A Combinatorial Benders Decomposition for the Lock Scheduling Problem

J. Verstichel\textsuperscript{a,c,*}, J. Kinable\textsuperscript{a,d}, P. De Causmaecker\textsuperscript{b,c}, G. Vanden Berghe\textsuperscript{a}

\textsuperscript{a}CODeS, KAHO Sint-Lieven, Gebroeders De Smetstraat 1, 9000 Gent, Belgium
\textsuperscript{b}CODeS, Department of Computer Science, KU Leuven, ITEC-iMinds
\textsuperscript{c}KU Leuven campus Kortrijk, Etienne Sabbelaan 53, 8500 Kortrijk, Belgium
\textsuperscript{d}ORSTAT, KU Leuven, Naamsestraat 69, 3000 Leuven, Belgium

Abstract

This paper presents an exact algorithm for the Lock Scheduling Problem (LSP) based on a Combinatorial Benders decomposition.

LSP consists of three strongly interconnected sub problems: scheduling the lockages, assigning ships to chambers, and positioning the ships inside the chambers. These three sub problems are interpreted resp. 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 which can be solved efficiently using a Combinatorial Benders approach. First the master problem is solved, thereby sequencing the ships into a number of lockages. Next, for each lockage, a packing sub problem is solved to verify its feasibility, possibly returning a number of combinatorial inequalities (cuts) to the master problem.

Experiments are conducted on generated real world instances. The results indicate that our decomposition approach significantly outperforms other exact approaches previously presented in literature.

Keywords: Lock Scheduling Problem, Combinatorial Benders’ Decomposition

*E-mail: jannes.verstichel@kahosl.be, Telephone: +329/265.87.04
Email address: jannes.verstichel@kahosl.be (J. Verstichel)


1. Introduction

The port of Antwerp (Belgium), one of the largest harbors in Europe, processed more than 180 MTE (Million Tonnes Equivalent) of cargo and 70000 ships in 2012, for an average of almost 200 ships a day (Port of Antwerp, 2012). It is a major hub for both inland as well as intercontinental cargo traffic. The harbor is situated at a large tidal river, having tidal differences averaging five meters. To ensure a persistent water level at the harbor’s docks, the docks are separated from the main water way by a number of locks. These locks present us a complex optimization problem: the vast number of ships that enter and leave the harbor on a daily basis 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 valuable waiting time of ships, and make the port even more economically attractive.

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 a total of over 37 MTE of processed cargo 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 optimize the lock operations by reducing the number of lockage operations (i.e. water usage) and the waiting time of ships.

From a wider perspective, the Lock Scheduling Problem (LSP) encompasses three, interconnected sub problems: an assignment, a packing, and a scheduling problem. LSP was first introduced in an inland setting by Verstichel and Vanden Berghe (2009), where the authors proposed a heuristic approach. Although this heuristic approach is capable of efficiently solving large instances, it does not provide any insights as to the quality of the solutions. Later, in Verstichel et al. (2013b) a mathematical model for the generalized LSP applicable to both inland and port settings was presented, along with an exact branch-and-bound procedure. Due to the complex nature of LSP, only relatively small instances were solved to optimality using this ‘combined’ approach. In this work a new, fast exact approach is presented 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 the 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 (and feasible) schedule is obtained.

The main focus of this paper goes to the decomposition approach and its application to LSP, thereby omitting detailed discussions on the exact sub problems as they exhibit a large number of application specific constraints. First, in Section 2, the LSP is described in more detail. Next, Section 3 provides a literature review. Section 4 presents the Benders’ decomposition approach, thereby defining the master and sub problem in more 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 real-world instances based on data obtained at several locks in Belgium. The results are presented in Section 5. Finally, 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 in detail is described in Verstichel et al. (2013b); here we only sketch the general outlines.

2.1. Scheduling, Assignment and Packing

A lock consist 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 resp. \( N^1 \), \( N^2 \), \( N^1 \cap N^2 = \emptyset \), \( N^1 \cup N^2 = N \). For each ship \( i \in N \), an arrival time \( r_i \) is known, as well as the dimensions of the ship. The ships are grouped into a number of batches, 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 $p_i$ of the ships $i \in N$ that are assigned to the lockage, plus a constant amount $p_t$ depending on the type $t \in T$ of chamber that performs the lockage.

By nature, a lock chamber is always in one out of two possible states. Depending on its state, the chamber can handle a downstream or an upstream transfer. Each transfer switches the lock’s state. Consequently, to perform two upstream (or downstream) lockage operations consecutively on the same chamber, an additional empty lockage needs to be performed in between the two lockages to switch the chamber’s state. Put formally, for 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. $s_{lk} = p_t$ if $k, l$ are both upstream (or downstream) lockages, $s_{lk} = 0$ otherwise. Finally, in lock scheduling, a first come first served (FCFS) policy is often enforced by the lock authorities, meaning that for two ships $i, j \in N : r_i < r_j$, the lockage containing ship $i$ must be finished no later than the lockage of ship $j$, i.e. $c_i \leq c_j$. Here, it is assumed that no two ships ever arrive at the lock at the exact same time.

To assign ships to a specific lockage operation, a number of constraints need to be taken into consideration. Obviously, the ships assigned to a single lockage $k \in M_t$, may not surpass the capacity of a chamber of type $t \in T$. Moreover, for each of these ships, its exact location 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. (2013a).

In essence, $LSP$ is interpretable as a traditional machine scheduling problem with sequence dependent setup times: a set of tasks (ships) are grouped into jobs (lockages) which are then assigned to machines (lock chambers). An overview of the parameters used in this paper is provided in Table 1.
Parameters:

- $N, N^1, N^2$ is the set of ships, subdivided in upstream ships $N^1$ and downstream ships $N^2$.
- $T$ is the set of different chamber types.
- $M = M^1 \cup M^2$ is the set of lockages, thereby distinguishing between upstream $M^1$ and downstream $M^2$ lockages.
- $M_t = M^1_t \cup M^2_t$ 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. $M_t = \{1, 2, \ldots, m^1_t, m^1_t+1, \ldots, m^1_t+m^2_t\}$, where $m^i_t$, $i = 1, 2$, are bounds on the number of upstream resp. downstream lockages for chamber type $t \in T$.
- $U_t$ is the set of chambers of type $t \in T$.
- $p_t, p_i$ are the fixed processing time of a chamber of type $t \in T$, and the processing time of ship $i \in N$.
- $r_i$ is the time at which ship $i \in N$ arrives at the lock.
- $s_{lk}$ is the transition time required between two consecutive lockages $k, l \in M_t, t \in T$ performed on the same chamber. $s_{lk} = p_t$ 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)

Table 1:

3. Literature review

Benders’ decomposition is an efficient and popular exact decomposition method introduced by Benders (1962). It is applicable to problems that can be decomposed into a general Restricted Master Problem (MP) and a linear programming Sub Problem (SP). By temporarily fixing the variable values in the MP and solving (the dual of) the remaining SP cuts can be derived that guide the MP towards optimality. Benders’ approach was generalized to a broader class of problems for which the SP can be solved by means of dual-adequate algorithm (i.e. an algorithm that provides both the primal and dual solution to the problem) by Geoffrion (1972). Hooker and Otto-son (2003) extends the application of Benders’ decomposition even further
through the introduction of inference duals and a logic-based Benders’ decomposition method. A downside this approach is that the cuts may contain a large number of disjunctions, which reduces its applicability in the case of a (mixed integer) linear programming master problem. Saharidis et al. (2010) reduce the number of iterations required by Benders’ method through the generation of a bundle of low-density (i.e. containing only a small number of MP decision variables) cuts instead of a single low-density cut. This bundle of cuts is generated in such a way that it covers a certain percentage of the MP variables, cutting away a larger part of the MP solution space in a single iteration. Hooker (2007) and Rasmussen and Trick (2007) combine mixed-integer linear programming and constraint programming in hybrid Benders’ decomposition frameworks and substantially improve on the state of the art in their respective application domains. Chu and Xia (2004) present Benders’ cuts for class of problems where the objective contains only the MP variables. At each iteration \( k \), the restricted master problem \( MP^{(k)} \) is solved to optimality, and the corresponding sub problem \( SP(y^{(k)}) \) is constructed. If this sub problem is feasible, optimality is reached and the procedure exits. Otherwise, a minimally relaxed integer Benders’ cut is generated by adding slack variables \( r \) to the sub problem (rendering it feasible) and solving the corresponding dual sub problem to optimality. Codato and Fischetti (2006) introduce a similar solution approach for mixed-integer programming (MIP) problems involving logical implications (big M constraints), but instead of generating cuts through solving the dual of the linear programming sub problem they rely on the generation of minimal infeasible subsystems, distilling combinatorial information (i.e. logical implications) from the original model and adding them to the master problem. By applying this cut generation method at each node in a branch and cut scheme for the MP, the aforementioned logical implications are only added on-the-fly and do not deteriorate the linear relaxation of the master problem like they would in the original MIP model.

4. A Combinatorial Benders’ Decomposition

In contrast to (Verstichel et al., 2013b) where the authors attempted to solve the LSP via a single, large, Mixed Integer Linear Programming problem, we propose to solve LSP via a decomposition approach. The decomposition results in a master problem and a sub problem, each of which have to be solved iteratively. 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 these problems, whereas there may not exist such an algorithm capable of tackling the entire problem at once. The presented method is very similar to that of Codato and Fischetti (2006). The main differences are that we work with a integer programming sub problem instead of a linear one, and that we apply a constructive algorithm for determining minimal infeasible subsets, whereas Codato and Fischetti (2006) determined MIS through an LP.

In the decomposition, the master problem is provided with a list of ships that need to traverse the lock, the direction in which the ships want to traverse the lock, and their arrival times at the lock. The master problem first partitions the 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, for the generated subsets, the master problem proposes a schedule, thereby determining the exact starting times of the lockage operations. Subsequently, the sub problem verifies for each subset whether the ships in that set can be transferred simultaneously, i.e. whether they fit together inside the lock chamber. The latter boils down to solving a complex rectangle packing problem. 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 the same lockage operation.

The following two subsections 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. 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 to speed up the model are not included in the problem description below. For a complete model, including the omitted constraints,
we refer to (Verstichel et al., 2013b).

\[
\min \quad \lambda_1 \sum_k z_k + \lambda_2 \sum_{i \in N} c_i + \lambda_3 T_{max}
\]

subject to:

\[
\sum_{k \in M/j} f_{ik} = 1 \quad \forall i \in N^j, j = 1, 2
\]

\[
z_k \leq \sum_{i \in N} f_{ik} \quad \forall k \in M
\]

\[
z_k + 1 \leq z_{k+1} \quad \forall k \in M \quad \text{and} \quad t \in T
\]

\[
c_i \geq C_{max}(f_{ik} - 1) + C_k \quad \forall i \in N, k \in M
\]

\[
P_k \geq p_z z_k + \sum_{i \in N} p_i f_{ik} \quad \forall k \in M \quad \text{and} \quad t \in T
\]

\[
\sum_{u \in U, t} \text{proc}_{ku} = z_k \quad \forall k \in M \quad \text{and} \quad t \in T
\]

\[
\text{proc}_{ku} + \sum_{v \in U, v \neq u} \text{proc}_{lv} + \text{seq}_{kl} \leq 2 \quad \forall k, l \in M, l > k, \text{ and} \quad t \in T
\]

\[
C_k - C_l + 2C_{max}(3 - \text{seq}_{kl} - \text{proc}_{ku} - \text{proc}_{lu}) \geq P_l + s_{kl} \quad \forall k \in M, l \in M, l > k, \text{ and} \quad t \in T
\]

\[
C_l - C_k + 2C_{max}(2 + \text{seq}_{kl} - \text{proc}_{ku} - \text{proc}_{lu}) \geq P_k + s_{lk} \quad \forall k \in M, l \in M, l > k, \text{ and} \quad t \in T
\]

\[
C_k \geq r_i f_{ik} + P_k \quad \forall i \in N \quad \text{and} \quad k \in M
\]

\[
T_{max} \geq c_i - r_i \quad \forall i \in N
\]

The master problem uses four sets of variables: 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 \(k \in M\) and finally integer variables \(C_k, k \in M\) denoting the completion time of lockage \(k \in M\).

The objective, equation (1), minimizes (a) the number of lockages, (b) the time that a ship leaves 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 \(\lambda_2 = \lambda_3 = 1.0\). Constraints (2)-(4) assign ships to lockage operations. Constraints (2) ensure that each ship is assigned to a lockage operation. 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)): a lockage \(z_{k+1}\) cannot be used if \(z_k\) is unused, \(k \in M, t \in T\). The remaining constraints take care of
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 scheduled on the same physical chamber, then one must precede the other
(Constraints (8)). Constraints (9), (10) perform the actual scheduling of the
lockages per chamber (see (Verstichel et al., 2013b) for a detailed discussion
of 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 each lockage needs to be verified. For each lockage, 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 prob-
lem. Note that this version of the problem is a satisfiability problem; it has
no objective. The constants used in the model are summarized in Table 1.

\[
\begin{align*}
x_i + w_i & \leq W_t & \forall i \in \overline{N}_k \\
y_i + l_i & \leq L_t & \forall i \in \overline{N}_k \\
left_{tij} + left_{tji} + b_{ij} + b_{ji} & \geq 1 & \forall i < j, i, j \in \overline{N}_k \\
x_i + w_i & \leq x_j + W_t(1 - left_{tij}) & \forall i \neq j, i, j \in \overline{N}_k \\
y_i + l_i & \leq y_j + L_t(1 - b_{ij}) & \forall i \neq j, i, j \in \overline{N}_k \\
\text{safety constraints} & & \forall i \neq j, i, j \in \overline{N}_k \\
\text{mooring constraints} & & \forall i \neq j, i, j \in \overline{N}_k
\end{align*}
\]

In the above model, integer variables \( x_i, y_i \) model resp. the x and y coor-
dinates 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, for clarity, some problem specific constraints have been omitted here; for the full sub problem, we refer to (Verstichel et al., 2013a).

Algorithm 1: Combinatorial Benders Decomposition of the Lock Scheduling Problem

**Input:** Set of ships $N$, arrival times and ship dimensions, lock parameters.

**Output:** An optimal lock schedule

1. add initial cut(s) to [MP] ;
2. repeat ← true ;
3. while repeat do

   4. Solve [MP];
   5. get solution $(z_k, f_{ik}, C_k), \forall i \in N, k \in M$;
   6. repeat ← false ;
   7. for $k \in M : z_k = 1$ do

      8. Solve [SP] for $\overline{N}_k = \{i \in N : z_{ik} = 1\}$;
      9. if [SP] is infeasible then

         10. repeat ← true ;
         11. add feasibility cut(s) to [MP] ;
         12. break ;
      13. else

         14. get solution $(x_i, y_i), \forall i \in \overline{N}_k$;

4. return Optimal schedule $(f_{ik}, C_k, x_i, y_i), \forall i \in N, k \in M$

4.3. Feasibility Cuts

Each time an infeasible sub problem is encountered, one or more feasibility cuts are generated and added to the master problem, effectively preventing the master problem from assigning specific ships to the same lockage. In addition, some initial cuts can be computed beforehand to strengthen the master problem, thereby reducing the number of infeasible lockages generated.
The general form of cuts we consider is:

$$\sum_{i \in S \subseteq N} f_{ik} \leq |S| - 1 \quad \forall k \in K' \subseteq M$$  \hspace{1cm} (20)

Stated informally, this cut prevents the ships in $S \subseteq N$ to be 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 $N_k$ ships, by setting $S = N_k$ and $K' = M'$. Usually, these no-good cuts are relatively weak, especially if $|S|$ is large. Stronger cuts may be obtained for smaller subsets of ships. The strongest cuts in this category are based on Minimal Infeasible Subsets (MIS). In this context, a MIS is a subset of ships which 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 Minimal Infeasible Subsets for a given set of ships $N' \subseteq N$ is unfortunately notoriously difficult; in fact, it would require to solve the sub problem from section 4.2 for every possible subset of $N'$. The next subsection discusses several approaches to compute strong cuts, requiring far less computational effort.

4.4. Cut separation

Minimal infeasible subsets are generated by 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. For each of these sets the sub problem (section 4.2) is solved, and for each infeasible subset a cut is generated. Next, all subsets of size $n + 1$ are generated and compared against the infeasible subsets generated in the previous iterations. Every subset of size $n + 1$ which is a superset of a MIS generated in a previous iteration is discarded. For the remaining sets $N'$ of size $n + 1$, the sub problem is 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 constructive procedure will be referred to as ‘subset cuts’.

An alternative means to generate 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 the first cut. Next, remove all ships in $S$ from $N'$, except the last ship added to $S$. This procedure is repeated until $N'$ is exhausted. Cuts constructed using this procedure will be referred to as ‘order cuts’. Note that these cuts are particularly effective under a FCFS lockage policy. When generating initial order cuts, the procedure is slightly different. Instead of removing all but the last ship in $S$ from $N'$, only the first ship added to $S$ is removed.

An efficient approach for identifying small infeasible subsets of ships is based on surface calculations: any set of ships having a combined surface which exceeds the total surface of the lock chamber is clearly infeasible. Whenever surface calculations are used to identify infeasible subsets, this will be denoted as follows: subsurf (subset based) and surf (order based). As simple surface calculations provide a non-tight upper bound on the number of ships that can be placed, they are applicable as initial cuts only. Indeed, applying them as feasibility cuts does not guarantee that the MP will converge towards a feasible solution.

5. Experiments

To assess the quality of our 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 allows us to experiment with different lock configurations using the same traffic data and vice versa.

A summary of the ship data used in the instances is given in table 2. 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 known. Hence, these arrival times were simulated using a random generator.
Inter arrival time distribution: Uniform (R)
Average inter arrival time $\sigma$ (minutes): 1, 2, 3, 4, 5, 10, 15, 30
Number of ships: 10, 20, 30, 40, 50, 60, 70, 80, 90
Upstream/Downstream fraction: 50/50, 30/70
Ship sizes: between 4.25m x 16.27m and 10.50m x 110m

Table 2: Traffic data.

The ship inter arrival times have been selected from a uniform distribution between 0 and $2\sigma$. Finally, the fraction of ships arriving upstream or downstream is either symmetric (50% upstream traffic, 50% downstream), or asymmetric (70% upstream traffic, 30% downstream).

The lock data is presented in table 3. 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 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.

<table>
<thead>
<tr>
<th>Chamber Type</th>
<th>Width (m)</th>
<th>Length (m)</th>
<th>Lockage duration $p$ (min)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Small chamber</td>
<td>16.0</td>
<td>136.0</td>
<td>16</td>
</tr>
<tr>
<td>Large chamber</td>
<td>24.0</td>
<td>200.0</td>
<td>16</td>
</tr>
</tbody>
</table>

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. For each group of instances the performance of the different feasibility cuts and initial cuts is evaluated, and a comparison of the Benders’ procedure against the Branch-and-Bound (B&B) approach from Verstichel et al. (2013b) 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 effects of adding some initial cuts to the MP. The results are depicted in Figure 1 and 2. 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 the generation of both the feasibility and initial cuts.

The most basic version of the Benders procedure solely relies on no-good cuts 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 1 (a)). Especially for the larger instances, a significant decrease in computation time is observed. Another approach is to generate a number of initial cuts and add them to the initial MP. Figure 1 (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 to combining no-good cuts with initial surface cuts (surf init) or order cuts (order init). When considering the SLC setting (Figures 2 (a) and (b)), the difference between the cuts becomes much smaller. 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 ($\geq 40$ ships).

In Figure 1 (c), the results obtained using the Benders’ approach in combination with initial order cuts are plotted against the Branch-and-Bound (B&B) procedure presented in Verstichel et al. (2013b) for a large number of instances. Clearly, the former method outperforms the latter. Especially for several of the larger instances, where computation times are reduced by 95%. The remaining approaches shown in Figures 1 (a,b) could not outperform the B&B procedure and where therefore omitted from the graph.

The differences between the B&B procedure and the Benders’ decomposition approach become more profound in the SLC setting (Figure 2 (c)). The largest difference in computation time is observed for the instance with
20 ships, $\sigma = 2$ and symmetric traffic: the B&B approach timed out after 12 hours, whereas the Benders’ procedure with initial order cuts attested optimality in just 1.3 seconds. The maximum computation time for the Benders’ approach for instances with up to 40 ships is 12 minutes, while the B&B procedure fails to solve 16 out of 64 instances within 12 hours. For the instances with 50 and 60 ships, the B&B approach could only solve a single instance to optimality and failed to produced a feasible solution for 7 out of 32 instances. The decomposition approach on the other hand finds feasible solutions for all instances, and attests optimality in 24 cases. Finally, for the instances consisting of 70 to 90 ships, the decomposition method solves 21 out of the 48 instances to optimality while for the remaining instances, feasible solutions were found.

5.2. No FCFS single chamber lock

The second series of experiments is conducted with the same traffic and lock data, 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
Figure 2: Comparison of the computation time of the different cut generation methods for a single large chamber lock, under a FCFS policy.

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. In the MP, a number of constraints are dropped as they 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 on 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 the previous subsection 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 3 (a) and (b) compare the performance of various methods. For the small chamber lock, using initial subset cuts appears to be the best approach. For the larger chamber, a combination of initial order cuts and feasibility subset cuts works best. In either case, the Benders’ approaches outperform the B&B approach by a large extent.
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.

5.3. FCFS parallel identical chamber lock

We now summarize our results for locks with two identical parallel chambers. Again the results for instances with 10 and 30 ships or more were omitted. Figure 4 (a) shows that for the small parallel chamber lock, the Benders’ approaches with feasibility subset and initial subset cuts are significantly slower than the B&B approach for the instances with the largest average inter arrival time. On the other instances, the Benders’ decomposition is nearly always faster. For a large parallel chamber lock (Figure 4 (b)), the feasibility and initial subset cut generation methods are much faster than the B&B approach when the ship inter arrival time is short. For the instances with medium inter arrival time (∼ 10 minutes) no clear winner can be found, while the B&B approach is the best choice when facing large inter arrival times.

5.4. FCFS multi chamber type lock

The results for the multi chamber type lock are summarized in Figure 5. Here only the instances having 30 ships or more are omitted. Similar to the SSC results, it appears that the ship inter arrival time influences the required computation time. Applying initial subset cuts produces the best results for the MCT instances: 31 out of 32 instances are solved to optimality, whereas the B&B approach only solves 22 instances within the 12 hour time limit.
6. Conclusion

An exact decomposition method for the lock scheduling problem is introduced in this paper. By solving the scheduling and assignment parts of LSP in the master problem and dealing with the ship placement aspect of LSP in the sub problem, an efficient solution method is realized. Not only are most big-M constraints removed from the model, thereby enabling a much tighter linear relaxation of the master problem, the resulting sub problem is also one for which some very effective (exact) solution methods exist. Through the application of efficient cut separation methods for several types of combinatorial Bender’s cuts in a Branch-and-Bound scheme, numerous new optimal results are obtained for instances with up to 90 ships. The decomposition
approach is most effective when several ships can be transferred in a single lockage operation, i.e. when the solution to the packing part of LSP is non-trivial. At the present time, the master problem is the limiting factor of this solution approach. Any improvements to the MP solution method with respect to, for example, convergence speed and scalability would enable solving larger instances to optimality in shorter computation times.

**Acknowledgement**

Research funded by a Ph.D. grant 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**


