex optimization problem: the vast number of ships entering and leaving the harbour every day have to be assigned to lock chambers, their exact position inside the locks need to be determined, and the lockages have to be scheduled. Improving the efficiency of the lock operations could reduce the expensive waiting time of ships, and make the port more attractive for business and economy.

The Albertkanaal is an important inland waterway connecting the Port of Antwerp with the Port of Liège. Over the years, numerous industrial activities have emerged on its banks, leading to over 37 MTE of cargo processed in 2012 (nv De Scheepvaart, 2012). Six locks are used to overcome the height difference of 56 meters between Antwerp and Liège. The increase of barge traffic and recent periods of drought make it of paramount importance to reduce the number of lockage operations (i.e. water usage) and the waiting times of ships.

The present contribution is a new, fast exact approach to the lock scheduling problem (LSP) based on a combinatorial Benders' decomposition approach. The LSP is decomposed into a master and a sub problem. The master problem (MP) first assigns the ships to lock chambers, after which it attempts to schedule lockages. The sub problem (SP) takes care of positioning the ships inside the lock chambers. Whenever the sub problem identifies an infeasible lockage, i.e. a set of ships that cannot be transferred simultaneously due to the chamber's capacity or safety constraints, combinatorial inequalities (cuts) are generated and added to the master problem. The master problem and sub problem are solved iteratively, until a provable optimal schedule is obtained.

The main focus of this paper is on the decomposition approach and its application to LSP, thereby omitting detailed discussions on the sub problems as they exhibit a large number of application specific constraints. Section 2 describes the LSP. Section 3 provides a literature review. Section 4 presents

the Benders' decomposition approach, defining the master problem and the sub problem in detail, as well as their interaction. Particular attention goes to the generation of feasibility cuts for the master problem, as they largely determine the efficiency of the algorithm. Experiments are conducted on a large number of instances based on data obtained from different Belgian locks. The results are presented in Section 5. Section 6 offers the conclusions.

2. Problem Outline

The Lock Scheduling Problem consists of three interconnected sub problems: an assignment, a packing and a scheduling problem. Each problem comes with a large number of constraints, mainly resulting from safety and nautical regulations. The problem has been described in detail by Verstichel et al. (2014a); here we only sketch the general outline. The main lock scheduling specific terms used throughout this paper are elucidated in Figure 1, which shows a lock with two identical parallel chambers.

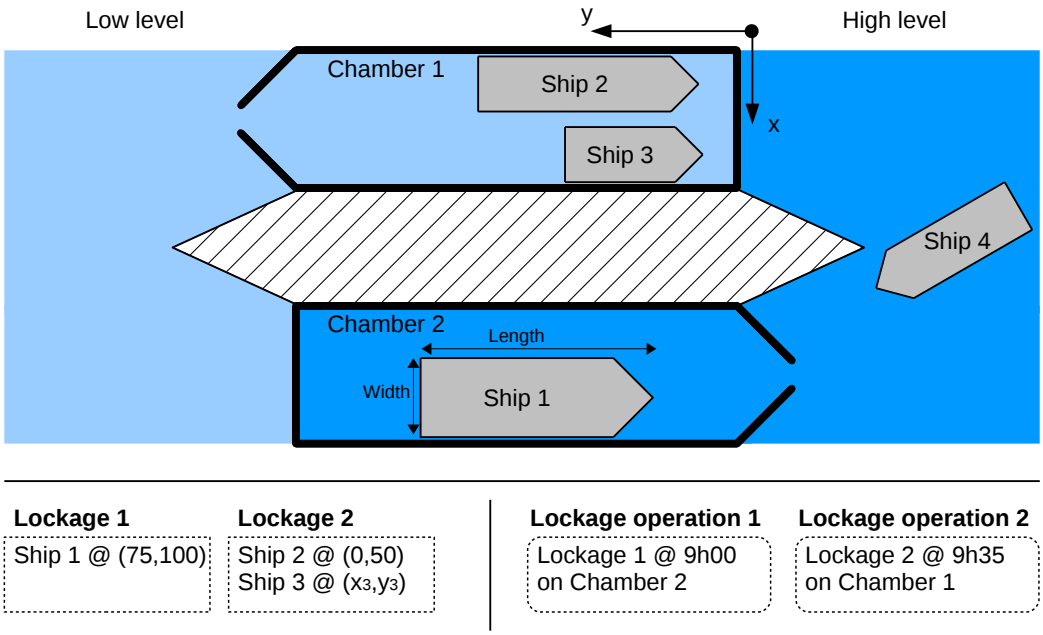

Figure 1: An example of a lock with two parallel chambers ($|T| = 1$, $|U_1| = 2$), four ships travelling through the lock ($|N^1| = 3$, $|N^2| = 1$) and important lock scheduling specific terms.

2.1. Scheduling, Assignment and Packing

A lock consists of one or more chambers, which can perform lockage operations independently of each other. Each chamber is of a specific type $t \in T$, defining the chamber's dimensions, transfer speed, etc. The set of chambers of the same type is denoted by U_t.

A number of ships N need to traverse the lock, either in the upstream, or in the downstream direction. Upstream (resp. downstream) ships are denoted as N^1 (resp. N^2), $N^1 \cap N^2 = \emptyset$, $N^1 \cup N^2 = N$. For each ship $i \in N$, an arrival time r_i is known, as well as the dimensions of the ship. The ships must be grouped into a number of batches (i.e. lockages), where each batch contains the ships that are transferred in a single lockage operation. The set of all lockage operations is denoted by M, thereby distinguishing upstream M^1 and M^2 downstream lockages ($M = M^1 \cup M^2$). Each lockage operation $k \in M$ needs to be assigned to a physical chamber $u \in U_t$ that will execute the lockage. We therefore distinguish between lockages $M_t \subseteq M$ of a specific type $t \in T$, i.e. lockage operations that can be performed on a chamber of type $t \in T$.

All lockages $M_t, t \in T$ need to be distributed over the available chambers U_t, while adhering to a strict schedule. A lockage $k \in M$ cannot commence before the last ship assigned to the lockage has arrived. The duration of the lockage depends both on the processing times ps_i of the ships $i \in N$ assigned to the lockage, plus a constant time pc_t depending on the type $t \in T$ of chamber that performs the lockage.

A chamber is always in one of it's two possible states, which handle either downstream or upstream transfer. Each transfer switches the lock's state. Consequently, two consecutive upstream (or downstream) lockage operations on the same chamber require an empty lockage to be scheduled in between to switch the chamber's state. Formally, given two consecutive lockages, $k, l \in M_t$, scheduled on a chamber of type $t \in T$, a transition time s_{lk} is needed to put the chamber in the correct state, i.e. $s_{lk} = pc_t$ if k, l are both upstream (or downstream) lockages, $s_{lk} = 0$ otherwise. A first-come-first-served (FCFS) policy is often enforced by the lock authorities. Therefore ship i must depart from the lock no later than ship j should ship i arrive at the lock before ship j. It is assumed that no two ships ever arrive at the lock at exactly the same time.

Assigning ships to a specific lockage operation requires evaluation of a number of constraints. Obviously, the ships assigned to a single lockage $k \in M_t$, may not surpass the capacity of a chamber of type $t \in T$. In

addition, the exact location of a ship inside the chamber $u \in U_t$ has to be determined, while complying with a number of safety restrictions. In short, verifying whether a set of ships can be assigned to the same lockage operation, amounts to solving a complex rectangle packing problem, where each rectangle represents a ship (Verstichel et al., 2014b).

LSP corresponds to a traditional machine scheduling problem with sequence dependent setup times: a set of tasks (ships) are grouped into jobs (lockages) which need to be assigned to machines (lock chambers).

An overview of the parameters of the LSP is provided in Table 1.

Parameters:	
N, N^1, N^2	$N = N^1 \cup N^2$ is the set of ships, subdivided in upstream ships N^1 and downstream ships N^2.
T	Set of different chamber types.
M	$M = M^1 \cup M^2$ is the set of lockages, thereby distinguishing between upstream M^1 and downstream M^2 lockages.
M_t	$M_t = M_t^1 \cup M_t^2$ is the set of lockages suitable for chambers of type $t \in T$, again distinguishing between resp. upstream and downstream lockages. Note that M_t is an ordered set, i.e. $M_t = \{1, 2, \ldots, m_t^1, m_t^1+1, \ldots, m_t^1+m_t^2\}$, where m_t^i, $i = 1, 2$, are bounds on the number of upstream resp. downstream lockages for chamber type $t \in T$.
U_t	Set of chambers of type $t \in T$.
pc_t, ps_i	The minimal processing time of a chamber of type $t \in T$, and the processing time of ship $i \in N$.
r_i	Time at which ship $i \in N$ arrives at the lock.
s_{lk}	Transition time required between two consecutive lockages $k, l \in M_t, t \in T$ performed on the same chamber. $s_{lk} = pc_t$ if k, l are both upstream (or downstream) lockages, $s_{lk} = 0$ otherwise.
W_t, L_t:	Width and length of a chamber of type $t \in T$ (integer)
w_i, l_i:	Width and length of ship $i \in N$ (integer)
C_{max}:	Sufficiently large big-m parameter

Table 1: Parameters used throughout the paper.

3. Literature review

LSP was first introduced in an inland setting by Verstichel and Vanden Berghe (2009), who presented a heuristic approach. Although it is capable of efficiently solving large instances, the heuristic does not provide any insights as to the quality of the solutions. An in depth analysis of the ship placement sub problem of the LSP was made by Verstichel et al. (2014b), who presented several exact and heuristic solution approaches and a decomposition method for this variant of two dimensional rectangular bin packing. These methods were tested extensively on both generated and real-life data, showing the practical applicability of the presented algorithms. Later, Verstichel et al. (2014a) presented a mathematical model for the generalized LSP applicable to both inland and port settings, along with an exact monolithic branch-and-bound procedure. Due to the complex nature of LSP, only relatively small instances were solved to optimality. Verstichel et al. (2014b) present a detailed literature review on the lock scheduling problem and its variations. The discussion in the remainder of this section will therefore be limited to an introduction to Benders' decomposition, and a number of its applications to related problems.

Benders' decomposition is a mathematical approach that exploits the fact that fixing a number of difficult variables in a mathematical model may simplify the problem considerably. The decomposition approach divides a problem into a master problem (MP) and a sub problem (SP), which are solved iteratively. The MP, considering a subset of the variables, is solved first. Next, the sub problem is solved for the remaining variables, while temporarily fixing the variables' values of the MP. Finally, based on the outcome of the SP, one or more cuts are generated and added to the MP, thereby effectively preventing the MP from revisiting similar areas of the search space.

Traditional (classical) Benders' decomposition (Benders, 1962), considers the SP as a linear programming problem, where cuts are derived from its dual solution. In more recent work, e.g. Geoffrion (1972) and Hooker and Ottoson (2003), the Benders' decomposition approach has been generalized to a broader class of problems, no longer requiring the sub problem to be linear. Hooker and Ottoson (2003) introduced the concept of Logic Based Benders' decomposition. In contrast to traditional Benders' decomposition, cuts are not necessarily obtained from the dual formulation of a linear sub problem, but through the so-called inference dual. Whenever the sub prob-

lem is a feasibility problem, the inference dual is a condition which, when satisfied, implies that the master problem is infeasible (Rasmussen and Trick, 2007). This condition can then be used to obtain Benders' cuts to cut off infeasible solutions. A particular case of Logic Based Benders' decomposition, frequently referred to as Combinatorial Benders' decomposition, is discussed by Codato and Fischetti (2006), where it is applied to mixed-integer programming (MIP) problems involving large numbers of logical implications (big-M constraints). Whenever a particular assignment of variable values in the MP renders the SP infeasible, a Combinatorial Benders' cut is generated and added to the master problem. This cut, stating that at least one of the variables in the master problem must change its value, distils a logical implication from the original model and adds it to the master problem. Note that this approach is ineffective for continuous variables. Stronger Combinatorial cuts may be obtained by identifying small subsets of variables responsible for the infeasibility of the sub problem, and expressing cuts in terms of these variables. The smallest of these subsets are referred to as Minimum Infeasible Subsets (MIS). The latter approach is well suited for the LSP, as will be explained in detail in Section 4.

A number of successful applications of Logic-Based Benders' decompositions to related packing, scheduling and assignment problems illustrate its potential. Bai and Rubin (2009) investigate the allocation of tollbooths to roads, thereby minimizing the number of tollbooths required to cover the entire road network. Similar to the LSP, the tollbooth problem suffers from a large number of conditional (big-M) constraints. By decomposing the problem, many of these conditional constraints can be omitted. A master problem assigns the tollbooths to roads. Subsequently, the sub problem verifies whether the proposed assignment is feasible. Whenever an infeasible solution is encountered, cover cuts are generated, stating that at least one tollbooth must be placed on a specific subset of roads. Côté et al. (2013) apply Combinatorial Benders' decomposition to the Strip Packing Problem (SPP). A relaxed version of the SPP is solved first, thereby treating the SPP as a Parallel Processor Scheduling Problem. Next, the sub problem attempts to reconstruct a feasible solution to the SPP based upon the relaxed solution. Whenever this is not possible, a combinatorial cut is added to the master problem, requiring that at least one rectangle must change its position. Strong cuts are obtained by identifying small subsets of rectangles responsible for the infeasibility of the sub problem.

Tran and Beck (2012) solve a Parallel Machine Scheduling Problem (PMSP)

with machine and sequence dependent setup times through Logic Based Benders' decomposition. The master problem assigns jobs to machines, while the sub problem minimizes the makespan for each individual machine by determining the job sequence. This is efficiently realized by a dedicated TSP solver. Note that, due to the fact that the machines are independent, these sequencing sub problems may be solved in parallel. To strengthen the master problem, a relaxed version of each sub problem is added. Based on a comparative study, Tran and Beck (2012) claim a six orders of magnitude speedup compared to a traditional Branch-and-Bound approach.

4. A Combinatorial Benders' Decomposition

Verstichel et al. (2014a) attempted to solve the LSP via a single, large, Mixed Integer Linear Programming problem. The present paper introduces a Benders' decomposition which splits the LSP in a master problem and a sub problem. The advantage of this decomposition is that part of the complexity of the problem is shifted to a separate sub problem, thereby obtaining two simpler problems. In addition, efficient dedicated algorithms can be employed to solve the master and sub problem, whereas there may not exist an algorithm capable of tackling the entire problem at once. Finally, a number of logical implications modeled through inefficient big-M constraints by Verstichel et al. (2014a) are no longer required, as they will be enforced through addition of cuts to the master problem. The presented method is based on Codato and Fischetti (2006)'s original algorithm. The main differences are 1) an integer programming sub problem (LSP) in contrast to a linear one (Codato and Fischetti, 2006), and 2) a constructive algorithm for determining minimal infeasible subsets (MIS), whereas Codato and Fischetti (2006) determined MIS through an LP.

The master problem is provided with a list of ships that need to traverse the lock, the direction in which the ships need to traverse the lock, and their arrival times at the lock. The master problem first partitions the list of ships in an arbitrary number of non-overlapping subsets. Each set represents a group of ships that will be transferred in a single lockage operation. Obviously, each subset contains only ships that traverse the lock in the same direction. Next, the master problem proposes a schedule for the generated subsets, thereby determining the exact starting times of the lockage operations. Subsequently, the sub problem verifies, for each subset, whether the ships in this set can be transferred simultaneously, i.e. whether they fit to-

gether inside the lock chamber. The latter appears to be a decision version of the ship placement problem (Verstichel et al., 2014b). The LSP is solved whenever an optimal MP schedule is determined in which each subset satisfies the packing constraints of the sub problem. Whenever the sub problem identifies an infeasible combination of ships, a *feasibility cut* is generated and added to the master problem, thereby preventing the master problem from assigning these ships to a single lockage operation.

The following two sections discuss the master problem and sub problem in detail. An overview of the entire algorithm is given in Procedure 1.

4.1. Master problem

The following Mixed Integer Linear Programming problem defines the master problem. In order to keep the model concise, some problem specific constraints were omitted, e.g. constraints that manage tidal windows, ship dependent pre- and post-processing times, ship draft, etc. Similarly, some redundant constraints that enable to speed up the model are not included in the problem description below. The complete model is available from (Verstichel et al., 2014a). The parameters are defined in Table 1; the variables (marked in **bold**) are discussed below the model.

$$min \;\; \lambda_1 \sum_{k} \boldsymbol{z_k} + \lambda_2 \sum_{i \in N} \boldsymbol{c_i} + \lambda_3 \boldsymbol{T_{max}} \tag{1}$$

s.t.

$$\sum_{k \in M^j} \boldsymbol{f_{ik}} = 1, \quad \forall i \in N^j, j = 1, 2 \tag{2}$$

$$\boldsymbol{f_{ik}} \leq \boldsymbol{z_k}, \quad \forall i \in N, k \in M \tag{3}$$

$$\boldsymbol{z_{k+1}} \leq \boldsymbol{z_k}, \quad \forall k \in M_t, t \in T \tag{4}$$

$$\boldsymbol{c_i} \geq C_{max}(\boldsymbol{f_{ik}} - 1) + \boldsymbol{C_k}, \quad \forall i \in N, k \in M \tag{5}$$

$$\boldsymbol{P_k} \geq pc_t \boldsymbol{z_k} + \sum_{i \in N} ps_i \boldsymbol{f_{ik}}, \quad \forall k \in M_t, t \in T \tag{6}$$

$$\sum_{u \in U_t} \boldsymbol{proc_{ku}} = \boldsymbol{z_k}, \quad \forall k \in M_t, t \in T \tag{7}$$

$$\boldsymbol{proc_{ku}} + \sum_{v \in U_t, v \neq u} \boldsymbol{proc_{lv}} + \boldsymbol{seq_{kl}} \leq 2 \tag{8}$$
$$\forall k, l \in M_t,\ l > k, \forall u \in U_t,\ t \in T$$

$$\boldsymbol{C_l} - \boldsymbol{C_k} + 2C_{max}(3 - \boldsymbol{seq_{kl}} - \boldsymbol{proc_{ku}} - \boldsymbol{proc_{lu}}) \geq \boldsymbol{P_l} + s_{kl} \tag{9}$$
$$\forall k, l \in M_t, l > k, \forall u \in U_t, t \in T$$

$$\boldsymbol{C_k} - \boldsymbol{C_l} + 2C_{max}(2 + \boldsymbol{seq_{kl}} - \boldsymbol{proc_{ku}} - \boldsymbol{proc_{lu}}) \geq \boldsymbol{P_k} + s_{lk} \tag{10}$$
$$\forall k, l \in M_t, l > k, \forall u \in U_t, t \in T$$

$$\boldsymbol{C_k} \geq r_i \boldsymbol{f_{ik}} + \boldsymbol{P_k}, \quad \forall i \in N, k \in M \tag{11}$$

$$\boldsymbol{T_{max}} \geq \boldsymbol{c_i} - r_i, \quad \forall i \in N \tag{12}$$

The master problem uses several sets of variables. The main variables are binary variables $z_k, k \in M$, denoting whether lockage $k \in M$ is used, binary variables $f_{ik}, i \in N, k \in M$ denoting whether ship $i \in N$ is assigned to lockage $k \in M$, integer variables $P_k, k \in M$ denoting the processing time of lockage $k \in M$, integer variables $C_k, k \in M$ denoting the completion time of lockage $k \in M$ and finally T_{max} denoting the maximal transfer time over all ships passing the lock. The auxiliary integer variables $c_i, i \in N$ denote the departure time of each ship at the lock, the binary variables $proc_{ku}, k \in M, u \in U_t, t \in T$ denote whether lockage $k \in M$ is executed by chamber $u \in U_t, t \in T$ or not, while $seq_{kl}, k, l \in M$ denote whether lockage $k \in M$ precedes lockage $l \in M$ in the same chamber or not. The objective function, equation (1), minimizes (a) the number of lockages, (b) the sum of all ship's departure times from the lock and (c) the maximum waiting time of a ship at the lock, where $\lambda_1, \lambda_2, \lambda_3$ are independent weight factors. In this work, $\lambda_1 = 0.1$, and $\lambda_2 = \lambda_3 = 1.0$. This represents a common lock operator policy prioritizing

total waiting time minimization (λ_2) over maximum waiting time (λ_3) and the total number of lockages (λ_1) required to transfer all ships. Other values would favor, for example, the number of lockages, maximum waiting time, etc. Constraints (2)-(4) assign ships to lockage operations. Constraints (2) ensure that each ship is assigned to a lockage. Obviously, downstream ships cannot be assigned to upstream lockages and vice versa. Constraints (3) are linking constraints; a lockage $k \in M$ is used, i.e. $z_k = 1$, if at least a single ship is assigned to it. Note that the lockage operations are ordered (Constraint (4)): lockage z_{k+1} cannot be active if z_k is not used, $k \in M_t, t \in T$. The remaining constraints determine the scheduling part of the problem. A ship cannot leave the lock before the lockage operation for that ship is completed (Constraints (5)). The duration of a single lockage operation depends on a fixed value plus an additional amount per ship (Constraints (6)). Each lockage operation must be mapped to a physical lock chamber (Constraints (7)). When two lockage operations are executed by the same physical chamber, they must be sequenced (Constraints (8)). Constraints (9), (10) describe the actual scheduling restrictions on the lockages per chamber. Verstichel et al. (2014a) provide a detailed discussion on these scheduling constraints. A lockage cannot commence before all ships have arrived at the lock (Constraints (11)). Finally, Constraint (12) records the maximum waiting time of a ship at the lock.

4.2. Sub problem

Once the master problem has assigned the ships to a number of lockages, the feasibility of these lockages needs to be verified. i.e for all $k \in M_t : z_k = 1$, $t \in T$, a small sub problem is solved to test whether the given configuration of ships fits inside a chamber of type t. Whenever a configuration is considered infeasible, a combinatorial cut will be generated and added to the master problem. The latter will be elaborated in the next section.
Let $\overline{N_k} = \{i \in N : f_{ik} = 1\}$ be the set of ships assigned to lockage $k \in M$. For a given lockage $k \in M$, we obtain the following rectangle packing problem. Note that this version of the problem is a satisfiability problem; it has no objective. The parameters of the model are summarized in Table 1 (page 5).

$$\boldsymbol{x_i} + w_i \leq W_t \qquad \forall i \in \overline{N_k} \qquad (13)$$

$$\boldsymbol{y_i} + l_i \leq L_t \qquad \forall i \in \overline{N_k} \qquad (14)$$

$$\boldsymbol{left_{ij}} + \boldsymbol{left_{ji}} + \boldsymbol{b_{ij}} + \boldsymbol{b_{ji}} \geq 1 \qquad \forall i < j,\ i,j \in \overline{N_k} \qquad (15)$$

$$\boldsymbol{x_i} + w_i \leq \boldsymbol{x_j} + W_t(1 - \boldsymbol{left_{ij}}) \qquad \forall i \neq j,\ i,j \in \overline{N_k} \qquad (16)$$

$$\boldsymbol{y_i} + l_i \leq \boldsymbol{y_j} + L_t(1 - \boldsymbol{b_{ij}}) \qquad \forall i \neq j,\ i,j \in \overline{N_k} \qquad (17)$$

$$\text{safety constraints} \qquad \forall i \neq j,\ i,j \in \overline{N_k} \qquad (18)$$

$$\text{mooring constraints} \qquad \forall i \neq j,\ i,j \in \overline{N_k} \qquad (19)$$

The integer variables x_i, y_i model resp. the x and y coordinates of a ship $i \in \overline{N_k}$ inside the chamber. In addition, auxiliary (binary) variables $left_{ij}, b_{ij}$, $i,j \in \overline{N_k}$, record resp. whether ship i is located left of ship j, and whether ship i is located behind ship j.
Constraints (13)-(14) ensure that the x and y coordinates of a ship $i \in \overline{N_k}$ are located within the chamber's dimensions. The remaining constraints ensure that the ships do not overlap. Finally, as stated before, some problem specific constraints have been omitted here for clarity. The full problem is described by Verstichel et al. (2014b).

Algorithm 1: Combinatorial Benders' Decomposition of the lock scheduling problem

Input: Set of ships N, arrival times and ship properties, lock parameters

add initial cut(s) to MP
repeat $\leftarrow$ true
while repeat **do**
  Solve MP
  get solution $(\boldsymbol{z_k}, \boldsymbol{f_{ik}}, \boldsymbol{C_k})$, $\forall i \in N, k \in M$
  repeat $\leftarrow$ false
  for $k \in M : z_k = 1$ **do**
    Solve SP for $\overline{N_k} = \{i \in N : \boldsymbol{z_{ik}} = 1\}$
    if SP is infeasible **then**
      repeat $\leftarrow$ true
      add feasibility cut(s) to MP
    else
      get solution $(\boldsymbol{x_i}, \boldsymbol{y_i})$, $\forall i \in \overline{N_k}$
    end if
  end for
end while
return Optimal schedule $(\boldsymbol{f_{ik}}, \boldsymbol{C_k}, \boldsymbol{x_i}, \boldsymbol{y_i})$, $\forall i \in N, k \in M$

4.3. Combinatorial Benders' cuts

When an infeasible sub problem is encountered, one or more combinatorial Benders' cuts are generated and added to the master problem, effectively preventing the master problem from assigning specific ships to the same lockage.

The general form of cuts considered is:

$$\sum_{i \in S \subseteq N} \boldsymbol{f_{ik}} \leq |S| - 1 \qquad \forall k \in K' \subseteq M \tag{20}$$

Stated informally, this cut prevents the ships in $S \subseteq N$ from being assigned to the same lockage k.

A straightforward 'no-good' cut arises from an infeasible sub problem, i.e. an infeasible lockage of type $t \in T$ with $\overline{N_k}$ ships, by setting $S = \overline{N_k}$ and $K' = M_t$. These no-good cuts can be very weak, especially if $|S|$ is large. Stronger cuts may be obtained by considering smaller infeasible subsets of ships. The strongest cuts are based on minimum infeasible subsets (MIS).

In this context, a MIS is a subset of ships that cannot be transferred in a single lockage of type $t \in T$; removing any of the ships from the set would however result in a feasible lockage. Computing all MIS for a given set of ships $N' \subseteq N$ would be notoriously hard. It would require solving the sub problem from Section 4.2 for every possible subset of N'. Section 4.4 presents approaches to computing strong cuts requiring far less computational effort.

Combinatorial Benders' cuts can be generated at different times in the process: *Initial cuts* are added to strengthen the MP before the first MP-SP iteration. Applying initial cuts reduces the number of infeasible MP lockages generated. *Feasibility cuts* are generated on-the-fly every time the MP generates a solution. These cuts are applied so as to cut away infeasible parts of the search space and guide the MP towards a feasible lock scheduling solution.

4.4. Cut separation

The different cut separation methods are clarified using the example from Figure 2, where the MP proposes a solution in which ships 1 through 7 are assigned to a single lockage. The feasible lockages for this example (under a first-come-first-served policy) are displayed on the right side of this figure.

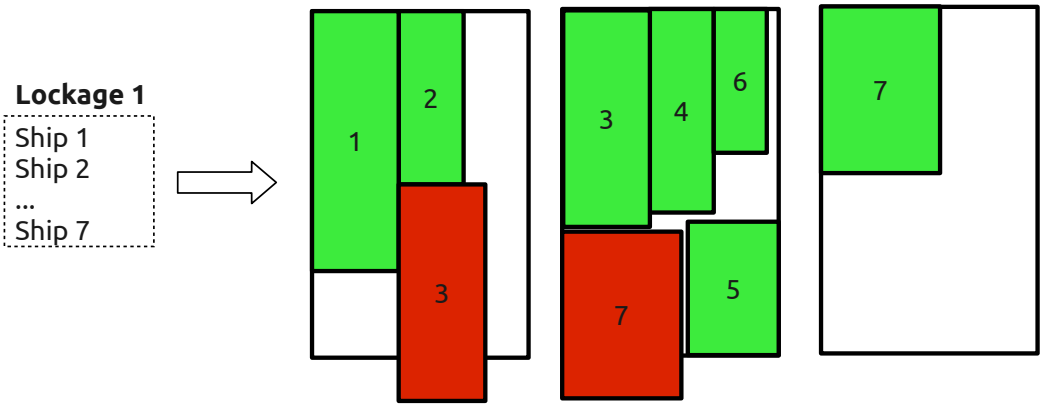

Figure 2: An example where the MP result puts ships 1 through 7 in a single lockage, and the feasible (after removing the red ships) first-come-first-served based lockages.

No-good cuts can easily be computed by solving one single lockage ship placement problem for each MP lockage. For the example from Figure 2, the

following weak cut is generated:

$$\sum_{i=1}^{7} f_{ik} \leq 6, \quad \forall k \tag{21}$$

Minimal infeasible subsets (MIS) can be found by applying the following constructive procedure. First, for a given set of ships, all subsets of size n are calculated, where n is initially set to 2. The sub problem (Section 4.2) is solved for each of these subsets, and a feasibility cut is generated when necessary. Next, all subsets of size $n+1$ are generated and compared against the infeasible subsets generated in the previous iterations. Every subset of size $n+1$ that is a superset of a MIS generated in a previous iteration is discarded. The sub problems of the remaining sets N' of size $n+1$ are solved and cuts are generated where applicable. The constructive procedure terminates whenever no new cuts can be identified (i.e. all generated subsets of size n are infeasible). Note that the larger the number of ships that can be transferred in a single lockage, the more computationally expensive this procedure becomes. Cuts produced by this procedure will be referred to as 'subset cuts'. The thirteen subset cuts that are generated for the given example are presented in Figure 3.

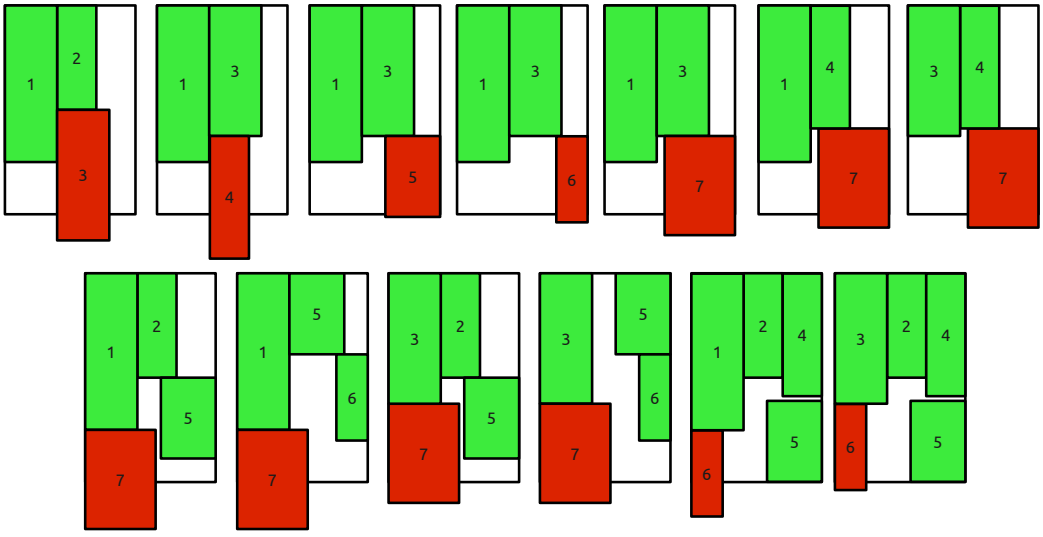

Figure 3: Visualization of the thirteen subset cuts for the example in Figure 2.

An alternative means to generating minimal infeasible subsets of ships utilizes a strict ordering on the ships. Let $N' = \{1, 2, \ldots, n\}$ be an ordered

set of ships, based on their arrival time. Start by setting $S = \{1\}$. Iteratively add ships from the head of N' to S until S becomes an infeasible subset of ships. This is an 'order cut'. When generating *feasibility* cuts, all ships in S are removed from N', except the last ship added to S, and the procedure is repeated until N' is exhausted. When generating *initial* cuts, only the first ship in S is removed from N' before the procedure is repeated. Note that these cuts are particularly effective under a FCFS lockage policy. When considering the example in Figure 2, two feasibility order cuts can be generated:

$$\sum_{i=1}^{3} f_{ik} \leq 2, \quad \forall k \tag{22}$$

$$\sum_{i=3}^{7} f_{ik} \leq 4, \quad \forall k \tag{23}$$

An efficient approach to identifying *small* infeasible subsets of ships is based on surface calculations: any set of ships having a combined surface that exceeds the total surface of the lock chamber is infeasible. Whenever surface calculations are used to identify infeasible subsets, it will be denoted as follows: 'subsurf' (subset based) and 'surf' (order based). Surface calculations provide a non-tight upper bound on the number of ships that can be placed, and therefore they can be applied as initial cuts only. Indeed, applying them as feasibility cuts does not guarantee that the MP will converge towards a feasible solution, as surface based cuts may be unable to cut away some infeasible parts of the search space.

5. Experiments

To assess the quality of the combinatorial Bender's approach, a number of experiments have been conducted on instances based on real-world data originating from the Albertkanaal in Belgium (Verstichel, 2013). A problem instance consists of two parts:

1. traffic data:
    - exact arrival times of the ships
    - the directions of the ships
    - the dimensions of the ships; safety distances have been accounted for in the reported dimensions.

2. lock data: the characteristics of the locks.

This setup enables experimenting with variations of lock configurations and traffic data.
Table 2 presents a summary of the ship data used in the instances. The dimensions of the ships and locks are extracted from traffic reports on the Albertkanaal in 2008; the actual arrival times of the ships, however, were not recorded. Hence, these arrival times were simulated using a random generator. The ship inter arrival times have been selected from a uniform distribution between 0 and 2σ (Table 2). Finally, the fraction of ships arriving upstream or downstream is either symmetric (50% upstream traffic, 50% downstream), or asymmetric (70% upstream traffic, 30% downstream).

Inter arrival time distribution	Uniform (R)
Average inter arrival time σ	1,2,3,4,5,10,15,30 minutes
Number of ships	10, 20, 30, 40, 50, 60, 70, 80, 90
Upstream/Downstream fraction	50/50, 30/70
Ship sizes	from 4.25m x 16.27m to 10.50m x 110m

Table 2: Traffic data.

Table 3 presents the lock data. We consider five possible lock configurations: a lock with a single small chamber (SSC), a single large chamber (SLC), two parallel small chambers (PSC), two parallel large chambers (PLC), and a multi chamber type lock (MCT) consisting of two small chambers and a single large chamber. Note that the latter lock configuration is identical to the real locks on the Albertkanaal. All ship and lock data are available online (Verstichel, 2013).

The experiments have been performed on a Dell Optiplex 790 with an Intel(R) Core(TM) i7-2600 (3.40GHz) and 8GB of memory running a 64-bit Linux Mint. The mathematical models were solved using Gurobi 5.1 under an academic license, with a time limit of 12 hours.

	Width (m)	Length (m)	Lockage duration p (min)
Small chamber	16.0	136.0	16
Large chamber	24.0	200.0	16

Table 3: Attributes of the chambers.

The complexity of each instance largely depends on the lock configuration. Therefore, the discussion of the computational results is structured with respect to the different lock configurations. The performance of the different feasibility cuts and initial cuts is evaluated for each group of instances. A comparison of the Benders' procedure and the monolithic approach from Verstichel et al. (2014a) is presented.

5.1. FCFS single chamber lock

The first series of experiments is performed on a single chamber lock with a first-come-first served policy for the ships. Both a single small chamber (SSC) and a single large chamber (SLC) are considered. The experiments assess the performance of the feasibility cuts and the effect of adding initial cuts to the MP. The results are depicted in Figure 4 and 5. The x-axis of each figure displays the different instances, which are ordered, from left to right, based on (1) increasing number of ships (2) increasing inter arrival time and (3) traffic ratio (first 70/30, then 50/50). The numbers underneath the axis are formatted as I_S, with I denoting the inter arrival time, and S the number of ships. Computation times in seconds are shown on the y-axis of each figure (logarithmic scale). Note that these computation times include generation of both the feasibility and initial cuts, unless stated otherwise.

The most basic version of the Benders' procedure relies on no-good cuts only and is referred to as 'no-good' in the graphs. A stronger version is obtained by replacing the no-good cuts by order cuts (Figure 4 (a)). Especially for the larger instances, order cuts contribute to a significant decrease in computation time. Another approach is to generate a number of initial cuts and add them to the initial MP. Figure 4 (b) reveals a drastic reduction of computation times under the presence of such initial cuts. Here, applying no-good feasibility cuts without initial cuts (no init) is compared with combining no-good cuts with initial surface cuts (surf init) or order cuts (order init). Interesting results are found when considering the number of MP-SP iterations and the total number of cuts added to the MP in Figure 4 (c) and (d). The number of iterations for the simple no-good feasibility cuts increases strongly with instance size, while more advanced cut generation methods (e.g. order) scale much better with instance size. Note that when applying the initial order cuts, no MP-SP iterations are required, as all necessary cuts are added to the MP beforehand. When considering the number of cuts added, the performance difference between the cut generation methods is even larger. The number of cuts generated when applying no-good

cuts (with and without initial surface cuts) varies between 0 and 2.5 times the instance size, while the initial order cut method (order init) consistently generates slightly less cuts than the number of ships in the instance. The order feasibility cuts generally require less cuts to be added to the MP than the initial order cuts. Therefore, the large number of required iterations negatively influences the overall computation time.

The computation time difference between the cut generation methods becomes much smaller when considering the SLC setting (Figures 5 (a) and (b)). Contrary to the SSC results, the average inter arrival time also seems to influence the computation time: for the small instances (< 30 ships) the computation time decreases when the average inter arrival time increases, while the opposite trend shows for the large instances (≥ 40 ships). Figure 5 (c), (d) considers the number of MP-SP iterations and cuts added. A slightly larger variation in the number of required iterations and cuts for equal instance size can be observed compared to the small chamber settings. The difference between generated cuts for the initial order cuts and the order feasibility cuts is also more significant. Nevertheless, the initial order cuts maintain the lead with respect to total computation time.

Figure 4 (e) plots the results obtained using the Benders' approach in combination with initial order cuts against the monolithic procedure presented in Verstichel et al. (2014a) for a large number of instances. Clearly, the former method outperforms the latter. Especially for several of the larger instances, computation times are reduced by 95%. The graph also shows that the total time spent in the sub problem (in this case generating the initial cuts) is very small compared to the total computation time, especially for the larger instances (order init SP). This is largely due to the limited size of the ship placement instances that have to be solved in the sub problem given this small chamber setting. The remaining approaches shown in Figure 4 (a,b) could not outperform the monolithic procedure and were therefore omitted from the graph.

The differences between the monolithic procedure and the Benders' decomposition approach become more profound in the SLC setting (Figure 5 (e)). The largest difference in computation time is observed for the instance with 20 ships, $\sigma = 2$ and symmetric traffic: the monolithic approach timed out after 12 hours, whereas the Benders' procedure with initial order cuts attested optimality in only 1.3 seconds. The maximum computation time of the Benders' approach for instances with up to 40 ships is 12 minutes, while the monolithic procedure fails to solve 16 out of 64 instances within 12 hours.

For the instances with 50 and 60 ships, the monolithic approach could only solve a single instance to optimality and failed to produce a feasible solution for 7 out of 32 instances. The decomposition approach on the other hand generates feasible solutions for all instances, and attests optimality in 24 cases. Finally, for the 48 instances consisting of 70 to 90 ships, the decomposition method solves 21 instances to optimality while for the remaining instances, feasible solutions were found. When considering the time spent in the sub problem (order init SP) for the instances with up to 40 ships, we find that the initial order cuts method spends on average 33.35% of its time in the sub problem. The minimum and maximum time spent int he SP are 2.15% and 95.85%. This decreases to 2.29%, 0.01% and 37.84% respectively for the larger instances.

An analysis of the convergence speed of the master problem showed that optimality is attested shortly after the optimal solution is found in all but a few cases for both the single small and single large chamber setting.

5.2. No FCFS single chamber lock

The second series of experiments is conducted on the same instances, but without the FCFS policy. The results for instances with 10 and 30 ships or more were omitted, as all 10 ship instances were solved in less than 2 seconds and only a few of the large instances were solved in less than 12 hours.

Dropping the FCFS policy has several implications for the decomposition method. A number of constraints of the MP no longer apply in the absence of the FCFS policy. Consequently, the MP becomes substantially harder to solve. Furthermore, the absence of an explicit ordering of the ships permits a significantly larger number of ship-to-lockage assignments in the unrestricted MP. Consequently, adding no-good and order cuts becomes ineffective. The resulting performance decrease is reflected by the number of MP-SP iterations: the instances from Section 5.1 are solved within a few iterations, whereas the same instances in the absence of the FCFS rule require up to 3200 iterations. In the subsequent experiment, the no-good and order cuts have been replaced by the computationally more expensive subset cuts. Figures 6 (c) and (d) compare the performance of various methods. For the small chamber lock, initial subset cuts appear to be the best approach. For the large chamber, a combination of initial order cuts and feasibility subset cuts works best. In either case, the Benders' approaches outperform the monolithic approach to a large extent. As expected, the number of generated cuts increased after removing the FCFS restrictions on the master problem

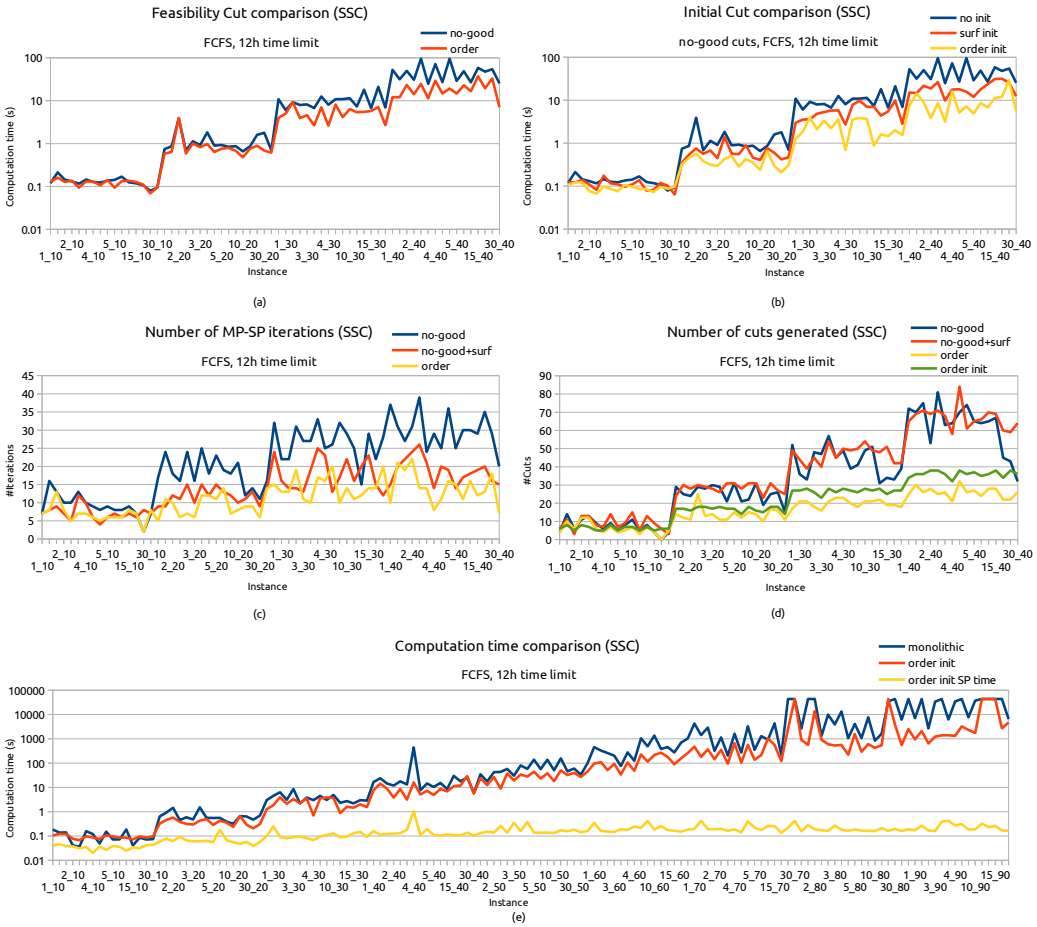

Figure 4: Comparison of the computation time of the different cut generation methods for a single small chamber lock, under a FCFS policy.

(Figure 6 (a) and (b)). When excluding one outlying instance ($\sigma = 2$ and unbalanced traffic), the initial subset cuts (init subship) generated on average 3.2 cuts per ship for the small chamber setting, while the feasibility subset cuts combined with initial order cuts (subship+order) generated 5.0 cuts per ship. For the outlying instance, these values are 8.2 and 20.5 respectively. Both methods spent no more than one second in the sub problem on all but the outlying instance, where the subship+order method spent 200 seconds (or 21% of the total computation time) in the sub problem. The optimal solution (init subship no proof) was on average reached after 33%

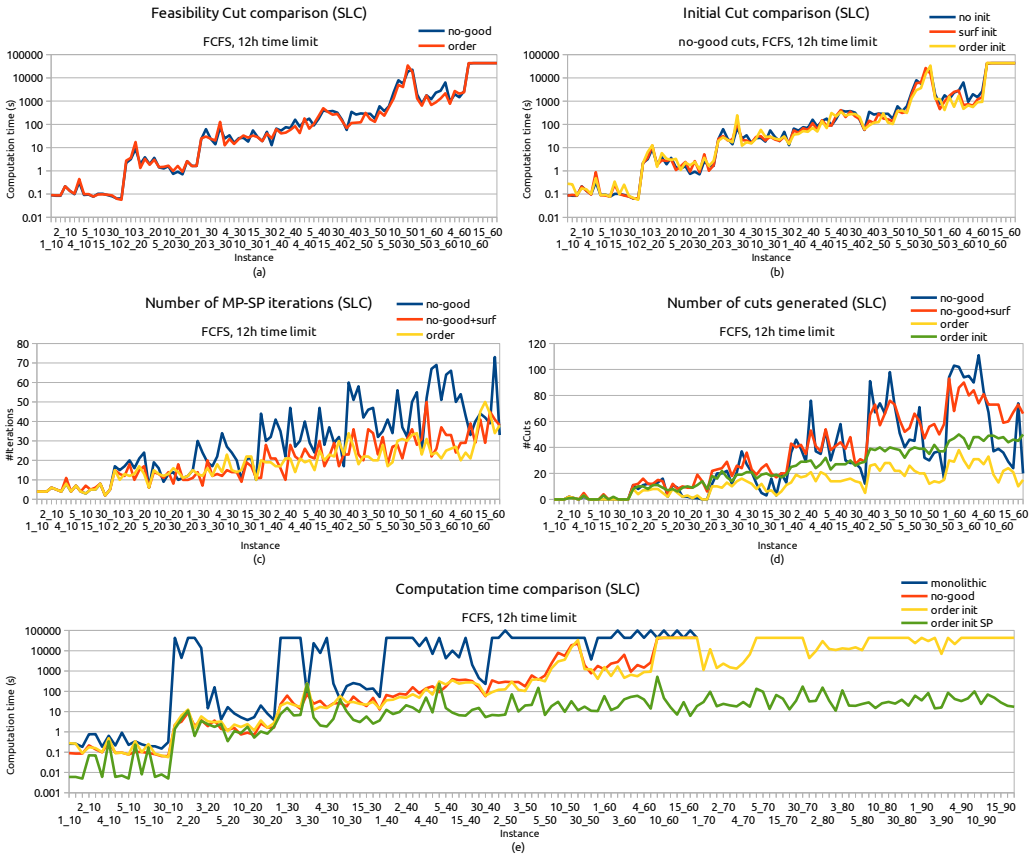

Figure 5: Comparison of the computation time of the different cut generation methods for a single large chamber lock, under a FCFS policy.

of the total computation time with a minimum of 0.5% and a maximum of 77.6%. The difference with the FCFS case is significant. FCFS required very little time to attest optimality once the optimal solution was found. On the single large chamber setting, there is a large variation in the number of cuts generated by the initial subset cuts method (init subship on Figure 6 (b)). For the instance with $\sigma = 30$ and unbalanced traffic, for example, over 2200 cuts were generated, taking up 53% of the total computation time of 257 seconds. The subship+order cut method on the other hand needed 27 cuts for the same instance, while computing only 12 seconds. The subship+order and init subship methods spent respectively 40 minutes and 65 minutes in

the sub problem for the $\sigma = 2$ (unbalanced traffic) instance, with more than 10 times the maximum over the other instances. Attesting optimality after the optimal solution was found (subship+order no proof) took between 5% and 97% of the total computation time, with an average of 42%.

From the above results, it is apparent that the absence of the FCFS rule has a significant impact on the computation times. We therefore adopt the FCFS policy for the remaining experiments.

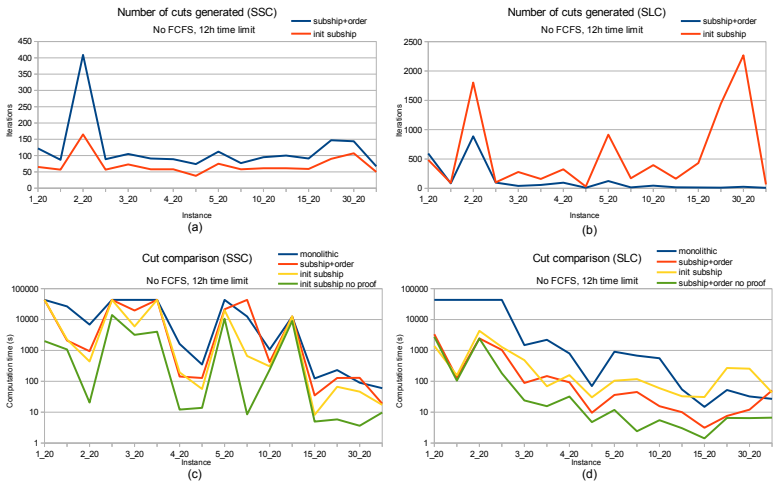

Figure 6: Comparison of the computation time of different approaches for single chamber locks without FCFS.

5.3. FCFS parallel identical chamber lock

For the identical parallel chamber instances, the results for instances with 10 and ≥ 30 ships were omitted. The difference between the initial cuts is limited for a lock with two small parallel chambers (Figure 7 (a)). Applying no initial cuts (subship) results in the best overall performance. Figure 7 (b) shows that the feasibility subset cuts is slower than the monolithic approach

when inter arrival times are large. The decomposition method is almost always faster for the other instances, with a total computation time of 6.5 for the monolithic approach and only 2 hours for the feasibility subset cuts. The difference in computation time between finding and attesting the optimal solution seems to decrease with increasing inter arrival time. No significant differences were found with respect to the time spent in the sub problem (always less than 1.4 seconds) or the number of cuts/iterations.

For a large parallel chamber lock, the initial order cuts show the best overall performance with a total computation time over all instances of 40 minutes, closely followed by the initial surface cuts (55 minutes) (Figure 8 (a)). All other initial cuts are significantly slower than the aforementioned initial cuts. Some interesting results can be observed when considering the number of cuts generated for each method. The subship+order and subship+surf methods clearly have the upper hand with an average of 74 (subship+order) and 50 (subship+surf) cuts generated per instance. Applying one of the other methods (no initial cuts, subset based surface initial cuts or subset based ship placement initial cuts) results in an average of $570 - 1007$ cuts per instace. The subship+order method spends on average 14.4% of the total computation time in the sub problem, with the instance with $\sigma = 2$ and unbalanced traffic again providing an outlyer where 91% of the time was spent in the sub problem. When comparing with the monolithic approach (Figure 8 (b)), we find that the decomposition method is much faster when the ship inter arrival time is short. For instances with medium inter arrival time ($\sim$ 10 minutes) no clear winner can be determined, while the monolithic approach is the best choice when facing large inter arrival times. Finally, all cut generation methods were able to attest optimality on one instance more than the monolithic approach.

5.4. FCFS multi chamber type lock

The results for the multi chamber type lock are summarized in Figure 9. Here only the ≥ 30 ship instances were omitted. Similar to the SSC results it appears that, aside from the number of ships, the ship inter arrival time has the largest influence on the required computation time. Applying feasibility subset cuts combined with initial subset based surface cuts (subship+subsurf) appears to be the best way of solving LSP for this multi chamber type setting. It is the fastest approach on all but a few instances and it is able to attest optimality for 31 out of 32 instances, while the monolithic approach fails to do so for 10 instances. This result is noteworthy, as

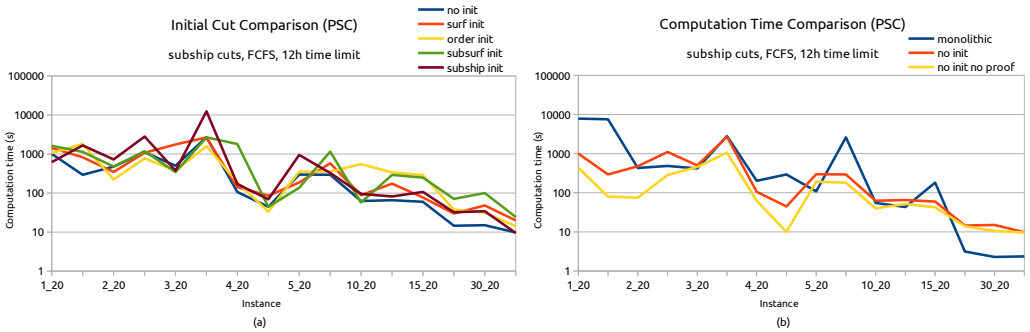

Figure 7: Comparison of the computation time of different cut generation methods for a small parallel chamber lock under a FCFS policy.

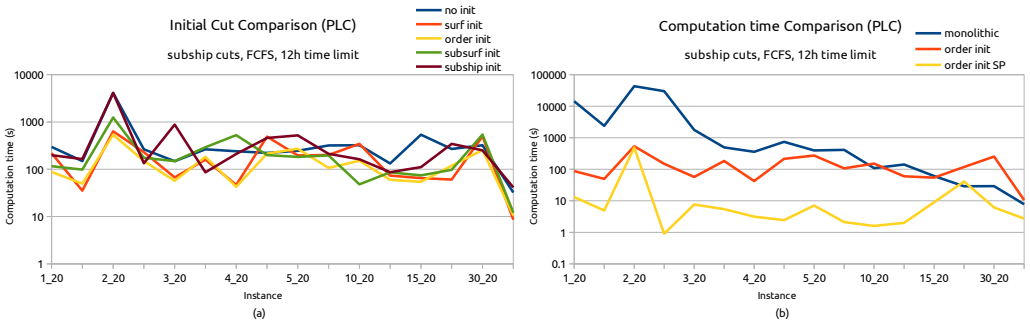

Figure 8: Comparison of the computation time of different cut generation methods for a large parallel chamber lock under a FCFS policy.

the subship+subsurf method produces significantly more cuts than the other approaches (Figure 9 (c)).

5.5. Heuristic sub problem approach

The last experiment considers the effects of applying the multi-order best-fit heuristic for the ship placement problem (Verstichel et al., 2014b) to the SP. This enables an investigation of the quality of the packing heuristic from a scheduling point of view. As no more than a few (milli)seconds will be spent in the sub problem, this might be especially interesting for some of the large chamber instances where a lot of time was spent in the exact sub problem algorithm. The results of this heuristic approach under a time limit of 12 hours are summarized in Table 4. All heuristic results are compared

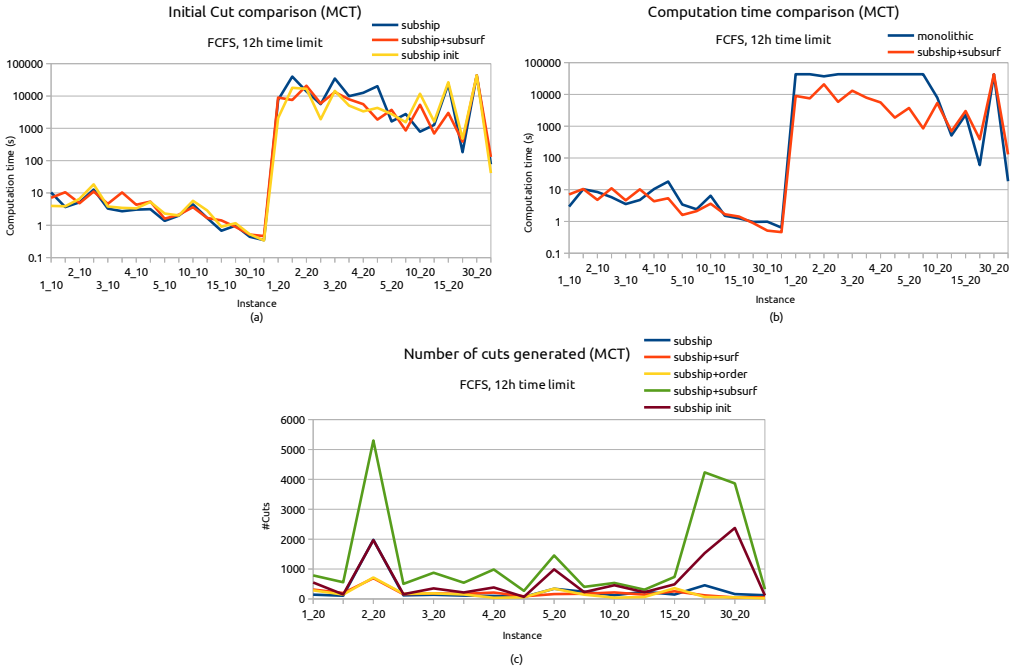

Figure 9: Comparison of the computation time and number of cuts generated for different cut separation methods for the multi-chamber type lock under a FCFS policy.

to those of the initial order cuts decomposition approach from Section 5.1, which are referred to as the exact results. The gap is computed as follows: $Gap\% = (heuristic - exact)/exact$.

When applying initial order cuts to the single small chamber lock, all instances with less than 70 ships were solved to optimality in less than 5 minutes. For the larger cases the heuristic approach matched the exact results on all but one instance, where a gap of 0.03% remained after 12 hours of computation time. The gap was however closed after increasing the computation time for this instance. For the single large chamber lock, the average gap for the heuristic approach on the instances with less than 60 ships was 0.10%, with a maximum of 1.19%. As all these istances were solved within the time limit, this gap can be fully attributed to the application of the multi-order best-fit heuristic. When considering the large instances, an average gap of 0.35% is observed, with a maximum of 2.93% and a minimum of -0.35%. Indeed, the heuristic approach improved on the exact results in

15 cases, denoted by a negative gap. Closer analysis of the results showed that the exact approach had reached the time limit on these instances, and that increasing the maximum computation time enabled the exact method to match or improve the heuristic results. The heuristic method failed to match the exact results on 34 instances, reaching the time limit in 12 cases. By increasing the computation time limit for these 12 instances, we were able to determine that in 3 cases the exact results could be matched by the heuristic approach. The gap remained for the other 9 instances, showing that the multi-order best-fit heuristic was unable to produce the needed exact ship placement solutions. One interesting result is observed when analysing the total time spent in the SP. While the exact method spent up to 527 seconds computing solutions for the sub problem, the heuristic always produced the results in less than 0.5 seconds.

All PSC instances were solved to optimality by the heuristic decomposition method when applying initial subset cuts. For the PLC setting, the multi-order best-fit heuristic prevented the decomposition method from constructing the optimal solution, resulting in a small optimality gap of 0.04% and 0.12%. Similar results are obtained for the multi chamber type lock, where optimality could not be obtained in four cases, leading to a maximum gap of 0.45%.

These results show that applying a high-performance heuristic for solving the sub problem has little or no impact on solution quality, but drastically reduces the computational effort.

5.6. Summary of the experiments

The experimental results show that the proposed Combinatorial Benders' decomposition is very effective for the lock scheduling problem. This is especially the case for instances with a complex packing aspect, i.e. where several ships can be transferred in a single lockage operation, and (very) large instances. Indeed, for the single large chamber experiments, we find the largest decrease in computation time for the instances with short inter arrival times, with computation time differences of several orders of magnitude on several instances. Furthermore, the decomposition approach was able to produce feasible solutions, and often attest optimality, on a large number of instances that could not be tackled by the existing 'monolithic' approach. The same advantages are seen when applying a heuristic method to the sub problem. The majority of the instances are still solved to optimality, while the heuristic never induced a gap of more than 2.93% on the other instances. The

time spent in the sub problem is strongly reduced on all instances, leaving more time for the master problem, whose convergence speed appears to be the limiting factor of the presented decomposition approach.

6. Conclusion

An exact Combinatorial Benders' decomposition to the lock scheduling problem was proposed. The LSP, encompassing a scheduling, assignment and packing problem, is decomposed into a master problem and a sub problem. The master problem handles resp. the scheduling and assignment problems, whereas the packing problem is dealt with in the sub problem. When compared to Verstichel et al. (2014a), where LSP is solved as single integrated 'monolithic' MIP problem, the Benders' decomposition has two fundamental advantages. The monolithic model links the variables involving the assignment, packing and scheduling constraints through a number of inefficient big-M constraints. Due to the decomposition, many of these constraints are no longer required, as they are enforced through the addition of cuts to the master problem. Another key advantage is that each packing sub problem can be solved in parallel by efficient dedicated algorithms, such as the ones presented by Verstichel et al. (2014b).
The effectiveness of the proposed Combinatorial Benders' approach is illustrated on a large number of problem instances. Optimal solutions are discovered for instances with up to 90 ships at a single large chamber lock, whereas the monolithic approach failed to find feasible solutions on instances with 50 ships within a time limit of 12 hours. Furthermore, for most instances, a significant decrease in computation time is observed. Especially instances having ships that can be transferred simultaneously in a single lockage operation benefit from the new approach, as is shown in the experiments where the ship inter arrival times are short. Finally, applying a heuristic to the sub problem instead of an exact algorithm results in high quality solutions on all instances, with a maximal optimality gap of 2.93%, while the maximal time spent in the sub problem is reduced from 527 to 0.5 seconds.

Future work may be aimed at improving the MP's procedure, which is currently the main bottleneck of the decomposition approach. For example, the convergence rate of the MP could be improved by adding a number of valid inequalities. Alternatively, one could extend this work by also taking a heuristic method for the MP into consideration.

Acknowledgement

Research funded by Ph.D. grants SB091152 and SB093152 of the Institute for the Promotion of Innovation through Science and Technology in Flanders (IWT-Vlaanderen).

We would like to thank the Scheepvaartmanagement of the Port of Antwerp for sharing their experience and real-life data on the ship placement problem. The real-life data provided by IT-Bizz and nv De Scheepvaart was also greatly appreciated.

References

Bai, L., Rubin, P.A., 2009. Combinatorial Benders Cuts for the Minimum Tollbooth Problem. Operations Research 57, 1510–1522.

Benders, J., 1962. Partitioning procedures for solving mixed-variables programming problems. Numerische Mathematik 4, 238 – 252.

Codato, G., Fischetti, M., 2006. Combinatorial benders' cuts for mixed-integer linear programming. Operations Research 54, 756–766.

Côté, J.F., Dell'Amico, M., Iori, M., 2013. Combinatorial Benders' Cuts for the Strip Packing Problem. Technical Report CIRRELT-2013-27. Interuniversity Research Centre on Enterprise Networks, Logistics and Transportation and Department of Computer Science and Operations Research, Université de Mon (CIRELT).

Geoffrion, A., 1972. Generalized benders decomposition. Journal of Optimization Theory and Applications 10, 237 – 260.

Hooker, J., Ottoson, G., 2003. Logic-based benders decomposition. Mathematical Programming 96, 33 – 60.

nv De Scheepvaart, 2012. Annual report 2012 (in Dutch). `http://www.descheepvaart.be/uploads/scheepvaart/FILE_1DAC7D33-98D9-4971-978F-12C499037150.PDF`.

Port of Antwerp, 2012. Annual report 2012. `http://www.portofantwerp.com/en/annual-report-2012`.

Rasmussen, R., Trick, M., 2007. A benders approach for the constrained minimum break problem. European Journal of Operational Research 177, 198–213.

Tran, T.T., Beck, J.C., 2012. Logic-based benders decomposition for alternative resource scheduling with sequence dependent setups., in: ECAI, IOS Press. pp. 774–779.

Verstichel, J., 2013. Project web page of lock scheduling with parallel chambers. `http://allserv.kahosl.be/~jannes/lockplanning`.

Verstichel, J., De Causmaecker, P., Spieksma, F.C., Vanden Berghe, G., 2014. The generalized lock scheduling problem: An exact approach. Transportation Research Part E: Logistics and Transportation Review 65, 16–34.

Verstichel, J., De Causmaecker, P., Spieksma, F.C., Vanden Berghe, G., 2014. Exact and heuristic methods for placing ships in locks. European Journal of Operational Research 235, 387–398.

Verstichel, J., Vanden Berghe, G., 2009. A late acceptance algorithm for the lock scheduling problem. Logistik Management , 457 – 478.

Lock	# ships	Total	Feasible	Exact	Gap
SSC	10-60	96	96	96	0.000%
	70-90	48	48	47	0.001%
SLC	10-50	80	80	62	0.105%
	60-90	64	64	30*	0.354%
PSC	10-20	32	32	32	0.000%
PLC	10-20	32	32	30	0.005%
MCT	10-20	32	32	28	0.029%

Table 4: Summary of the heuristic experiments. '# ships' denotes the instance size range for the row and 'Total' the total number of instances in this range. The number of feasible solutions generated by the heuristic decomposition approach is added under 'Feasible'. 'Exact' shows the number of instances for which the exact solution was matched by the heuristic approach, while 'Gap' shows the average gap between the exact and heuristic solutions. *The heuristic decomposition outperformed the exact approach for 15 instances.