The Lockmaster's Problem

Sofie Coene[1] and Frits C. R. Spieksma[1]

1 **Research group Operations Research and Business Statistics (ORSTAT)**
Katholieke Universiteit Leuven, Belgium
Sofie.Coene@econ.kuleuven.be

Abstract

Inland waterways form a natural network that is an existing, congestion free infrastructure with capacity for more traffic. The European commission promotes the transportation of goods by ship as it is a reliable, efficient and environmental friendly way of transport. A bottleneck for transportation over water are the locks that manage the water level. The lockmaster's problem concerns the optimal strategy for operating such a lock. In the lockmaster's problem we are given a lock, a set of ships coming from downstream that want to go upstream, and another set of ships coming from upstream that want to go downstream. We are given the arrival times of the ships and a constant lockage time; the goal is to minimize total waiting time of the ships. In this paper a dynamic programming algorithm (DP) is proposed that solves the lockmaster's problem in polynomial time. We extend this DP to different generalizations that consider weights, water usage, capacity, and (a fixed number of) multiple chambers. Finally, we prove that the problem becomes strongly NP-hard when the number of chambers is part of the input.

1998 ACM Subject Classification F.4.1 Computability Theory

Keywords and phrases Lock Scheduling, Batch Scheduling, Dynamic Programming, Complexity

Digital Object Identifier 10.4230/OASIcs.ATMOS.2011.27

1 Introduction

Transportation of goods by ship, over sea as well as over waterways, is a promising alternative for transport over land. Reasons are its reliability, its efficiency (a ship of 1200 tons can transport as much as 40 train couches and 60 trucks), and its environmental friendliness. Here, we focus on transport by inland ships over waterways. The European Commission promotes the better use of inland waterways in order to relieve heavy congested transport corridors. Carriage of goods by inland waterways is a mode of transport which can make a significant contribution to sustainable mobility in Europe [6, 1]. Not only is its energy consumption per km/ton of transported goods approximately 17% of that of road transport and 50% of rail transport, it also has a high degree of safety and its noise and gas emissions are modest. This natural network is the only existing infrastructure that is congestion free and has capacity for more traffic [8]. Typically, these waterways are interrupted by locks such that higher water levels can be maintained and such that larger and heavier ships are able to use it. These locks are a bottleneck for transportation over water and hence, operating locks wisely contributes to the popularity of transportation over water. However, the algorithmic problem how to operate a lock has not been studied broadly in the scientific literature. We aim to fill this gap. We now continue with the description of a very basic situation that will act as our core problem: the lockmaster's problem. Later, we will discuss extensions to more realistic settings. Consider a lock consisting of a single chamber. Ships coming from upstream, wanting to go downstream, arrive at the lock at given times r_i, $i = 1, \ldots, n_1$ with $r_1 \leq r_2 \leq \ldots \leq r_{n_1}$. Other ships, coming from downstream, wanting to go upstream, arrive

11th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems.
Editors: Alberto Caprara & Spyros Kontogiannis; pp. 27–37

OpenAccess Series in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

at the lock at given times s_i, $i = 1, \ldots, n_2$ with $s_1 \leq s_2 \leq \ldots \leq s_{n_2}$. Let $n = n_1 + n_2$, and let T denote the lockage duration, this is the time between closing the lock for entering ships, and opening the lock so that ships can leave. We assume that all data are integral. Our goal is to find a feasible lock-strategy that minimizes total waiting time of all ships. In other words, we need to determine at which moments in time the lock should start to go up (meaning at which moments in time ships that are downstream are lifted), and at which moments in time the lock should start to go down (meaning at which moments in time ships that are up are being lowered). Clearly, for such a strategy to be feasible, (i) going-up moments and going-down moments (referred to as moments) should alternate, and (ii) consecutive moments should be at least T time-units apart. It is clear that this particular problem is a simplified version of reality; we will, however, add capacity restrictions and other extensions in Section 4.

2 Literature

Literature on lock scheduling problems is rather limited. Some recent papers deal with the optimal sequencing for locking ships when a queue emerges due to some lock malfunction or accident. Nauss [13] determines an optimal sequence in the presence of setup times and non-uniform lockage processing times. Smith et al. [15] perform a simulation study on the impact of alternative decision rules and infrastructures improvement on traffic congestion in a section of the Upper Mississippi River. Ting and Schonfeld [16] study several control alternatives, such as sequencing, in order to improve lock service quality. They use heuristic methods. Verstichel and Vanden Berghe [17] mention the increasing occupation of logistic infrastructure in ports and waterways. They develop (meta)heuristics for a lock scheduling problem where a lock has at least one chamber, but often consists of multiple parallel chambers of different dimensions and lockage times. They deal with capacity restrictions in the sense that ships have sizes and the lock area is restricted, making this problem at least as hard as a bin packing problem. None of these papers study the lockmaster's problem.

The lockmaster's problem is closely related to a batch scheduling problem. Batch scheduling involves a machine that can process multiple jobs simultaneously. As far as we are aware, this connection has not been observed so far. Suppose that, in our problem, we only have downstream going ships. Then, the lock can be seen as a batching machine and the jobs are the arriving ships with release dates and equal processing times (i.e. the lockage time T). Following the notation of Baptiste [2] this is problem $1|p-batch, b = n, r_i, p_i = p|\sum w_i F_i$. In words: we have a single parallel batching machine with unrestricted capacity ($b = n$), release dates on the jobs, and uniform processing times. The objective is to minimize the sum of weighted flow times, however, in the basic lockmaster's problem there are only unit weights. Baptiste [2] shows that this problem is polynomially solvable for a variety of objective functions. Cheng et al. [5] developed an $O(n^3)$ algorithm for $1|p - batch, b = n, r_i, p_i = p|f$ where f can be any regular objective function. Condotta et al. [7] show that feasibility of the same problem with deadlines can be checked in $O(n^2)$. Clearly, the lockmaster's problem is more general. Indeed, when there are upstream going and downstream going ships, we are dealing with two families of jobs, and only jobs of the same family can be together in a batch. Further, in our case, processing a batch of one family needs to be alternated by processing a (possibly empty) batch containing jobs of the other family; i.e. it is not possible to process two batches of the same family consecutively. The concept of a "family" of jobs is also described by Webster and Baker [18], however not in combination with a batch processing machine. In their paper, Webster and Baker deal with a scheduling problem where scheduling jobs of

the same family consecutively reduces setup times. In our problem, dealing with jobs of the same family consecutively, i.e. in one batch, reduces the total batching time. This type of problem is also known under the name of batch scheduling with job compatibilities. Jobs within a batch need to be pairwise compatible, and these compatibilities can be expressed using a compatibility graph. Boudhar [3] and Finke et al. [9] study different variants of these batch scheduling problems when the compatibility graph is bipartite or an interval graph. In our case the compatibility graph is the union of two cliques. Our problem can be summarized as being $1|p-batch, b=n, r_i, p_i=p, \Phi=2, s_{fg}|\sum F_i$, with $s_{fg}=2T$ if $f=g$ and $s_{fg}=T$ if $f\neq g$, where Φ refers to the number of families and s_{fg} to the setup times between batches; we will refer to our problem as the lockmaster's problem. For a review on scheduling a batching machine we refer the reader to Potts and Kovalyov [14] and Brucker et al. [4]. Lee et al. [12] develop dynamic programming algorithms for scheduling a batching machine with release dates, deadlines, and constant processing times when the goal is to minimize makespan or minimize the number of tardy jobs. In conclusion, this literature study reveals that the complexity of our lockmaster's problem does not follow from results in literature. Further, we consider the lockmaster's problem with multiple parallel chambers. Again, when considering the uni-directional case, the problem is related to parallel batch scheduling problems. Condotta el al. [7] have developed a polynomial time algorithm in case of parallel batching machines and deadlines where the objective is minimizing the maximum lateness.

2.1 Our results

We show that
(1) there is an $O(n^6)$ algorithm for the lockmaster's problem (see Section 3);
(2) this algorithm can be extended to deal with regular objective functions (4.1), non-uniform lockage times (4.2), settings with a limited number of times that there can be locked (4.3), capacities (4.4), and with a constant number of parallel chambers (4.5);
(3) if the number of parallel chambers is part of the input, the problem becomes strongly NP-hard (Section 5).

3 A DP for the lockmaster's problem

When is a lock likely to start going up or down? Either upon arrival of a ship or immediately upon arrival of the lock. This suggests that the number of moments the lock starts moving is limited. Garey et al. [11] and recently Condotta et al. [7] use the concept of "forbidden regions" in the presence of deadlines to define periods of time in which no job/batch can start in a feasible schedule. Given that there are no deadlines, the same concept can be used to define periods of time in which no batch can start in an optimal schedule. We introduce a set of moments U at which it is possible to go up. These upmoments are referred to as u_i. Similarly, we introduce a set of moments at which it is possible to go down, the set D. These downmoments are referred to as d_i. Let us define set $S=\{s_i\}$, set $R=\{r_j\}$ and $\Theta=\{0, 2T, 4T, \ldots, 4nT\}$; further in the text it will be shown why this set is limited to $4nT$. We use the Minkowski-sum to sum two sets, i.e. the sum of two sets $A=\{a_i\}$ and $B=\{b_j\}$ as follows:

$$A+B=\{a_i+b_j | a_i\in A, b_j\in B\}.$$

Then, bearing this definition in mind, here is a proposal for U and for D:

$$U=(S+\Theta)\cup(R+\Theta+\{T\}),$$

$D = (R + \Theta) \cup (S + \Theta + \{T\})$.

For example, suppose we have two ships traveling downstream and two ships traveling upstream with $R = \{1, 7\}$ and $S = \{2, 4\}$ and $T = 5$. Then,
$U = \{2, 4, 6, 12, 14, 16, 22, 24, 26, \ldots, 162, 164, 166\}$ and
$D = \{1, 7, 9, 11, 17, 19, 21, 27, 29, \ldots, 161, 167, 169\}$.

We will come back to the cardinality of U and D.

▶ **Lemma 1.** *There is an optimal lock strategy for the lockmaster's problem whose upmoments are contained in U, and whose downmoments are contained in D.*

Proof. Contradiction. Suppose there is an instance such that each optimal strategy has either an upmoment not in U or a downmoment not in D (such a moment is called a failure). Consider an optimal strategy for this instance for which its earliest failure is minimal, say at time t. Let us assume for convenience that at time t, the lock went up. Notice that t cannot be equal to an s_i. Consider that moment in time t. Let $\epsilon > 0$ be a very small quantity. There are two possibilities:

(i) at $t - \epsilon$ the lock was waiting to go up. If, in our optimal strategy, there are ships transported up at time t, it cannot have been optimal to wait until t, since no downstream ships arrive at time t (since t is not in S). Hence, there are no ships transported. But then, we need not have waited, and there is an optimal strategy that immediately went up after the last time before t we went down.

(ii) at $t - \epsilon$ the lock was going down. Thus, at $t - T$, the lock started a down-operation. This moment in time is, by assumption, in D. But then it follows that t is in U. Contradiction.

◀

Now, let us further analyze U and D.

▶ **Lemma 2.** *When, for a given instance for the lockmaster's problem, during a time period equal to $4T$ no ships arrive at the lock, the instance can be divided into two instances. The solution can then be found by solving these two smaller instances.*

Proof. We observe that if during a period of time of length $4T$ nothing happens (meaning that there are no ship arrivals), the instance can be subdivided into two instances. Indeed, suppose the final arriving ship in the instance is an upstream going ship with arrival time s_p (the same analysis can be done when the final ship arriving is going downstream). The latest possible optimal lockage time for this ship is $s_p + 2T - \epsilon$, with $\epsilon > 0$ and small. Suppose that this ship is locked at time $t \geq s_p + 2T$. Since it was the final ship arriving, it would have been better to lock the ship at time $t - 2kT$ with k an integer such that $t - 2kT$ is in $[s_p, s_p + 2T)$. If this ship is locked at time $s_p + 2T - \epsilon$, then the lock is at upstream level at $s_p + 3T - \epsilon$ and again at downstream level at $s_p + 4T - \epsilon$. This means that, when no ship arrives in a time interval of $4T$, the instance can be split into two separate instances. ◀

From now on we assume (without loss of generality, due to lemma 2) that each instance of the lockmaster's problem has the property that a ship arrives during any $4T$ interval. This allows us to bound the cardinality of $U \cup D$. For each period of time of length $4T$, we have at least one arrival. In a $4T$ interval there can be at most $O(n)$ elements in $U \cup D$ by the construction of the sets. Partition the time-axis into consecutive $4T$ intervals: there can be at most n of them (since each needs to contain at least one arrival). Thus, there are at most $O(n^2)$ elements in $U \cup D$.

We now define a dynamic programming algorithm (DP) where $f(u_i, d_j)$ (with $u_i \leq d_j - T$) represents the minimal costs of a lockage strategy that takes care of all up-requests up to u_i, all down-requests up to d_j, which features an upmoment at time $t = u_i$, which features a downmoment at time $t = d_j$, and such that there are no other up- or downmoments in between u_i and d_j.

Here is a recursion. For each $u_i \in U$, $d_j \in D$, with $u_i \leq d_j - T$ we have:

$$f(u_i, d_j) = \min_{\substack{d_{j'} \leq u_i - T \\ u_{i'} \leq d_{j'} - T}} \{f(u_{i'}, d_{j'}) + \sum_{\ell : s_\ell \in (u_{i'}, u_i]} (u_i - s_\ell) + \sum_{k : r_k \in (d_{j'}, d_j]} (d_j - r_k)\};$$

for all $u_i > d_j - T$ we set:

$$f(u_i, d_j) = \infty.$$

The recursion is initialized as follows:

$$f(u_1, u_1 + T) = 0.$$

For this recursion to work we set $u_1 = \min\{s_1, r_1 - T\}$. The optimal value is given by $\min\{f(u_i, d_j) | u_i \geq s_{n_2}, d_j \geq r_{n_1}, u_i \in U, d_j \in D\}$. A straightforward way to determine the complexity of DP is to observe that there are $O(n^4)$ states and since for each state we enumerate over all other states, we arrive at an $O(n^8)$ algorithm. To improve the time complexity of DP, we observe the following.

▶ Observation. If $d_j \in D \setminus R$, then the previous upmoment was $d_j - T$.

Indeed, notice that if the lock goes down at a moment in time (say t) that is not an arrival moment in R, then the previous upmoment was at $t - T$. If the lock went up earlier than $t - T$, then there is an optimal solution in which the next downmoment is earlier than t; as no ship is arriving at t, there is no need to wait for t.

▶ **Theorem 3.** *DP is a polynomial-time algorithm for the lockmaster's problem.*

Proof. Correctness follows from lemma's 1 and 2 and the following. We argue that the observation above implies that it is sufficient for DP to consider $O(n^3)$ states. Indeed, there are $O(n^2)$ states with $d_j \in D \setminus R$, and $O(n^3)$ states with $d_j \in R$. The latter fact follows from the insight that $|R| = O(n)$ (combined with the fact that U and D have cardinality $O(n^2)$). Computing each state can be done by evaluating $O(n^3)$ states, leading to a total time complexity for this algorithm equal to $O(n^6)$. ◀

4 Extensions

4.1 Regular objective functions

For the analysis above we chose as an objective to minimize the sum of the waiting times, which is a very natural objective function for this problem. The algorithm, however, works for any regular, i.e. non-decreasing in (waiting) time, objective function. Such a function can be for instance minimizing the weighted sum of waiting times or minimizing the maximum waiting time. Indeed, in the recursion, a cost of a state can be computed by taking the cost of a previous state and adding the extra cost incurred. These are cost-functions non-decreasing in t and it is clear how the extra cost can be calculated, independent of the value of the previous state. Let us consider, for example, the weighted lockmaster's problem. In practice, it happens that not all ships are of equal importance, e.g. it is conceivable that the waiting

cost for ships transporting goods is higher than the waiting cost of leisure ships or ships transporting dangerous goods get priority over normal cargo ships. This can be dealt with by assigning weights to the ships revealing their priority. Taking into account weights w for the ships in the DP recursion can be done straightforwardly as follows:

$$f(u_i, d_j) = \min_{\substack{d_{j'} \leq u_i - T \\ u_{i'} \leq d_{j'} - T}} \{f(u_{i'}, d_{j'}) + \sum_{\ell : s_\ell \in (u_{i'}, u_i]} w_\ell (u_i - s_\ell) + \sum_{k : r_k \in (d_{j'}, d_j]} w_k (d_j - r_k)\}.$$

Initialization and determination of the optimal value are identical to the basic DP in the previous section.

4.2 Non-uniform lockage times

It is not uncommon that lockage times for going up (T_u) and down (T_d) are not equal. Then, for $u_i \in U$ and $d_j \in D$:

$$f(u_i, d_j) = \min_{\substack{d_{j'} \leq u_i - T_d \\ u_{i'} \leq d_{j'} - T_u}} \{f(u_{i'}, d_{j'}) + \sum_{\ell : s_\ell \in (u_{i'}, u_i]} (u_i - s_\ell) + \sum_{k : r_k \in (d_{j'}, d_j]} (d_j - r_k)\}$$

where Θ, U and D are now:

$$\Theta = \{0, T_d + T_u, 2(T_d + T_u), 3(T_d + T_u), \ldots, n(2T_d + 2T_n)\},$$

$$U = (S + \Theta) \cup (R + \{T_d\} + \Theta),$$

$$D = (R + \Theta) \cup (S + \{T_u\} + \Theta).$$

It is not difficult to verify that all results from Section 3 apply to this setting. Also, initialization and the optimal solution are determined equivalently to the basic DP.

4.3 Water usage

Due to organizational/environmental reasons, there could be a limit on the number of times there can be locked. In this situation, Lemma 2 no longer holds. Indeed, splitting an instance, would also mean dividing the number of allowed lockage times over the two instances and it is not straightforward how this should be done. The cardinality of U and D needs to be reconsidered. Let us define alternative sets U' and D' as follows. For all pairs of consecutive ships (t, t') with $m_{t'} - m_t \geq 4T$ and $m_t, m_{t'} \in S \cup R$, let $U' = U \backslash \{u_i | u_i \in [m_t + 4T, m_{t'})\}$ and $D' = D \backslash \{d_j | d_j \in [m_t + 4T, m_{t'})\}$.

▶ **Lemma 4.** *All optimal up and downmoments are in U' and D' respectively.*

Proof. From Lemma 1 we know that all optimal up- and downmoments are contained in U and D. Suppose a ship arrives at time m_t and during a time period of $4T$ after that no other ships arrive. Suppose further, without loss in generality, that the ship arriving at m_t is an upward going ship. Then, following the same argument as in the proof of Lemma 2, all u_i and d_j later than $m_t + 4T$ and earlier than $m_{t'}$, the first arrival after m_t, will not be part of an optimal solution and can be deleted from the sets U and D. ◀

What is now the cardinality of U' and D'? When, in an instance, there is no gap of $4T$, it holds that every $4T$ interval at least one ship arrives, and there are at most n such intervals, yielding size $O(n^2)$. When there are x such gaps, cardinality is x times $O(n^2)$, with $x \leq n$, yielding size $O(n^3)$.
In a dynamic programming recursion for this problem (DPw), an entry is needed to keep track of the number of times there has been locked before. It still holds that all ships arrived before or upon lockage time will be handled. Now, we use v for the number of times there has already been locked and V for the maximum number of times there can be locked. For $u_i \in U'$, $d_j \in D'$,$v \leq V$, the algorithm DPw is given by:

$$f(u_i, d_j, v) = \min_{\substack{d_{j'} \leq u_i - T \\ u_{i'} \leq d_{j'} - T}} \{f(u_{i'}, d_{j'}, v-1) + \sum_{\ell : s_\ell \in (u_{i'}, u_i]} (u_i - s_\ell) + \sum_{k : r_k \in (d_{j'}, d_j]} (d_j - r_k)\}.$$

The initial state is

$$f(u_1, u_1 + T, V - 1) = 0$$

with $u_1 = \min\{s_1, r_1 - T\}$.
The optimum is given by $\min\{f(u_i, d_j, v | u_i \geq s_{n_2}, d_j \geq r_{n_1}, v \leq V)\}$. In these states all ships are locked and the maximum number of allowed lockage times is not exceeded.

▶ **Lemma 5.** *DPw is a polynomial-time algorithm for the water-usage constrained lockmaster's problem.*

Proof. See also the proof of Theorem 3. There are $O(Vn^4)$ states, with $V \leq n$. Computing each state can be done by evaluating $O(n^4)$ states, leading to a total time complexity for the algorithm equal to $O(Vn^6)$. ◀

4.4 Capacity

Until now we did not take into account any capacity restrictions. Suppose the sizes of the ships are uniform and the lock can accommodate at most b ships at once. It is easy to see that Lemma 1 and its proof also hold in this case. Upmoments and downmoments in an optimal solution will be contained in U and D, respectively. However, Lemma 2 is not directly applicable. Indeed, it can happen that ships need to wait longer than $4T$ when the capacity of the lock is filled. Suppose that during a certain time period no ships arrive at the lock. Then, the lock will go up and down with full capacity and without waiting until the waiting queue is empty. In other words, the strategy of the lock is very simple in this time period. Given that there are $n_1 + n_2$ ships in the instance, let $\eta = max\{n_1, n_2\}$. Then the following lemma holds:

▶ **Lemma 6.** *When, for a given instance of the lockmaster's problem with capacity constraint, during a time period equal to $2T\lceil \frac{\eta}{b} \rceil$ no ships arrive at the lock, the instance can be divided into two instances. The solution can then be found by solving these two smaller instances.*

Proof. Suppose η ships are waiting at the lock to go up, then the lock needs to go up and down until all ships are handled. Given that the lock has a capacity b, the queue will be empty after at most $\lceil \frac{\eta}{b} \rceil$ upmoments of the lock. $2T$ time units pass between two upmoments, such that the last ships go upstream at time $2T(\lceil \frac{\eta}{b} \rceil - 1)$. Note that the lock does not spend any time waiting as the ships have already arrived and are waiting to move as soon as possible. Thus, T time units later the lock is at the upstream level, and another T time

units later again at the downstream level. If during $2T\lceil\frac{\eta}{b}\rceil$ time units no ships arrive, the instance can be split into two separate instances. ◀

It follows that we can assume, without loss of generality, that each $2T\lceil\frac{\eta}{b}\rceil$ time units at least one ship and at most n ships arrive. In a $2T(\lceil\frac{\eta}{b}\rceil+1)$ interval there can be at most $O(n^2)$ elements in $U\cup D$ for that interval. We have at most n intervals, such that there are at most $O(n^3)$ elements in $U\cup D$.

Define a dynamic programming algorithm (DPc) with $f(u_i,d_j,p,q)$ (with $u_i\le d_j-T$) as the minimal costs of a lockage strategy that includes the accumulated cost for all up-requests up to u_i and the cost for all down-requests up to d_j. Part of these ships is still waiting at the lock, i.e. p is the number of ships waiting to go upstream and q is the number of ships waiting to go downstream; the cost for these ships is only partial (indeed, their waiting time is not completed yet). This state features an upmoment at time $t=u_i$, a downmoment at time $t=d_j$, and there are no other up- or downmoments in between u_i and d_j. Let $l(u_{i'},u_i)$ be equal to the number of ships i with arrival time s_i in the interval $(u_{i'},u_i]$ and $k(d_{j'},d_j)$ the number of ships j with arrival time r_j in the interval $(d_{j'},d_j]$.
Then, let:

$$P=\begin{cases}\{max\{p+b-l(u_{i'},u_i),0\}\} & \text{if } p>0\\ \{0,1,\dots,b-l(u_{i'},u_i)\} & \text{if } p=0\end{cases}$$

$$Q=\begin{cases}\{max\{q+b-k(d_{j'},d_j),0\}\} & \text{if } q>0\\ \{0,1,\dots,b-k(d_{j'},d_j)\} & \text{if } q=0\end{cases}$$

and:

$$f(u_i,d_j,p,q)=\min_{\substack{d_{j'}\le u_i-T\\ u_{i'}\le d_{j'}-T\\ p'\in P\\ q'\in Q}}\{f(u_{i'},d_{j'},p',q')+ \sum_{\ell:s_\ell\in(u_{i'},u_i]}(u_i-s_\ell)+\sum_{k:r_k\in(d_{j'},d_j]}(d_j-r_k)+p'(u_i-u_{i'})+q'(d_j-d_{j'})\}. \quad (1)$$

with initial state

$$f(u_1,u_1+T,0,0)=0$$

and $u_1=\min\{s_1,r_1-T\}$.
When $p,q>0$ it means that the lock was operated at full capacity in the previous state. Just before operating the lock there were thus $p+b$ ships ready to go up, from which $l(u_{i'},u_i)$ arrived between the previous upmoment of the lock and the current upmoment. Thus, after the previous upmoment of the lock there were $p+b-l(u_{i'},u_i)$ ships not handled yet. If this is a negative number it means that all ships are handled up till $s_l\le u_{i'}$ and $p'=0$. When $p,q=0$, it means that no ships are waiting and full capacity b was not necessarily used, meaning that $p'+l(u_{i'},u_i)\le b$. It follows that $p'\le b+l(u_{i'},u_i)$, and idem for q'. The waiting time of any ship l that arrived between u_i and $u_{i'}$ is at least u_i-s_l, which explains the second part of (1). However, for the p' ships that could not enter the lock at $u_{i'}$, the waiting time increases with $(u_i-u_{i'})$, which is dealt with in the third part. Analogue arguments hold for the downmoments. The optimal value is given by $\min\{f(u_i,d_j,0,0)|u_i\ge s_{n_2},d_j\ge r_{n_1},u_i\in U,d_j\in D\}$.

▶ **Theorem 7.** *DPc is a polynomial-time algorithm for the lockmaster's problem with a capacity restriction.*

Proof. The proof is analogue as the proof for Theorem 3. U and D have cardinality $O(n^3)$; p and q have cardinality $O(n)$. It follows that this algorithm will have $O(n^6)$ states. For each state we need to evaluate $O(n^6)$ states, yielding a total time complexity of $O(n^{12})$. ◀

Notice that when the ships are weighted, ships might no longer be locked in order of arrival and hence algorithm DP (or an extension of it) might fail to find an optimal solution. When considering the unidirectional case, Baptiste's algorithm [2] (see Section 2) yields a polynomial time procedure.

4.5 Multiple (parallel) chamber lock

In practice a lock often consists in multiple chambers that operate independently such that ships can be dealt with in parallel. We will show that when the number of chambers is independent from the input and all chambers have identical lockage times, the problem can be solved in polynomial time by adapting DP. However, when the number of chambers is part of the instance and the chambers have arbitrary lockage times, the problem becomes NP-hard.

First, consider a problem with $k < n$ identical chambers in parallel, lockage time for all locks is equal to T. All possible lockage times are identical to the single chamber case, such that Lemma 1 and Lemma 2 are applicable. Indeed, each of the chambers will only move upon arrival of a ship or immediately after an up- (or down-) movement of the chamber. Let $\bar{u}_i$ be a vector of size k containing elements from U, thus $\forall l \leq k : \bar{u}_i(l) \in U$; and $\bar{d}_j$ a vector of size k containing elements from D, thus $\forall l \leq k : \bar{d}_j(l) \in D$.

▶ **Lemma 8.** *Given that there are k uniform parallel chambers in the lockmaster's problem, an optimal solution exists where the lockage sequence of the chambers is ordered as follows $\bar{u}_i^*(1) < \bar{u}_i^*(2) < \ldots < \bar{u}_i^*(k)$ and $\bar{d}_j^*(1) < \bar{d}_j^*(2) < \ldots < \bar{d}_j^*(k)$, $\forall \bar{u}_i^*, \forall \bar{d}_j^*$.*

Proof. Suppose the optimal solution is not in accordance to Lemma 8. Then, there is a moment in time where the lockage sequence alters, let this moment be e.g. $\bar{d}_j(2) < \bar{d}_j(1)$. Given that $\bar{u}_i(1) < \bar{u}_i(2)$, it holds that chamber 1 is available to go down earlier than chamber 2. All chambers are identical, thus the solution value will not change when chamber 1 goes down at $t = \bar{d}_j(2)$ and chamber 2 at $t = \bar{d}_j(1)$, yielding a solution as described in Lemma 8. ◀

Let us now define $f(\bar{u}_i, \bar{d}_j)$ with $\bar{u}_i$ and $\bar{d}_j$ ordered and $\bar{u}_i(l) \leq \bar{d}_j(l) - T, \forall l \in 1 \ldots k, \bar{u}_i(l) \in U, \bar{d}_j(l) \in D$, as the minimal cost of a lockage strategy where all up requests up to $t = \bar{u}_i(k)$ and all down requests up to $t = \bar{d}_j(k)$ are dealt with. For each $l \in 1 \ldots k$, chamber l moves up at time $t = \bar{u}_i(l)$ and down at time $t = \bar{d}_j(k)$ and there are no other up- or down-moments in between.

Then, for all $\bar{u}_i$ and $\bar{d}_j$, $\bar{u}_i(l) \leq \bar{d}_j(l) - T$ we have

$$f(\bar{u}_i, \bar{d}_j) = \min_{\substack{\bar{u}'_i(k) < \bar{u}_i(1) \\ \bar{d}'_j(k) < \bar{d}_j(1)}} \{ f(\bar{u}'_i, \bar{d}'_j) + \sum_{l=0\ldots k-1} \sum_{\substack{m: \\ s_m \in (\bar{u}_i(l), \bar{u}_i(l+1)]}} (\bar{u}_i(l) - s_m) + \sum_{l=0\ldots k-1} \sum_{\substack{o: \\ r_o \in (\bar{d}_j(l), \bar{d}_j(l+1)]}} (\bar{d}_j(l) - r_o) \},$$

with $\bar{u}_i(0) = \bar{u}'_i(k)$ and $\bar{d}_j(0) = \bar{d}'_j(k)$. The optimal value is given by $\min\{f(\bar{u}_i, \bar{d}_j) | \bar{u}_i(k) \geq s_{n_2}, \bar{d}_j(k) \geq r_{n_1}, \bar{u}_i(l) \in U, \bar{d}_j(l) \in D, \forall l \leq k\}$.

▶ **Lemma 9.** *The lockmaster's problem with multiple identical parallel chambers is solvable in polynomial time.*

Proof. See also the proof of Theorem 3. There are $O(n^{3k})$ states, computing each state can be done by evaluating $O(n^{3k})$ states, leading to a total time complexity for the algorithm equal to $O(n^{6k})$. ◀

5 Non-identical parallel chambers

In this section we prove that in the case of multiple non-identical parallel chambers where the number of chambers is part of the input, the lockmaster's problem is NP-hard.

▶ **Lemma 10.** *The lockmaster's problem with non-identical parallel chambers is strongly NP-hard.*

Proof. We show that the lockmaster's problem with multiple non-identical parallel chambers is at least as hard as numerical matching with target sums (NMTS). In an instance of NMTS we are given positive integers a_i $(1 \leq i \leq n)$, b_j $(1 \leq j \leq n)$ and t_κ $(1 \leq \kappa \leq n)$. It holds that $\sum_\kappa t_\kappa = \sum_{i,j}(a_i + b_j)$. The question is whether there exists a collection of m triples (i, j, κ) such that (i) $a_i + b_j = t_\kappa$ for each triple, and (ii) each integer in the input occurs exactly once. This problem is proven to be NP-hard by Garey and Johnson [10]. We assume, without loss of generality, that the a_i's and t_κ's are pairwise different and that $\min_j b_j > \max_i a_i$. We now construct an instance of the lockmaster's problem as follows. There are $2n$ ships, n ships travel upstream and arrive at the lock at $s_l := a_i$ and n ships travel downstream arriving at the lock at times $r_k := t_\kappa$. There are n chambers, each with a certain lockage time b_j. Is there a solution for the lockmaster's problem with total waiting time equal to 0? If there is a solution to NMTS, each triple (a_i, b_j, t_κ) corresponds to a combination of a chamber with an upstream and a downstream going ship. The upstream going ship arrives at time a_i, enters the chamber that needs b_j time units to arrive at the downstream level and after t_κ time units the downstream going ship enters the chamber and spends b_j time units in the lock. Each ship can enter a chamber upon arrival time and total waiting time is equal to 0. On the other hand, if a solution to the lockmaster's problem with value 0 exists, it means that each upstream going ship is assigned upon arrival to exactly one chamber. Moreover, since $\min_j b_j > \max_i a_i$, it follows that each chamber accommodates one upstream going ship. Since downstream going ships also have waiting time equal to 0, there must exist triples for which it holds that $a_i + b_j = t_\kappa$ and we have a solution to NMTS. ◀

References

1 H. Allaeys. Optimalisering van een sluis (in Dutch), 2010. Master Thesis, Katholieke Universiteit Leuven.

2 P. Baptiste. Batching identical jobs. *Mathematical Methods of Operations Research*, 52:355–367, 2000.

3 M. Boudhar. Scheduling a batch processing machine with bipartite compatibility graphs. *Mathematical Methods of Operations Research*, 57:513–527, 2003.

4 P. Brucker, A. Gladky, H. Hoogeveen, M. Y. Kovalyov, C. N. Potts, T. Tautenhahn, and S. L. Van De Velde. Scheduling a batching machine. *Journal of Scheduling*, 1:31–54, 1998.

5 T. C. E. Cheng, J. J. Yuan, and A. F. Yang. Scheduling a batch-processing machine subject to precedence constraints, release dates and identical processing times. *Computers and Operations research*, 32:849–859, 2005.

6 European commission. Promotion of inland waterway transport, January 2011. `http://ec.europa.eu/transport/inland/promotion/promotion-en.htm`.

7 A. Condotta, S. Knust, and N. V. Shakhlevich. Parallel batch scheduling of equal-length jobs with release and due dates. *Journal of Scheduling*, 13:463–477, 2010.

8 Inland Navigation Europe. Water webletter, November 2010. `www.inlandnavigation.org`.

9 G. Finke, V. Jost, M. Queyranne, and A. Sebő. Batch processing with interval graph compatibilities between tasks. *Discrete Applied Mathematics*, 156:556–568, 2008.

10 M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-completeness*. Freeman, San Francisco, 1979.

11 M. R. Garey, D. S. Johnson, B. B. Simons, and R. E. Tarjan. Scheduling unit-time tasks with arbitrary release times and deadlines. *SIAM Journal on Computing*, 10:256–269, 1981.

12 C. Lee, R. Uzsoy, and L. A. Martin-Vega. Efficient algorithms for scheduling semiconductor burn-in operations. *Operations Research*, 40:764–775, 1992.

13 R. M. Nauss. Optimal sequencing in the presence of setup times for tow/barge traffic through a river lock. *European Journal of Operational Research*, 187:1268–1281, 2008.

14 C. N. Potts and M. Y. Kovalyov. Scheduling with batching: A review. *European Journal of Operational Research*, 120:228–249, 2000.

15 L. D. Smith, D. C. Sweeney, and J. F. Campbell. Simulation of alternative approaches to relieving congestion at locks in a river transportation system. *Journal of the Operational Research Society*, 60:519–533, 2009.

16 C. Ting and P. Schonfeld. Control alternatives at a waterway lock. *Journal of Waterway, Port, Coastal, and Ocean Engineering*, 127:89–96, 2001.

17 J. Verstichel and G. Vanden Berghe. A late acceptance algorithm for the lock scheduling problem. *Logistik Management*, 5:457–478, 2009.

18 S. Webster and K. R. Baker. Scheduling groups of jobs on a single machine. *Operations Research*, 43:692–703, 1995.