Velocity optimisation on waterways

Doctoral thesis

VELOCITY OPTIMISATION ON WATERWAYS

Julian Golak

2021

VELOCITY OPTIMISATION ON WATERWAYS

Dissertation

To obtain the degree of Doctor at Maastricht University,
on the authority of the Rector Magnificus, Prof. Dr. R.M. Letschert,
in accordance with the decision of the Board of Deans,
to be defended in public
on Tuesday 16th of November 2021, at 13.00 hours

by

Julian Arthur Pawel Golak

Promotor
Prof. Dr. A. Grigoriev

Copromotor
Dr. C. Defryn

Assessment Committee
Prof. Dr. T. Vredeveld (chair, Maastricht University)
Prof. Dr. F. C. R. Spieksma (Eindhoven University of Technology)
Dr. A. Abiad (Eindhoven University of Technology)
Prof. Dr. T. Comes (Maastricht University)

Published by Global Academic Press
Printed in The Netherlands by ProefschriftMaken
ISBN 978-94-6423-530-2

Acknowledgments

First and foremost, I would like to thank my supervisor, Alexander Grigoriev. Your enthusiasm, positivity and attitude is inspiring and made my PhD a fun and relaxing experience.

Second, I want to thank my co-supervisor Christof Defryn for his support and Aida Abiad, Tina Comes, Frits Spieksma and Tjark Vredeveld for the careful reading of this thesis.

Lastly, thanks to all my friends, family and co-workers for the great times throughout the years. Thank you Freija for your encouragements and our hourly coffee breaks, and thank you Anni and Nils for your patience and support during the last years, especially in Valencia.

Julian Golak
Valencia
June 1, 2021

Contents

1 Introduction

Transportation enables economic growth and job creation and thus it must be sustainable in the light of the new challenges we face. Oil will become scarcer in future decades, sourced increasingly from uncertain supplies. At the same time, the EU has called for a reduction of at least 90% of world greenhouse gas from the transportation sector by 2050 [1]. Among others, [1] lists two key strategies to achieve this goal:

1. Shift towards more sustainable transport modes,
2. Implementing new technologies for traffic management to lower transport emissions.

In comparison to other transportation modes, the use of waterways is more sustainable (less greenhouse gas emission) and relatively cheap (due to economies of scale). Moreover, as a single vessel can replace over 100 trucks, increased use of the water network is likely to reduce congestion and the number of accidents on the road network. The Netherlands, located around the mouth of multiple important European rivers, accounts for 58% of all European freight transportation

companies [2]. In 2018, the Dutch inland waterway transportation network consisted of a total of over $6297km$ of navigable inland waterways, on which 34.85% of all freight transport (in tonne-kilometres) took place [2], [3]. Furthermore, 9 of the essential inland waterways for transportation in the Netherlands contain locks [2]. In order to achieve reduction in CO_2 and greenhouse gas emissions, operational changes such as the digitalization of inland waterway operations has been proposed [2], [4]. As outlined in The European Green Deal [1] a goal of zero-emission and zero-pollution transportation is to be achieved via the increasing implementation of alternative fuels. The high cost of these alternative fuels further underlines the importance of fuel consumption.

Despite all efforts over the past decade(s), the majority of transport operations are still performed on the road using trucks [5]. In comparison to road transportation, the use of inland waterways is more environmentally friendly because of lower greenhouse gas emissions per volume or weight unit and it is relatively cheap due to economies of scale and the increased bundling opportunities. However, the inland waterway network is less dense than the road network.

Additionally, it contains many locks to overcome height differences between two adjacent river segments. As the capacity of these locks is limited, they form bottlenecks along the river network. Due to a lack of coordination between individual skippers (i.e., the person in charge of a vessel), queues are formed in front of these locks. To compensate for the waiting time, skippers need to speed up after a lock to arrive on time at their destination. Moreover, the lock management (i.e., the scheduling of all lock operations) is done based upon on intuition and ad-hoc decision making, typically resulting in a first-come-first-served queuing policy [1]. To ensure timely service at the lock, the skippers have the incentive to speed up in front of a lock, and overtake preceding vessels to receive the lock service earlier.

[1] Communicated to us by our industrial partner Trapps Wise. B.V.

Existing research on the optimization efficiency on waterways is mainly focused on lock scheduling. In a single lock scheduling problem, the operating times of a single lock are optimized for a set of vessels with given arrival times at the lock. By batching the vessels together and determining the optimal service time for each batch, the goal is to reduce overall waiting time at the lock.

A summary on literature and collection of theoretical results mentioned below can be found in the doctoral thesis by [6]. [7] provide a polynomial time algorithm to optimally solve the single lock scheduling problem, given the arrival times of the vessels and the capacity of the lock. A complexity analysis together with a polynomial time algorithm that applies to special cases for the single lock scheduling problem with multiple parallel chambers is presented in [8]. The problem of physically placing vessels inside the chamber of the lock has been addressed in [9] and [10]. In [11], authors consider the lock scheduling problem in an online setting. The joint optimization of multiple sequential locks on the river is considered by [12] and [13]. [13] propose a variable neighborhood search, whereas [12] use a MILP to find an exact solution. In the latter two contributions, the goal is to minimize the aggregated fuel cost or emission, while selfish behavior of skippers is not addressed. Furthermore, authors in [14] propose results on the complexity of scheduling vessels that travel along a waterway with multiple sequential locks.

There are also multiple case studies conducted for the lock scheduling problem, focused on specific lock sequences on important waterways in the world. [15] consider the *Welland Canal* in North America for which they provide a heuristic that employs optimal dynamic programming models for scheduling individual locks in order to determine operating schedules for the lock sequence. [16] present a simulation model to evaluate the quality of different heuristics on lock operations on the *Upper Mississippi River* in the US. This research has been extended by [17]. Here, the authors propose a MILP model to solve the lock scheduling problem with sequence-dependent setup and processing times. Using the same river segment, [18] incorporate the mal-

functioning of locks and study different responses to such a disruption so to minimize additional queue lengths. Also, a model for the lock scheduling problem with multiple parallel chambers for this river layout has been investigated by [19]. Finally, the *Kiel Canal* is considered by [20]. In their paper, the authors incorporate collision of ships in the optimization model and provide a heuristic to determine a routing and scheduling for a fleet of ships in a collision-free manner.

To the best of our knowledge, only [12] take into account that skippers can choose the speed of their vessel, and hence influence the time at which they arrive at the lock. Their objective is to minimize overall CO_2 emissions by optimizing the speed at which vessels have to approach the locks using a MILP formulation. This approach is closely related to the problems addressed in this thesis. We propose several extensions that arise from real world problems and call for new methods for increasing efficiency in inland waterway transportation by either directly optimising velocities or regulating them by imposing schedules on the lock operations.

Chapter 2 We address the problem of minimizing the aggregated fuel consumption by the vessels in an inland waterway, e.g., a river, with a single lock. The fuel consumption of a vessel depends on its velocity and the slower it moves, the less fuel it consumes. Given entry times of the vessels into the waterway and the deadlines before which they need to leave the waterway, we start from the optimal velocities of the vessels that minimize their private fuel consumption, where we assume selfish behavior of the skippers. Presence of the lock and possible congestion on the waterway make the problem computationally challenging. First, we prove that in general, a Nash equilibrium might not exist, i.e., if there is no supervision on the vessels' velocities, there might not exist a strategy profile from which no vessel can unilaterally deviate to decrease its private fuel consumption. Next, we introduce simple supervision methods to guarantee the existence of a Nash equilibrium. Unfortunately, though a Nash equilibrium can be computed,

the aggregated fuel consumption of such a stable solution can be high compared to the social optimum, where the total fuel consumption is minimized. Therefore, we propose a mechanism involving payments between vessels, guaranteeing a Nash equilibrium while minimizing the fuel consumption. This mechanism is studied for both the offline setting, where all information is known beforehand, and online setting, where we only know the entry time and deadline of a vessel when it enters the waterway.

Chapter 3 The speeding behavior is a direct consequence of the management and scheduling of waterways locks. In this study, we consider a periodic lock scheduling problem. We assume that streams of vessels are arriving periodically and we aim to minimize the long-run average waiting time. We introduce closed-form formula for the case of only 2 streams of arriving vessels. For the general case, we provide exact algorithms and incremental approximation algorithm. This creates tools for policy makers to further reduce CO_2 emission in the inland waterway transportation section.

Chapter 4 In this chapter, we present a mathematical programming formulation of the speed optimization problem, which aims at minimizing the aggregated fuel consumption on an inland waterway network. The network can consist of multiple river segments, connected by a set of locks, without restrictions on the configuration. To allow scalability towards realistic waterway networks, we also propose a local-search based heuristic to optimize the speed for individual vessels. We evaluate the effectiveness of the heuristic by comparing it to the mathematical programming. For all computational experiments, we make use of real AIS data from a part of the Dutch river network. We observe that the heuristic is able to construct a high quality solution in realistic problem settings within reasonable amount of computation time.

Chapter 5 In this chapter, we consider a speed optimization problem on inland waterways, which is characterized by stochastic waiting times at the lock caused by uncertainty in lock processing time estimations of other vessels. The objective is to minimize fuel consumption of an approaching ship, such that it traverses the river segment in a set deadline. We introduce a mathematical model for this problem and evaluate the effectiveness and attractiveness of two solution approaches: an optimal solution and a simple heuristic. This creates intuitive guidelines for skippers based on information provision to select an appropriate speed decision approach to minimize the total expected fuel consumption and CO_2 emission of inland waterway transportation.

2

A cost-sharing mechanism for joint velocity optimisation

Adapted from: C. Defryn, J. A. P. Golak, A. Grigoriev, *et al.*, "Inland waterway efficiency through skipper collaboration and joint speed optimization," *European Journal of Operational Research*, 2020.

2.1 Our contributions

Previous research on lock scheduling is based on the assumption that lock operators have the full power to determine the operating schedule for the lock and operate under full information. In practice, this schedule is typically determined using the first come first serve (FIFO) principle based on the order at which vessels arrive at the lock. Skippers that know this have the incentive to speed up when approaching a lock in order to pass their predecessors and get served first. This action leads to overall longer waiting times before the locks, and increases the operational cost for these skippers due to the higher fuel consumption that is caused by maintaining a higher velocity.

In this chapter, we aim to minimize the aggregated fuel consumption by the vessels in the river, while keeping in mind that each skipper is a rational individual with the sole goal of minimizing his personal fuel cost or emissions. Our goal is to determine an optimal velocity for each individual vessel and for each river segment. The positive relation between vessel velocity and fuel consumption leads to the observation that maintaining the slowest velocity — yet meeting the arrival deadline at the destination harbour — minimizes the total fuel consumption of a single vessel. Unfortunately, even a single lock on the river becomes a source of congestion and the velocities of the vessels have to be adjusted accordingly.

The chapter is structured as follows. In Section 2.2, we model the problem as a non-cooperative game and discuss a variety of priority rules that can be used by the lock operators in case multiple vessels approach the lock (possibly in the opposite directions). Moreover, we discuss the existence of *Nash equilibria* — situations in which no skipper can unilaterally deviate from the proposed solution and decrease its individual cost. In Section 2.3, we introduce a *cooperative game* perspective on the traffic optimization problem at hand. We assume that binding contracts between different skippers are possible and propose a mechanism based on monetary payments. This situation will give rise to new Nash equilibria. We design an algorithm that computes

these Nash equilibria while minimizing total fuel consumption on the river. A MILP formulation to solve the lock scheduling problem is included in Section 2.4. Finally, in Section 2.5, we extend our algorithm to comply with an online setting.

2.2 Non-cooperative game for traffic optimization at river obstacles

2.2.1 Mathematical notation of the system

Without loss of generality, we assume a waterway with a single lock. Let this lock be defined by its capacity C, i.e., the number of vessels that can be leveled up or down simultaneously, and its current state P, indicating whether the level of the water is high (equal to the upstream level) or low (equal to the downstream level). Let T be the time to change the lock state from high to low or vice versa. If a batch of vessels is processed, an additional T_i times units are required for each vessel i in the batch. That time represents the loading and unloading of vessels and varies across different types and sizes of vessels. The processing time of a batch of vessels is the sum of the lockage duration T and the individual processing times T_i for every vessel i in the batch. Moreover, let L_u and L_d be the distances between the upstream and downstream end points of the waterway respectively and the lock. From the moment that a vessel is within that distance from the lock, we consider it to be in the system. The complete system is, therefore, determined by the tuple $L = \{C, P, T, (T_i)_{i \in S}, L_u, L_d\}$.

Now, let U and D be sets of vessels that sail upstream or downstream respectively, and let $S = U \cup D$ be the set of all vessels. The size of the entire fleet is denoted by $n = |S|$. For each vessel $i \in U$, we are given an arrival time at the downstream end point of the river, denoted by a_i, and a deadline d_i, the latest time when the vessel has to reach the downstream end point of the waterway. Similarly, a_j and d_j are defined for each vessel $j \in D$, sailing in the opposite direction.

Furthermore, we assume that vessels in set S are ordered according to their arrival times and that between any two sequential vessel arrivals at least $\varepsilon > 0$ time elapses. Finally, let $v_{i,p}$ denote the velocity of vessel i along river segment $p \in \{u, d\}$, where u and d represent the upstream and downstream segments respectively. We assume the minimum and the maximum velocity for any vessel is bounded by $v_{\min}$ and $v_{\max}$.

2.2.2 Model definition

In the game, each vessel $i \in S$ decides on $v_{i,d}$ and without loss of generality, we assume that all ships have equivalent velocity limits, i.e. $v_{i,u} \in [v_{\min}, v_{\max}]$. Furthermore, define $v_i = (v_{i,d}, v_{i,u})$. Furthermore, let v_{-i} denote the strategy profile of every player in the game except for i and let $\mathbf{v}_S = (v_i)_{i \in S}$. Note that only constant velocities have been specified for both, upstream and downstream, waterway segments. Due to the convexity of the cost function defined below, skippers will have no incentive to alter their velocity midway of the segments. The assumption of constant velocities is relaxed, when an online setting of the game is considered, in Section 2.5. To illustrate the game, consider the following example.

Example 2.2.1. *Assume three vessels (see also Figure 2.1): 1 and 2 sailing upstream and 3 sailing downstream. The waterway is 20 kilometers long, and the lock is placed in the middle of the waterway. As a result,* $L_u = L_d = 10$. *The entry/arrival times of the vessels are as follows:* $a_1 = 0$, $a_2 = \varepsilon$ *and* $a_3 = 2\varepsilon$. *Moreover, we know that* $(v_{1,u}, v_{1,d}) = (5, 5)$, $(v_{2,u}, v_{2,d}) = (10, 5)$ *and* $(v_{3,u}, v_{3,d}) = (5, 10)$. *Given the current velocities, vessel* 1 *arrives at the lock at time* 2, *vessel* 2 *at time* $1 + \varepsilon$ *and vessel* 3 *is expected to arrive at the lock at time* $1 + 2\varepsilon$.

The total fuel consumption is given by the function $F(v)$, where v represents the velocity of the vessel. The function is measured in tons per kilometer. We assume that fuel consumption is equal to zero if the vessel is not moving, i.e., its velocity is equal to zero, and vessels are

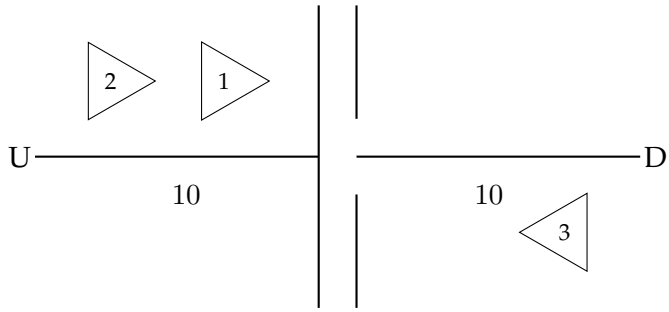


Figure 2.1: The setup of locks and vessel for Example 1

only standing still inside the lock. Following the conventions from the related literature, we assume convexity of $F(v),\ v > 0$ [12].

To further simplify notations, and without loss of generalization, we consider the fuel consumption function to be the same for every vessel and equal to

$$F_i(v_i) = L_u F(v_{i,u}) + L_d F(v_{i,d}). \tag{2.1}$$

The fuel consumption of vessels in set S can therefore be written as

$$F_{tot}(\mathbf{v}_S) = \sum_{i \in S} F_i(v_i). \tag{2.2}$$

Each skipper i aims to minimize its total fuel consumption $F_i(v_i)$, given its deadline (denoted as d_i) on the arrival time at the destination. This deadline is considered a hard constraint. Arriving at the destination after the predefined deadline is considered infeasible, represented by an infinite penalty cost. In case the deadline is unrestrictive for the vessel, it will sail at the minimum velocity $v_{\min}$. Therefore, we define the cost function for skipper $i \in S$ by

$$C_i(\mathbf{v}_S) = \begin{cases} F_i(v_i) & \text{if } a_i + L_u/v_{i,u} + L_d/v_{i,d} + q_i(\mathbf{v}_S) \leq d_i; \\ \infty & \text{otherwise,} \end{cases} \tag{2.3}$$

where $q_i(\mathbf{v}_S)$ is the total processing time of vessel i at the lock, i.e., waiting time before entering the lock plus the lock re-level time T and the individual loading times. This waiting time depends on the congestion induced by the strategy profile, i.e., individual velocities of vessels in set S.

We now define the *social cost* $C(\mathbf{v}_S)$ of a strategy profile $\mathbf{v}_S$ as the aggregated cost of all players in S, defined as

$$C(\mathbf{v}_S) = \sum_{i \in S} C_i(\mathbf{v}_S). \tag{2.4}$$

The strategy profile $\mathbf{v}_S$ that minimizes the social cost is called the *social optimum*, and has a social cost of

$$C_{opt} = \min_{\mathbf{v}_S} C(\mathbf{v}_S). \tag{2.5}$$

2.2.3 Nash equilibrium and queuing discipline at the lock

In a *non-cooperative* game (without binding contracts between the skippers), we assume that skippers act selfishly and aim to minimize their individual costs. One of the most important tools that game theorists have at their disposal is the *Nash equilibrium* [22]: a strategy profile v_S^* where no vessel can unilaterally deviate from its current strategy v_i^* and decrease its current cost. More formally, v_S^* is a Nash equilibrium if and only if

$$C_i(v_i^*, v_{-i}^*) \leq C_i(v_i, v_{-i}^*), \forall v_i \in \mathcal{V}_i. \tag{2.6}$$

The importance of the Nash equilibrium comes from the natural observation that agents/players/skippers are rather interested in selfishly minimizing their individual costs than reducing the social cost, i.e., the total cost of the entire fleet. The Nash equilibrium is calculated by minimizing the regret of the individual players, where regret is defined as the cost they could have saved by altering the strategy.

The existence of the Nash Equilibrium depends on the waiting time of vessels in front of the locks. In turn, this waiting time is subject to the *queuing discipline* of the lock. This queuing discipline dictates the order in which vessels are served by the lock operator. As the waiting time impacts the optimal (required) velocity after the lock, the queuing discipline directly affects the cost of each skipper. Therefore, different lock mechanisms yield different characteristics of the game. We consider the following three simple lock mechanisms:

Mechanism 1: Lock FIFO For any $i, j \in U \cup D$, vessel i is served by the lock before vessel j if i arrives at the lock before j. If vessels i and j arrive at the lock at the same time, i will be served first if $a_i < a_j$.

Mechanism 2: System FIFO For any $i, j \in U \cup D$, vessel i is served by the lock before vessel j if $a_i < a_j$.

Mechanism 3: System FIFO with filling idle time Consider vessel $i \in U \cup D$. Assume that skippers choose strategies sequentially and all $(v_j)_{j=1,\dots,i-1}$ are given. For any $i, j \in U \cup D$ such that $j < i$, vessel i is served before j if it does not affect the time of departure of vessel j determined by the strategy profile $(v_j)_{j=1,\dots,i-1}$. Thus, given the lock schedule for earlier arriving ships, vessel i can either join a non-full lockage in schedule or join an empty lockage, as long as this does not affect the existing schedule.

The following example illustrates how these three mechanisms work and how they affect the payoff of a strategy profile.

Example 2.2.2. *Consider again the setup of Example 2.2.1. Let us remind that the entry/arrival times of the vessels were* $a_1 = 0$, $a_2 = \varepsilon$ *and* $a_3 = 2\varepsilon$. *The lock has an infinite capacity and* $T = T_1 = T_2 = T_3 = 0.5$.. *Given the current velocities of the vessels, the arrival times at the locks are* 2, $1 + \varepsilon$ *and* $1 + 2\varepsilon$, *for vessels* 1, 2 *and* 3, *respectively.*

First, if the lock operates under Mechanism 1, only the arrival times at the lock are relevant. Note that vessel 2 *arrives at the lock first, vessel* 3 *second and vessel* 1 *is the last one. As vessels are processed in order of arrival time, the waiting times under the strategy profile are* $2 + \varepsilon, 1, 2 - \varepsilon$ *for vessel 1, 2 and 3 respectively.*

Second, under mechanism 2, only the arrival times into the system are relevant. Note that vessel 1 *arrives first in the system, vessel* 2 *second and vessel* 3 *last. The waiting times are* $1, 2 - \varepsilon, 3 - 2\varepsilon$ *for vessel 1, 2 and 3 respectively.*

Lastly, when Mechanism 3 is applied, the arrival times into the system and at the locks are relevant. Note that if vessel 2 *or* 3 *is served before vessel* 1, *the exit from the lock of vessel* 1 *would be delayed. Since vessel* 1 *arrives first into the system, it has priority and therefore it is processed first. Once vessel* 1 *is processed, the lock is open to the downstream side and vessels* 2 *and* 3 *are waiting on the upstream and downstream segments, respectively. Vessel* 2 *arrives first into the system, therefore it has priority. When vessel* 1 *has been processed, the lock is on the side of vessel* 3. *However, serving vessel* 3 *would increase the waiting time of vessel* 2 *by* 0.5. *Therefore, under this mechanism, vessel* 2 *is processed second and vessel* 3 *is processed last. The waiting times are* $1, 2 - \varepsilon, 3 - 2\varepsilon$ *for vessel 1, 2 and 3 respectively.*

Since the choice of a lock mechanism influences the behavior of vessels, it also influences the existence of Nash equilibria. Under the assumption of Mechanism 1, where the priority of vessels is determined by the arrival of vessels at the lock, a Nash equilibrium might not exist, which is shown in the following example.

Example 2.2.3 (Mechanism 1)**.** *Assume there are two vessels: vessel 1 sailing upstream and vessel 2 sailing downstream. The complete river segment is*

again 20 kilometers long, and the lock is placed in the middle of the waterway, hence, $L_u = L_d = 10$. The lock has capacity of 1 (though, any positive capacity will do) and its duration T and loading times T_1 and T_2 are set to 0.5. We assume that the fuel consumption function $F(v)$ is convex, non-negative and strictly increasing in velocities $v_{i,p} \in [5, 10],\ p \in \{u, d\}$. We assume that the lock starts on the upstream side, but can switch to the downstream side in time whenever vessel 2 is the first one to arrive at the lock. We assume the arrival times in the system are given by $a_1 = 0$ and $a_2 = \epsilon$ and the deadlines are $d_1 = 4$ and $d_2 = 4 + \epsilon$. Note only the velocity of a vessel before the lock affects the waiting time of other vessels. Therefore, for determining best responses, the strategy of the vessels can be expressed in their velocity before the lock (denoted by v_1 for vessel 1, and v_2 for vessel 2). Note that for this example, $v_{opt} = 20/3$, i.e., the optimal velocity for each vessel if it would be the only vessel on this waterway segment. We divide all possible velocity scenarios into six cases, presented in Table 2.1.

1. *Assume v_1 is equal to 10 and v_2 is any velocity in the interval $[5, v_{opt}]$. Note that vessel 1 arrives first at the lock. However, since he arrives early into the system, he can reduce his velocity to v_{opt} and still arrive first at the lock, which would reduce his costs.*

2. *Assume v_1 is equal to 10 and v_2 is any velocity in the interval $(v_{opt}, 10]$. Given the velocity of vessel 1, vessel 2 is unable to arrive earlier at the lock. Thus, his best response is to arrive after that the lock processed vessel 1, i.e. sailing at velocity 5.*

3. *Assume v_1 is in the interval $(5, 10)$ and $v_2 \leq v_1$. In this scenario, both players exhibit a racing behavior. The bets response of vessel 2 is to increases its velocity to $v_2' = v_1 + \varepsilon$. In response, vessel 1 increases its velocity to $v_1' = v_2' + \varepsilon$ and vessel 2 increases its velocity again to $v_2' = v_1' + \varepsilon$. This cycles until vessel 2 increase its velocity to 10 and the game is in a different scenario*

4. *Assume v_2 is in the interval $(5, 10)$ and $v_1 \leq v_2$. This scenario is symmetric to the previous one.*

Table 2.1: velocity scenarios for example 2.2.3.

Scenario	v_1	v_2	**Improving move**
1	10	$[5, v_{opt}]$	Player 1 should decrease v_1 to v_{opt}.
2	10	$(v_{opt}, 10]$	Player 2 should decrease v_2 to 5.
3	$(5, 10)$	$v_2 \leq v_1$	Player 2 should increase v_2 to 10.
4	$(5, 10)$	$v_2 \geq v_1$	Player 1 should increase v_1 to 10.
5	5	$(v_{opt}, 10]$	Player 2 should decrease v_2 to v_{opt}.
6	5	$[5, v_{opt}]$	Player 1 should increase v_1 to v_{opt}.

5. *Assume v_1 is equal to 5 and v_2 is in the interval $(v_{opt}, 10]$. In this scenario, vessel 2 arrives at the lock first. However, he would still arrive at the lock first if he decreases his velocity to his optimal velocity. Therefore, sailing at its optimal velocity is the best move for vessel 2.*

6. *Assume v_1 is equal to 5 and v_2 is in the interval $[5, v_{opt}]$. In this scenario, vessel 1 can sail at his optimal velocity and arrive the lock earlier than vessel 2. Thus, the optimal move for vessel 1 is to reduce his velocity to its optimal.*

We see that in every strategy profile, there is a skipper that can decrease its fuel consumption by changing its velocity. Hence, there does not exist a Nash equilibrium.

Under lock operating mechanisms 2 and 3, however, the Nash equilibrium does exist as the order in which the vessels enter the lock is determined solely by the order in which they arrive into the system. Hence, it cannot occur that vessels race each other to the lock, which is the main idea behind our previous example. Under these two mechanisms, vessels cannot affect the costs of vessels that entered the river section earlier. This implies that vessels can sequentially choose a best response, taking into account the arrival times of the previous vessels. We prove this statement more formally in the next theorem.

Theorem 2.2.4. *Consider the single lock scheduling problem, where the lock operates under Mechanism 2 or 3. Then, each game possesses at least one*

Nash equilibrium.

Proof. We provide a generic construction of a strategy profile and show that this strategy profile constitutes a Nash equilibrium. Observe that under both Mechanism 2 and 3, for any velocity v_i, the waiting time of vessel i only depends on the vessels arriving earlier in the system than vessel i. Consequently, knowing the strategies $v_1, \ldots, v_{i-1}$ is sufficient to determine optimal strategy v_i.

By construction of the strategy profile, it is apparent that each vessel i chooses its best possible strategy with respect to the vessels arriving earlier. Also, strategies of vessels that arrive later cannot influence the costs experienced by vessel i. Hence, vessel i can not decrease its private cost and therefore the resulting strategy profile is a Nash equilibrium.

Note, that the difference between the two mechanisms occurs in the individual optimization of strategies: under Mechanism 3 the waiting times caused by profile $\mathbf{v}_S$ might be different from the waiting times under Mechanism 2 using the same vector $\mathbf{v}_S$. However, the implications and the arguments stay the same: the cost for vessel i is only affected by the strategies of the first $i - 1$ vessels. □

A central authority could guarantee the existence of a Nash equilibrium by forcing the lock operators to use Mechanism 2 or Mechanism 3. However, the fact that a Nash equilibrium exists does not tell us anything about its cost efficiency. Selfish decision making may lead to a Nash equilibrium with a high social cost, which then leads to a waste of resources and high pollution on rivers. In Mechanism 2 and 3, individual costs highly depend on the strategies taken by the previous vessels. Therefore, selfish decision making may lead to the scenario in which later vessels are unable to cross the river segment before their deadline, resulting in a Nash equilibrium with an infinitely high social cost. Such scenario indicates that the *price of anarchy* of this game (the ratio between the highest social cost of any Nash equilibrium and

the minimal social cost) is unbounded. This becomes apparent in the following example.

Example 2.2.5. *We consider the same instance as in Example 2.2.3. However, this time we assume that the lock operates under Mechanism* 2*. We construct a Nash equilibrium with the procedure described in the proof of Theorem 2.2.4. This implies that* $v_1^* = (20/3, 20/3)$*. Note that there is no strategy in the strategy space of vessel* 2*, such that it passes the river segment before its deadline. Thus the social cost of this instance is infinitely high.*

There exists a strategy profile such that both vessels cross the river before their deadlines. More precisely, $\boldsymbol{v} = ((5, 10), (10, 5))$ *leads to a finite cost for both vessels. However, this strategy profile is not attained by Mechanism* 2 *nor by Mechanism* 3*. Because of this, the price of anarchy of the game at hand is unbounded. Note that the same results hold, when the lock is assumed to operate under Mechanism* 3*.*

Note that the results in the section do not rely on the assumption on equal cost functions and velocity limits. The goal of this section was to show that, though the concept of a Nash equilibrium seems appealing, in the non-cooperative setting it might not exist or it might be extremely inefficient compared to a socially optimal strategy profile. In the next section, we review the problem from a cooperative game point of view as we introduce the possibility to make binding contracts between the vessels.

2.3 Cooperative game for traffic optimization at river obstacles

We now assume that the vessels can make binding contracts and allow *payments* between skippers. As a result, the agents (skippers) can give an incentive to their counter-agents to adapt their velocities by reimbursing their extra costs. We aim to find a solution concept that is

cost optimal while making sure that no player can profit from a unilateral deviation from the social optimum. More precisely, we introduce a payment system that fulfills two criteria:

1. By participating in the payment system, the cost of a player can never be higher than when he/she did not participate.
2. The payment system should give a vessel an incentive to behave as in the social optimum.

In this section, we consider full information about the lock, river segments and vessels that will enter the system to be known in advance. An online variant of this problem is presented in Section 2.5, in which only the information about the river segment and the lock are publicly known while information about the vessels becomes only available when a vessel physically enters the waterway. Furthermore, we assume that the lock operates under mechanism 2 or 3. First, we propose an algorithm that returns for each vessel a velocity v_i, and the payment scheme $P_{i,j}$ indicating payment of skipper i to skipper j for the requested velocity adjustment. Second, we prove that the solution proposed by the algorithm satisfies the two criteria mentioned above.

2.3.1 Iterative payment scheme algorithm

The algorithm sequentially determines optimal velocities and payments in the order of vessels arrival by considering all vessels 1 through i, denoted by the set $\bar{S}_i$. In the first iteration, only vessel 1 is considered and its optimal velocity is determined. Let ζ_1 be the operating cost associated with this strategy such that $\zeta_1 = C_1(v_1)$. During future iterations, it will be ensured that the cost for this skipper will not go above the cost of this benchmark situation. To do this, other skippers should fully reimburse any cost increase that results from changing the strategy for the skipper.

Now, let $P'_{j,j'}$ be the payment scheme for all $j' < j < i$ at iteration i. Moreover, all guaranteed costs ζ_j are considered to be known for all

$j < i$. To determine the velocities v_j for all $j \in \bar{S}_i$ and payments $P_{i,j}$ for all $j < i$, we solve the following optimization problem: determine new velocities of the vessels from $\bar{S}_i$ such that the sum of the costs and payments for vessel i is minimized, while the total cost of each vessel $j < i$ is at most ζ_j. Then, we compute the value of the guaranteed cost ζ_i of player i. More formally, we define following relations.

$$P'_{i,k} := C_k(v'_{\bar{S}_i}) - \zeta_k - \sum_{j \in \bar{S}_{i-1} : j > k} P'_{jk} \;\; \forall\, k \in \bar{S}_{i-1}, \tag{2.7}$$

$$\zeta_i := C_i(v'_{\bar{S}_i}), \tag{2.8}$$

where v'_S and P' are the solutions to the following optimization problem. For a given vessel $i > 1$, having computed all optimal values P' for all $i' < i$, the mathematical program reads

$$\min_{\mathbf{v}_{\bar{S}_i}; P_{i,j}} \left(C_i(\mathbf{v}_{\bar{S}_i}) + \sum_{k \in \bar{S}_{i-1}} P_{i,k} \right) \tag{2.9}$$

subject to

$$C_k(\mathbf{v}_{\bar{S}_i}) - P_{i,k} - \sum_{\substack{j \in \bar{S}_{i-1} \\ j > k}} P'_{j,k} \leq \zeta_k, \quad \forall\, k \in \bar{S}_{i-1}. \tag{2.10}$$

Algorithm 1 represents the payment system which outputs both velocities and payments for all skippers. Note that the optimization problem has been replaced by a computation of the social optimal velocities. This is a valid substitution due to Theorem 2.3.3 below.

Algorithm 1: Payment mechanism

Input: $(L := (C, T, P, L_u, L_d), U, D, (a_i, d_i, v_{\min}, v_{\max})_{i \in U \cup D})$
Output: Optimal set of velocities and payments.

$\bar{S}_1 = \{1\}$;
$\zeta_1 = C_{opt}(S_1)$;
for *i from 2 to n* **do**
  $\bar{S}_i = \bar{S}_{i-1} \cup \{i\}$;
  Compute $C_{opt}(\bar{S}_i)$ and let $v^*_{\bar{S}_i}$ be the optimal parameters;
  $C_{opt,k}(\bar{S}_i) := C_k(v^*_{\bar{S}_i}) \ \ \forall k \in \bar{S}_i$;
  $P'_{i,k} := C_{opt,k}(\bar{S}_i) - \zeta_k - \sum_{j \in \bar{S}_{i-1} : j > k} P'_{jk} \ \ \forall\, k \in \bar{S}_{i-1}$;
  $\zeta_i := C_{opt,i}(\bar{S}_i)$;
end
return $(v^*_S, (P'_{ij})_{i,j \in S})$

The subroutine computing $C_{opt}(\bar{S}_i)$ can be implemented in various ways. In the next section, we provide a MILP-formulation to solve the lock scheduling problem to optimality. This formulation is based on the model in [12] and has been adjusted to comply with our problem statement. Moreover, we show that the problem is NP-complete

in the strong sense, this way motivating design of MILP-formulations and approximation algorithms for the problem. Note that the MILP can be extended to allow for non equivalent velocity limits and cost functions and therefore all the results generalise. Regarding existence of good approximation algorithms, we leave this question open. We stress that any α-approximation algorithm [23] directly leads to an α-approximate Nash equilibrium [24]. This follows from Theorem 4.3 and Theorem 4.4 below, in which we show that the social optimum and Nash Equilibrium coincide.

Definition 2.3.1 (α-approximation algorithm)**.** *An algorithm is considered an α-approximation algorithm for a problem if and only if for every instance of the problem it can find a solution within a factor α of the optimum solution. Let ALG be the cost of a solution provided by the algorithm, and OPT the cost of an optimal solution of the minimization problem, then*

$$ALG \leq \alpha \, OPT. \tag{2.11}$$

Definition 2.3.2 (α-approximate equilibrium)**.** *For any $\alpha \geq 1$, we define strategy $\vec{v}$ to be an α-approximate equilibrium when for every player i, and every alternative strategy $v_i' \in \mathcal{V}_i$:*

$$C_i(v_i, \vec{v}_{-i}) \leq \alpha C_i(v_i', \vec{v}_{-i}). \tag{2.12}$$

Given a solution to the optimization problem above, in theorem 2.3.3, we show that the optimal velocities in that problem are equivalent to the velocities in the social optimum computed on vessels in the set $\bar{S}_i$.

Theorem 2.3.3. $v'_{\bar{S}_i} = \operatorname{argmin}_{\boldsymbol{v}_{\bar{S}_i}} \sum_{k \in \bar{S}_i} C_k(\boldsymbol{v}_{\bar{S}_i})$ *or equivalently* $\boldsymbol{v}'_{\bar{S}_i} = \boldsymbol{v}^*_{\bar{S}_i}$

Proof. Consider the mathematical program given in equations 2.9 and 2.10. Note, that the constraint can be defined for all $k \in \bar{S}_{i-1}$:

$$C_k(\mathbf{v}_{\bar{S}_i}) - P_{i,k} - \sum_{\substack{j \in \bar{S}_{i-1} \\ j > k}} P'_{j,k} \leq \zeta_k \tag{2.13}$$

which can be rewritten as:

$$P_{i,k} \geq C_k(\mathbf{v}_{\bar{S}_i}) - \zeta_k - \sum_{\substack{j \in \bar{S}_{i-1} \\ j > k}} P'_{j,k} \tag{2.14}$$

Note the objective is to minimize and the objective function is increasing in $P_{i,k}$ for all k. Thus, the $P_{i,k}$ is as low as possible and the inequality is always binding. The program can furthermore be rewritten as follows:

$$P_{i,k} = C_k(\mathbf{v}_{\bar{S}_i}) - \zeta_k - \sum_{\substack{j \in \bar{S}_{i-1} \\ j > k}} P'_{j,k} \tag{2.15}$$

Then we can reformulate the objective function in the following way:

$$v'_{\bar{S}_i} = \underset{\mathbf{v}_{\bar{S}_i}}{\operatorname{argmin}} \left(C_i(\mathbf{v}_{\bar{S}_i}) + \sum_{k \in \bar{S}_{i-1}} P_{i,k} \right) \tag{2.16}$$

$$= \underset{\mathbf{v}_{\bar{S}_i}}{\operatorname{argmin}} \left(C_i(\mathbf{v}_{\bar{S}_i}) + \sum_{k \in \bar{S}_{i-1}} \left(C_k(\mathbf{v}_{\bar{S}_i}) - \zeta_k - \sum_{\substack{j \in \bar{S}_{i-1} \\ j>k}} P'_{j,k} \right) \right) \tag{2.17}$$

$$= \underset{\mathbf{v}_{\bar{S}_i}}{\operatorname{argmin}} \left(\sum_{k \in \bar{S}_i} C_k(\mathbf{v}_{\bar{S}_i}) - \sum_{k \in \bar{S}_{i-1}} \left(\zeta_k + \sum_{\substack{j \in \bar{S}_{i-1} \\ j>k}} P'_{j,k} \right) \right) \tag{2.18}$$

$$= \underset{\mathbf{v}_{\bar{S}_i}}{\operatorname{argmin}} \left(\sum_{k \in \bar{S}_i} C_k(\mathbf{v}_{\bar{S}_i}) \right) \tag{2.19}$$

$$= v^*_{\bar{S}_i}, \tag{2.20}$$

□

Lastly, in Theorem 2.3.4, we show that in the i-th iteration of the algorithm the best response for skipper i is to obey the payment mechanism. This means that the guaranteed cost of vessel i plus the payments this skipper has to pay to all other skippers is lower than or equal to the cost of any strategy not involving the payments.

Theorem 2.3.4. *In Algorithm 1 for each $\bar{S}_i$, it holds that*

$$\zeta_i + \sum_{k \in \bar{S}_{i-1}} P_{i,k} \leq C_i(v_i, \boldsymbol{v}^*_{\bar{S}_{i-1}}) \text{ for all } v_i \in V_i. \tag{2.21}$$

Proof. Note that by (2.13) to (2.15), we know that after every iteration i, it holds for every $k \in \bar{S}_{i-1}$

$$P'_{i,k} = C_{opt,k}(\bar{S}_i) - \zeta_k - \sum_{\substack{j \in \bar{S}_{i-1} \\ j>k}} P'_{jk} \tag{2.22}$$

which can be rewritten as

$$C_{opt,k}(\bar{S}_i) = \zeta_k + P'_{i,k} + \sum_{\substack{j \in \bar{S}_{i-1} \\ j>k}} P'_{jk} \tag{2.23}$$

$$= \zeta_k + \sum_{\substack{j \in \bar{S}_i \\ j>k}} P'_{jk} \tag{2.24}$$

This leads to the following equality.

$$\zeta_i + \sum_{k \in \bar{S}_{i-1}} P'_{ik} = C_{opt,i}(\bar{S}_i) + \sum_{k \in \bar{S}_{i-1}} \left(C_{opt,k}(\bar{S}_i) - \zeta_k - \sum_{\substack{j \in \bar{S}_{i-1} \\ j>k}} P'_{jk} \right) \tag{2.25}$$

$$= \sum_{k \in \bar{S}_i} C_{opt,k}(\bar{S}_i) - \sum_{k \in \bar{S}_{i-1}} (\zeta_k + \sum_{\substack{j \in \bar{S}_{i-1} \\ j>k}} P'_{jk}) \tag{2.26}$$

$$= \sum_{k \in \bar{S}_i} C_{opt,k}(\bar{S}_i) - \sum_{k \in \bar{S}_{i-1}} C_{opt,k}(\bar{S}_{i-1}) \tag{2.27}$$

$$= C_{opt}(\bar{S}_i) - C_{opt}(\bar{S}_{i-1}) \tag{2.28}$$

Furthermore, we know that, by definition, the social optimum of the set $\bar{S}_i$ cannot have a higher cost than the sum of the social optimum of the set $\bar{S}_{i-1}$ plus any individual strategy of vessel i. Therefore, we can formulate the following inequality.

$$C_{opt}(\bar{S}_i) \leq C_i(v_i, \mathbf{v}^*_{\bar{S}_{i-1}}) + C_{opt}(\bar{S}_{i-1}) \text{ for all } v_i \in \mathcal{V}_i \tag{2.29}$$

Then by rewriting the inequality and using the result established in equality 2.28.

$$C_{opt}(\bar{S}_i) - C_{opt}(\bar{S}_{i-1}) \leq C_i(v_i, \mathbf{v}^*_{\bar{S}_{i-1}}) \text{ for all } v_i \in \mathcal{V}_i \tag{2.30}$$

$$\zeta_i + \sum_{k \in \bar{S}_{i-1}} P'_{ik} \leq C_i(v_i, \mathbf{v}^*_{\bar{S}_{i-1}}) \text{ for all } v_i \in \mathcal{V}_i \tag{2.31}$$

□

From Theorems 2.3.3 and 2.3.4, it follows that the stated criteria for an efficient payment mechanism are fulfilled by Algorithm 1.

2.3.2 Truthfulness of the Payment Scheme

The solution concept takes deadline and cost functions of the vessels as input. For the solution to work in practice, the solution concept should fulfill the criteria that no player can profit from misreporting on their specifications. In mechanism design theory, this criteria is defined as *truthfulness* [24]. In this chapter, we show that the proposed solution is *truthful*.

Theorem 2.3.5. *Under payment scheme skippers report deadline truthfully*

Proof. Recall from 2.25 - 2.28 that

$$\zeta_i + \sum_{k \in \bar{S}_{i-1}} P'_{ik} = C_{opt}(\bar{S}_i) - C_{opt}(\bar{S}_{i-1}) \tag{2.32}$$

Assume that vessels $1 \ldots, i-1$ have reported their deadline truthfully and their payments $P_{j,k}$ and guaranteed costs ζ_j have been computed for all $j, k = 1 \ldots, i-1$. Furthermore, despite the fact that the true deadline of vessel i is d_i, he reports $\tilde{d}_i \neq d_i$. We do, however, assume that vessel i is following the velocity advice. In what follows, we show that a truthful reporting of deadline of vessel i is a best response.

Consider the case, in which vessel i extends his deadline, i.e. $\tilde{d}_i > d_i$. Due to the convexity of the the cost function, the optimal velocities are selected such that each vessel arrives exactly at its deadline at the end of the waterway. Therefore, by following the velocities, which result by the extended deadline, vessel i will arrive at the end of the waterway after his true deadline. Therefore, vessel i has infinite cost in this scenario.

Consider the case, in which vessel i shortens his deadline, i.e. $\tilde{d}_i < d_i$. Let $\tilde{\zeta}_i$, $\tilde{P}'_{ik}$ and $\tilde{C}_{opt}(\bar{S}_i)$ be the individual cost, payments of i to each vessel k and optimal cost, each under the assumption that vessel i reported $\tilde{d}_i$. In the proof of theorem 2.3.4, we know that that the individual costs for the true deadline and the shortened deadline are

$$\zeta_i + \sum_{k \in \bar{S}_{i-1}} P'_{ik} = C_{opt}(\bar{S}_i) - C_{opt}(\bar{S}_{i-1}) \tag{2.33}$$

and

$$\tilde{\zeta}_i + \sum_{k \in \bar{S}_{i-1}} \tilde{P}'_{ik} = \tilde{C}_{opt}(\bar{S}_i) - C_{opt}(\bar{S}_{i-1}) \tag{2.34}$$

respectively. However, since a shorter deadline of vessel i is more restrictive in the optimization, it has to hold that $C_{opt}(\bar{S}_i) \leq \tilde{C}_{opt}(\bar{S}_i)$. Thus, the total cost of the system can only decrease and therefore it follows

$$\zeta_i + \sum_{k \in \bar{S}_{i-1}} P'_{ik} \leq \tilde{\zeta}_i + \sum_{k \in \bar{S}_{i-1}} \tilde{P}'_{ik}. \tag{2.35}$$

Therefore, we have shown that vessel i cannot reduce its cost by reporting a non-truthful deadline. □

A similar result does not hold for reporting of non-truthful cost functions. If a vessel reports a cost function, which is steeper than his true cost function, it receives a higher payment for deviating his velocity and therefore he can reduce his cost by doing so. However, cost functions are defined in the technical manual of a vessel and therefore it is

physically impossible for skippers to lie about them.

2.4 A MILP-formulation for finding a social optimum

First, we show that even deciding on existence of a feasible solution to the optimization problem (FEASIBILITY CHECK) is strongly NP-complete. Consider an instance of the classic machine scheduling problem SEQUENCING WITH RELEASE TIMES AND DEADLINES. The problem is known to be strongly NP-complete, see e.g., [25]. Given a set J of n tasks and, for each task $j \in J$, a length $p_j \in Z^+$, a release time $r_j \in Z_0^+$, and a deadline $d_j \in Z^+$, the question is whether there exists a one-processor schedule for J that satisfies the release time constraints and meets all the deadlines. We reduce SEQUENCING WITH RELEASE TIMES AND DEADLINES to FEASIBILITY CHECK. Given an instance of SEQUENCING WITH RELEASE TIMES AND DEADLINES, we create an instance of FEASIBILITY CHECK as follows. Let each job be represented by a vessel with $a_j = r_j$, $d_j = d_j$ and $T_j = p_j$. Furthermore, let set U contain all vessels, set minimum velocity to 0 and let the maximum velocity be unbounded. Next, set capacity to 1 and lockage duration equal to 0. Clearly, an instance of SEQUENCING WITH RELEASE TIMES AND DEADLINES is a yes-instance if and only if the corresponding instance of FEASIBILITY CHECK is a yes-instance.

Next, we describe a MILP-formulation that can be used to compute a social optimum. This formulation is an adjustment of the model from [12]. In their paper, the authors propose a model to minimize emission on a waterway with multiple locks. Our program differs from that model in a few ways. First of all, it is a bit simpler as we solve a single lock scheduling problem. On the other hand, our model takes into account a need for compensation for the lost time in case of a slow velocity towards the lock, and/or for a high velocity towards the lock. We give a brief overview of the model. For more details, we refer the interested reader to [12].

We introduce the variables $\bar{v}_{i,p} = \frac{1}{v_{i,p}}$ and let $\bar{E}(\bar{v})$ express the emissions as a function of the reciprocal of vessel velocity, $\bar{v}_{i,p}$. To enable the usage of MILP, we use a piece-wise linear approximation of $\bar{E}(\bar{v})$. The decision variables in the model are based on possible lockages. It is clear that for each lock, the number of lockages in the optimal solution does not need to be greater than double the number of ships. Thus, the upper bound for the number of lockages is defined as $K = 2|S|$. The set of possible lockages is defined as $\mathcal{K} = \{1, \ldots, K\}$. In addition to variables $\bar{v}_i$, we introduce for all $k \in \mathcal{K}$ t_k as the starting time of the k'th lockage and for each $i \in S$ and $k \in \mathcal{K}$ we define:

$$z_{i,k} = \begin{cases} 1, & \text{if vessel } i \text{ is handled by the } k\text{th lock movement.} \\ 0, & \text{otherwise} \end{cases}$$

The mathematical programming formulation, presented below, returns an optimal strategy profile for a set of vessels.

$$\min \sum_{i \in S} \bar{E}(\bar{v}_i) \tag{2.36}$$

subject to

$$A_i \leq t_k - l_d \bar{v}_{i,d} + M_i^{A,u}(1 - \sum_{\kappa=0}^{k} z_{i,\kappa}), \quad \forall i \in U, k \in \mathcal{K} \tag{2.37}$$

$$A_i \leq t_k - l_d \bar{v}_{i,u} + M_i^{A,d}(1 - \sum_{\kappa=0}^{k} z_{i,\kappa}), \quad \forall i \in D, k \in \mathcal{K} \tag{2.38}$$

$$D_i \geq t_k + T + \sum_{j \in S} z_{j,k} T_j + l_d \bar{v}_{i,u} - M_i^{D,u}(1 - \sum_{\kappa=k}^{K} z_{i,\kappa}), \quad \forall i \in U, k \in \mathcal{K} \tag{2.39}$$

$$D_i \geq t_k + T + \sum_{j \in S} z_{j,k} T_j + l_u \bar{v}_{i,d} - M_i^{D,d}(1 - \sum_{\kappa=k}^{K} z_{i,\kappa}), \quad \forall i \in D, k \in \mathcal{K} \tag{2.40}$$

$$\sum_{k \in \mathcal{K}} z_{i,k} = 1, \quad \forall i \in S \tag{2.41}$$

$$t_k \geq t_{k-1} + T + \sum_{j \in S} z_{j,k-1} T_j, \quad \forall k \in \mathcal{K} \setminus \{1\} \tag{2.42}$$

$$z_{i,k} + z_{j,k} \leq 1, \quad \forall i \in U, j \in D, k \in \mathcal{K} \tag{2.43}$$

$$z_{i,k-1} + z_{j,k} \leq 1, \quad \forall i, j \in U, k \in \mathcal{K} \setminus \{1\} \tag{2.44}$$

$$z_{i,k-1} + z_{j,k} \leq 1, \quad \forall i, j \in D, k \in \mathcal{K} \setminus \{1\} \tag{2.45}$$

$$\sum_{i \in S} z_{i,k} \leq C, \quad \forall k \in \mathcal{K} \tag{2.46}$$

$$1/v^{max} \leq \bar{v}_{i,p} \leq 1/v^{min}, \quad \forall i \in S, p \in \{u, p\} \tag{2.47}$$

$$z_{i,k} \in \{0, 1\}, \quad \forall i \in S, k \in \mathcal{K} \tag{2.48}$$

$$t_k \in R_+, \quad \forall k \in \mathcal{K} \tag{2.49}$$

The objective function (2.36) minimizes the aggregated costs of the individual vessels. Constraints (2.37) and (2.38) ensure that vessels arrive at the lock before their respective lockage time has started and constraints (2.39) and (2.40) ensure that vessels are leaving the system before their deadline. Constraints (2.39) - (2.42) are based on values $M^{A,u}$,$M^{A,d}$,$M^{D,u}$ and $M^{D,d}$, which are large constants specified in [12]

Constraints (2.41) to (2.46) are used to model the working of the lockages. Thus, (2.41) ensures that each vessel is scheduled on exactly one lockage, while (2.42) ensures that difference between any two starting times of lockages is at least the lockage duration, $T + \sum_{i \in S} T_i$. Constraint (2.43) guarantees that a lockage does not contain vessels coming from opposing sides of the river. Furthermore, the requirement that lockages are alternating between opposing sides is ensured by constraints (2.44) and (2.45). Finally, constraint (2.46) restricts lockages to

carry only as many vessels as specified by the capacity, C.

2.5 Online setting

The assumption of perfect information on arrival times is likely to be violated in real-life. That is, there is no information prior to the arrival of the vessels at the boundaries of the system. Each time a vessel enters, the optimal velocity and payments are recomputed taking into account the location of the vessels already present on the waterway. Note that the definition of a social optimum and a best response of a player are dependent on the information setting of the game. Therefore, we have to dynamically redefine/adjust these quantities in an online setting.

Let the distance between vessel i and the exit of the waterway at time t be denoted by h_i^t. Furthermore, define $\mathbf{h}_S = (h_i^t)_{i \in S}$. The best response of vessel i, given the strategies of the other vessels, is defined as the strategy that minimizes the cost of vessel i conditional on the position of the other vessels at time t. The cost of vessel i under strategy profile $\mathbf{v}$ conditional on the position of all vessels in set $\bar{S}$ at time t is denoted as $C_i(\mathbf{v}_{\bar{S}}|\mathbf{h}_{\bar{S}}^t)$. The social optimum is defined as a strategy profile, which provides the lowest possible cost given the positions of vessels in $\bar{S}$ at time t.

Similar to the offline setting, the algorithm sequentially determines optimal velocities and payments at the arrival of each vessel. In each iteration, a set $\bar{S}_i$ is constructed containing all vessels currently in the system. Assume that vessel i arrives and vessels in $\bar{S}$ have not left the waterway yet. Moreover, the payments $P'_{j,j'}$ for all $k \leq j' < j < i$ and the guaranteed costs ζ_j for all $k \leq j < i$ are assumed to be given. Since each vessel is at a different position, payments and costs are normalized to units per kilometers. Therefore, we solve the following optimization problem: determine new velocities of the vessels from $\bar{S}_i$ such that the sum of the costs and payments for vessel i is minimized,

while the normalized total cost of each vessel $k \leq j < i$ is at most the normalized guaranteed cost. Given the following relations

$$C^{a_i}_{opt,k}(\bar{S}_i) = C_k(\mathbf{v}'_{\bar{S}_i}|\mathbf{h}^{a_i}_{\bar{S}_i}) \;\; \forall\, k \in \bar{S}_i, \tag{2.50}$$

$$P'_{i,k} := C^{a_i}_{opt,k}(\bar{S}_i) - h^{a_i}_k \Big(\sum_{\substack{j \in \bar{S}_i \setminus \{i\} \\ j>k}} \left(\frac{P'_{j,k}}{h^{a_j}_k} \right) + \frac{\zeta_k}{l_d + l_u} \Big) \;\; \forall\, k \in \bar{S}_i \setminus \{i\}\,, \tag{2.51}$$

$$\zeta_i := C^{a_i}_{opt,i}(\bar{S}_i) \tag{2.52}$$

we define the online optimization problem as

$$min_{\mathbf{v}_{\bar{S}_i}: P_{i,j}} \left(C_i(\mathbf{v}_{\bar{S}_i}|\mathbf{h}^{a_i}_{\bar{S}_i}) + \sum_{k \in \bar{S}_i \setminus \{i\}} P_{i,k} \right) \tag{2.53}$$

subject to

$$\frac{C_k(\mathbf{v}_{\bar{S}_i}|\mathbf{h}^{a_i}_{\bar{S}_i})}{h^{a_i}_k} - \frac{P_{i,k}}{h^{a_i}_k} - \sum_{\substack{j \in \bar{S}_i \setminus \{i\} \\ j>k}} \frac{P'_{j,k}}{h^{a_j}_k} \leq \frac{\zeta_k}{l_d + l_u} \;\; \forall\, k \in \bar{S}_i \setminus \{i\}\,. \tag{2.54}$$

Again, it can be shown that the two conditions for an efficient payment mechanism are fulfilled in the online setting. The proof is similar to the one discussed in the offline case. The resulting algorithm is given in Algorithm 2.

Theorem 2.5.1 below shows that the strategy profile is again equivalent to the social optimum, conditionally on the position of players at arrival time of vessel i. Thus, the optimal strategy profile is an output of Algorithm 2.

Theorem 2.5.1. $v'_{\bar{S}_i} = \text{argmin}_{(\boldsymbol{v}_{\bar{S}_i}} \sum_{k \in \bar{S}_i} C_k(\boldsymbol{v}_{\bar{S}_i}|\boldsymbol{h}^{a_i}_{\bar{S}_i})$ *or equivalently* $\boldsymbol{v}'_{\bar{S}_i} = \boldsymbol{v}^*_{\bar{S}_i}$

Proof. Consider the mathematical program given in equations 2.53 and 2.54. Note, that the constraint can be defined for all $k \in \bar{S}_{\setminus} \{i\}$:

$$\frac{C_k(\mathbf{v}_{\bar{S}_i}|\mathbf{h}^{a_i}_{\bar{S}_i})}{h^{a_i}_k} - \frac{P_{i,k}}{h^{a_i}_k} - \sum_{\substack{j \in \bar{S}_i \setminus \{i\} \\ j>k}} \frac{P'_{j,k}}{h^{a_j}_k} \leq \frac{\zeta_k}{l_d + l_u} \tag{2.55}$$

which can be rewritten as:

$$P_{i,k} \geq C_k(\mathbf{v}_{\bar{S}_i}|\mathbf{h}^{a_i}_{\bar{S}_i}) - h^{a_i}_k \Big(\sum_{\substack{j \in \bar{S}_i \setminus \{i\} \\ j>k}} \left(\frac{P^*_{j,k}}{h^{a_j}_k} \right) + \frac{\zeta_k}{l_d + l_u} \Big) \tag{2.56}$$

Note that with the same argument as in theorem 2.3.4, the constraint must be binding:

$$P_{i,k} = C_k(\mathbf{v}_{\bar{S}_i}|\mathbf{h}^{a_i}_{\bar{S}_i}) - h^{a_i}_k \Big(\sum_{\substack{j \in \bar{S}_i \setminus \{i\} \\ j>k}} \left(\frac{P^*_{j,k}}{h^{a_j}_k} \right) + \frac{\zeta_k}{l_d + l_u} \Big) \tag{2.57}$$

Then we can reformulate the objective function in the following way.

$$v'_{\bar{S}_i} \in \underset{\mathbf{v}_{\bar{S}_i}:P_{i,j}}{\operatorname{argmin}} \left(C_i(\mathbf{v}_{\bar{S}_i}|\mathbf{h}^{a_i}_{\bar{S}_i}) + \sum_{k\in\bar{S}_i\setminus\{i\}} P_{i,k} \right) \tag{2.58}$$

$$\in \underset{\mathbf{v}_{\bar{S}_i}:P_{i,j}}{\operatorname{argmin}} \left(C_i(\mathbf{v}_{\bar{S}_i}|\mathbf{h}^{a_i}_{\bar{S}_i}) + \sum_{k\in\bar{S}_i\setminus\{i\}} C_k(\mathbf{v}_{\bar{S}_i}|\mathbf{h}^{a_i}_{\bar{S}_i}) - h^{a_i}_k\Big(\sum_{\substack{j\in\bar{S}_i\setminus\{i\}\\ j>k}} \left(\frac{P'_{j,k}}{h^{a_j}_k}\right) + \frac{\zeta_k}{l_d+l_u}\Big) \right) \tag{2.59}$$

$$\in \underset{\mathbf{v}_{\bar{S}_i}:P_{i,j}}{\operatorname{argmin}} \left(\sum_{k\in\bar{S}_i} \Big(C_k(v_{j\in\bar{S}_i}|\mathbf{h}^{a_i}_{\bar{S}_i})\Big) - \sum_{k\in\bar{S}_i\setminus\{i\}} h^{a_i}_k\Big(\sum_{\substack{j\in\bar{S}_{i-1}\\ j>k}} \left(\frac{P'_{j,k}}{h^{a_j}_k}\right) + \frac{\zeta_k}{l_d+l_u}\Big) \right) \tag{2.60}$$

$$\in \underset{\mathbf{v}_{\bar{S}_i}:P_{i,j}}{\operatorname{argmin}} \left(\sum_{k\in\bar{S}_i} \Big(C_k(\mathbf{v}_{\bar{S}_i}|\mathbf{h}^{a_i}_{\bar{S}_i})\Big) \right) \tag{2.61}$$

$$= \mathbf{v}^*_{\bar{S}_i} \tag{2.62}$$

□

Likewise, Theorem 2.5.2 shows that it is the best response for vessel i to obey the payment mechanism.

Theorem 2.5.2. *In Algorithm 2 for each $\bar{S}_i$, it holds that*

$$\zeta_i + \sum_{k\in\bar{S}\setminus\{i\}} P_{i,k} \leq C_i(\boldsymbol{v}_i, v^*_{\bar{S}\setminus\{i\}}|\boldsymbol{h}^{a_i}_{\bar{S}_i}) \text{ for all } v_i \in V_i. \tag{2.63}$$

Proof. Note that by 2.55 to 2.55, we know that after every iteration i, it holds for every $k \in \bar{S}_i \setminus \{i\}$

$$P'_{i,k} = C^{a_i}_{opt,k}(\bar{S}_i) - h^{a_i}_k\Big(\sum_{\substack{j\in\bar{S}_i\setminus\{i\}\\ j>k}}\left(\frac{P'_{j,k}}{h^{a_j}_k}\right) + \frac{\zeta_k}{l_d+l_u}\Big) \tag{2.64}$$

which can be rewritten as

$$C^{a_i}_{opt,k}(\bar{S}_i) = P'_{i,k} + h^{a_i}_k\Big(\sum_{\substack{j\in\bar{S}_i\setminus\{i\}\\ j>k}}\left(\frac{P'_{j,k}}{h^{a_j}_k}\right) + \frac{\zeta_k}{l_d+l_u}\Big) \tag{2.65}$$

$$= h^{a_i}_k\Big(\sum_{\substack{j\in\bar{S}_i\\ j>k}}\left(\frac{P^*_{j,k}}{h^{a_j}_k}\right) + \frac{\zeta_k}{l_d+l_u}\Big) \tag{2.66}$$

This leads to the following equality.

$$\begin{aligned}
&\zeta_i + \sum_{k\in\bar{S}\setminus\{i\}} P'_{i,k} \\
&= C^{a_i}_{opt,i}(\bar{S}_i) + \sum_{k\in\bar{S}\setminus\{i\}}\left(C^{a_i}_{opt,k}(\bar{S}_i) - h^{a_i}_k\Big(\sum_{\substack{j\in\bar{S}_i\setminus\{i\}\\ j>k}}\Big(\frac{P'_{j,k}}{h^{a_j}_k}\Big) + \frac{\zeta_k}{l_d+l_u}\Big)\right) && (2.67)\\
&= \sum_{k\in\bar{S}} C^{a_i}_{opt,k}(\bar{S}_i) - \sum_{k\in\bar{S}\setminus\{i\}}\left(h^{a_i}_k\Big(\sum_{\substack{j\in\bar{S}_i\setminus\{i\}\\ j>k}}\Big(\frac{P'_{j,k}}{h^{a_j}_k}\Big) + \frac{\zeta_k}{l_d+l_u}\Big)\right) && (2.68)\\
&= \sum_{k\in\bar{S}} C^{a_i}_{opt,k}(\bar{S}_i) - \sum_{k\in\bar{S}\setminus\{i\}} C^{a_i}_{opt,k}(\bar{S}_i\setminus\{i\}) && (2.69)\\
&= C^{a_i}_{opt}(\bar{S}_i) - C^{a_i}_{opt}(\bar{S}_i\setminus\{i\}) && (2.70)
\end{aligned}$$

Furthermore, we know that, by definition, the social optimum of the set $\bar{S}_i$ cannot have a higher cost than the sum of the social optimum of

the set $\bar{S}_i \setminus \{i\}$ plus any individual strategy of vessel i. Therefore, we can formulate the following inequality.

$$C_{opt}^{a_i}(\bar{S}_i) \leq C_i(v_i, v^*_{\bar{S}_i \setminus \{i\}}) + C_{opt}^{a_i}(\bar{S}_i \setminus \{i\}) \text{ for all } v_i \in \mathcal{V}_i \tag{2.71}$$

Then by rewriting the inequality and using the result established in 2.67 - 2.70

$$C_{opt}^{a_i}(\bar{S}_i) - C_{opt}^{a_i}(\bar{S}_i \setminus \{i\}) \leq C_i(v_i, \mathbf{v}^*_{\bar{S}_{i-1}}) \text{ for all } v_i \in \mathcal{V}_i \tag{2.72}$$

$$\zeta_i + \sum_{k \in \bar{S}_{i-1}} P^*_{ik} \leq C_i(v_i, \mathbf{v}^*_{\bar{S}_{i-1}}) \text{ for all } v_i \in \mathcal{V}_i \tag{2.73}$$

□

Algorithm 2: Payment mechanism Online Setting

Input: $(L := (C, T, P, L_u, L_d), U, D, (a_i, d_i, v_{\min}, v_{\max})_{i \in U \cup D})$

Output: Optimal set of velocities and payments.

Vessel i arrives in the system at time a_i:

For each vessel currently present in the waterway, update the distance to the destination as follows:;

$h_i^{a_i} = l_d$ if $i \in U$;

$h_i^{a_i} = l_u$ if $i \in D$;

$h_j^{a_i} = h_j^{a_{i-1}} - v_j^*(a_i - a_{i-1})$ for $j \in S_{i-1}$;

Let $\bar{S}_i$ be the set of vessels in the waterway at time a_i as follows:;

$\bar{S}_i = \emptyset$ if $i = 0$;

$\bar{S}_i = \bar{S}_{i-1} \setminus \left\{ j \in \bar{S}_{i-1} | h_j^{a_i} \leq 0 \right\}$;

if $\bar{S}_i \neq \emptyset$ **then**

  Compute $C_{opt}^{a_i}(\bar{S}_i)$ and let $\mathbf{v}_{\bar{S}_i}^*$ be the optimal parameters;

  $C_{opt,k}^{a_i}(\bar{S}_i) = C_k(v_{\bar{S}_i}^* | \mathbf{h}_{\bar{S}_i}^{a_i}) \ \forall\, k \in \bar{S}_i$;

  $P_{i,k}^* := C_{opt,k}^{a_i}(\bar{S}_i)$

  $$- h_k^{a_i} \Big(\sum_{\substack{j \in \bar{S}_i \setminus \{i\} \\ j > k}} \left(\frac{P_{j,k}^*}{h_k^{a_j}} \right) + \frac{\zeta_k}{l_d + l_u} \Big), \ \forall\, k \in \bar{S}_i \setminus \{i\} \, ;$$

  $\zeta_i := C_{opt,i}^{a_i}(\bar{S}_i)$;

else

  $\zeta_i = \min_{v_i} C_i(v_i)$;

end

$\bar{S}_i = \bar{S}_{i-1} \cup \{i\}$;

2.6 Future work

Note that under iterative payment scheme, whenever a vessel enters the system, its total fuel cost and payments to the other vessels become known, and will not change anymore. Hence, the lock operator can also serve as a bank: whenever a vessel crosses the lock, it pays (or receives) the payments. This implies that the lock operator needs a cash reserve, as it is likely that the first vessels entering the lock receive money from the vessels that did not arrive at the lock yet. Clearly, this cash reserve needs to be at most the cost of an optimal profile minus the minimum fuel cost of all earlier vessels. An interesting open question arises: what is the minimum amount of cash reserves needed to cover all payments without any risk of bankruptcy.

Furthermore, we assume that missing the deadline results in a infinite cost for skippers. A reasonable assumption is to assume that a delay results in a specified cost. Therefore, an interesting - but probably challenging - question would be to know if similar results apply if we the assumptions on delayed skippers changes.

3

Periodic Lock Scheduling Problem

Adapted from: J. Golak, A. Grigoriev, F. van Lent, *et al.*, "Periodic lock scheduling problem," *Working Paper*, 2021.

3.1 Introduction

Generally, the lock management (i.e., the scheduling of all lock operations) is done based upon intuition and ad-hoc decision making. This typically results in a first-come first-serve queuing policy, which creates uncertainty for the skipper regarding their eventual arrival times. This uncertainty leads to inefficient speeding behavior by preemptively speeding and overtaking[1].

Regulation on operations and arrival time of vessels reduces uncertainty and would result into a more efficient inland waterway transportation. In this work, we investigate methods for creating a periodic schedule for lock operations. The aim is to find repeating pattern of operations that minimize the waiting time for incoming streams of vessels. Each stream of vessel is defined by an arrival pattern for each different types of vessel. Given a number of streams and a periodic schedule, the skippers would have a certain arrival and departure for a respective lock and can adjust their speed accordingly. In this work, we provide solutions for the periodic lock scheduling problem and theoretically derive behavior of these solutions. These results will give policy makers and practitioners tools to further reduce emission of the transportation sector.

Related Work To our knowledge, the combination of periodic and lock scheduling has not been addressed in literature. However, several publications have been made on periodic maintenance problem. In [27], the authors discuss the free periodic maintenance problem (FPMP), a variant of the periodic maintenance problem where the cycle length T is a decision variable, for m machines without servicing costs. The costs of operating a machine at a given time linearly depends on the time since its last maintenance. They prove that there exists an optimal schedule that is cyclic and provide an exact, network-flow based

[1] Communicated to us by our industrial partner Trapps Wise. B.V.

algorithm with exponential complexity. Additionally, an approximation is introduced which even provides an optimal solution for the 2 machine setting. The authors of [28] formulate a generalization of this problem, show that the feasibility problem is NP-hard. Additionally, they develop a near-optimal solution using non-linear programming and provide several approximation algorithms - one of which is a 1.57-approximation.

Furthermore, the perfect periodic scheduling problem has been focus of many studies. [29] proposes method for finding a feasible schedule for three products with three distinct periodicities. In [30], the results have been extended by deriving a $O(n^4)$ test for the existence of a feasible schedule, and a method of constructing a feasible schedule if one exists. In addition, [31] and [32] extend the perfect periodic scheduling problem in the context of Economic Lot Scheduling. Due to the relevant applications and the complexity of perfect periodic scheduling, several heuristics have been proposed, see [33] and [34].

Our contribution In this work, we propose the *periodic lock scheduling problem*. We develop solution concepts and provide theoretical behavior of these solutions. The paper is structured as follows. In Section 3.2 we introduce the mathematical framework and optimization model. In Section 3.3 we present an closed-form solution for the case of two streams of vessels. In Sections 5.3 and Sections 3.5 we tackle the general case and propose a exact solution method and an approximation algorithm.

3.2 Problem Statement

We consider an infinite horizon, discrete time scheduling problem of n incoming streams of vessels, $\mathcal{I} = I_1, \ldots, I_n$ on a single lock. A stream is defined by the following properties:

- Periodicity of vessels of stream i arriving at the lock, $\lambda_i \geq 2$,

- Time of first occurrence of a vessel of stream i, b_i. Without loss of generality, we assume for all streams i that $b_1 = 0$ and $0 \leq b_i < \lambda_i$ for $i > 1$, and
- The direction d_i, indicating whether vessels of stream i arrive from upstream, U, or downstream, D.

Furthermore, we define the set of arrival times of stream i, by $A_i = \{b_i + k\lambda_i \mid k \in \mathbb{N}\}$. The arrival sequence a_i, indicating whether a vessel of stream i has appeared at time t, is then constructed as follows

$$a_{i,t} = \begin{cases} 1 & \text{if } t \in A_i, \\ 0 & \text{if } t \notin A_i. \end{cases} \tag{3.1}$$

Thus, each stream is completely defined by the tuple $I_i = (\lambda_i, b_i, d_i, a_i)$.

The processing time of the lock is the sum of the time needed to change the lock state from high to low (or vice versa) and the duration of loading and unloading the lock chamber. We assume that the time space is discrete, such that the processing time is exactly one time unit. Furthermore, we assume that this lock is uncapacitated, i.e. it can level up or down any number of vessels simultaneously.

At each time period, the lock operator has three possible actions

- D: move the lock from downstream to upstream, while processing a batch of vessels, coming from downstream,
- U: move the lock upstream to downstream, while processing a batch of vessels, coming from upstream,
- W: wait at current position.

A *policy* σ, is a sequence $\sigma = \sigma_0, \sigma_1, \ldots$ where $\sigma_t \in D, U, W$ denotes the action of the lock operator during period t. A policy is *cyclic* if it consists of repetitions of a finite sequence, i.e. $\sigma_t = \sigma_{t+T}$ for some T and all t. We defined the base sequence by $S(\sigma) = \sigma_0, \ldots, \sigma_{T-1}$ and its length by $L(\sigma)$.

At each time step, we define the number of vessels arriving from upstream by $a_t^U = \sum_{i \in \mathcal{I}, d_i = U} a_{i,t}$, from downstream by $a_t^D = \sum_{i \in \mathcal{I}, d_i = D} a_{i,t}$, and the number of total arrivals by $a_t = a_t^U + a_t^D$. Given time t, let τ_U and τ_D be the time units elapsed since the last scheduled U/D operation. Given a policy σ, we define the waiting time accumulated at time t, when scheduling operation σ_t as

$$C_t(\sigma_t) = \begin{cases} \sum_{i=0}^{\tau_U} a_{t-i}^U & \text{if } \sigma_t = U, \\ \sum_{i=0}^{\tau_D} a_{t-i}^D & \text{if } \sigma_t = D, \\ 0 & \text{if } \sigma_t = W. \end{cases} \tag{3.2}$$

The objective function is then defined by the long run average cost of a policy σ, i.e.

$$C(\sigma) = \lim_{t \to \infty} \frac{1}{t} \sum_{i=0}^{t} C_i(\sigma_i).$$

A policy σ is *optimal* if it minimizes $C(\sigma)$.

3.3 2-Stream Problem

We now solve the defined problem for two streams. Note that for the general problem the encoding length is $O(n \log(\max \lambda_i))$. Given the short input, we expect a small output from a complexity point of view. When restricting the problem to only two streams, we propose a closed-form formula to create a feasible and optimal policy. The solution is therefore created in an instant, and is thus polynomial in the size of the input. Without loss of generality, we assume throughout this section that there is one stream of vessels incoming from upstream and one coming from downstream. They are defined by λ_u, b_u and λ_d, b_d, respectively, where $\lambda_u \neq \lambda_d$.

Definition 3.3.1 (Collision). *We consider time period t to be a* Collision *if a vessel arrives from both upstream and downstream, i.e. $a_u = a_d = 1$.*

Lemma 3.3.1. *Given λ_u and λ_d, there is at most one collision every $T = lcm(\lambda_u, \lambda_d)$ time steps.*

Proof. Let t be the time point of the first collision. If t does not exist, the statement holds. Assume t exist, each collision occurs when $t \mod k_u\lambda_u = 0$ and $t \mod k_u\lambda_d = 0$. Thus, a collision occurs exactly when t is dividable by λ_u and λ_d. This implies that a collision occurs every $T = lcm(\lambda_u, \lambda_d)$ time steps. □

Definition 3.3.2 (Cyclic Priority Rule). *Given the arrival sequences A_u and A_d, we define the cyclic rule with priority u as*

$$\tau_u(t) = \begin{cases} U & \text{if } t \in A_u \text{ and } t \notin A_d \\ D & \text{if } t \in A_d \text{ and } t \notin A_u \\ U & \text{if } t \in A_u \text{ and } t \in A_d \\ D & \text{if } t-1 \in A_u \text{ and } t-1 \in A_d \\ W & \text{otherwise} \end{cases}$$

symmetrically, we define the cyclic policy with priority d as

$$\tau_d(t) = \begin{cases} U & \text{if } t \in A_d \text{ and } t \notin A_u \\ D & \text{if } t \in A_u \text{ and } t \notin A_d \\ D & \text{if } t \in A_u \text{ and } t \in A_d \\ U & \text{if } t-1 \in A_u \text{ and } t-1 \in A_d \\ W & \text{otherwise} \end{cases}$$

Remark 1. *Given the cycling priority rules τ_u and τ_d, it is possible to construct a feasible cyclic policy by following either rule.*

Example 3.3.3.

$$\lambda_u = 3, \lambda_d = 4, b_U = b_D = 0$$

$$a_u = 0, 3, 6, 9, \ldots$$

$$a_d = 0, 4, 8, \ldots$$

Then the following base sequence for a policy σ satisfies the cyclic priority rule τ_u,

$$S(\sigma) = U, D, W, U, D, W, U, D, W, U, W, D.$$

Lemma 3.3.2. *If $\lambda_u = 2$ and $\lambda_d > 2$, then any optimal cyclic policy satisfies τ_u. Similarly, if $\lambda_u > 2$ and $\lambda_d = 2$, then any optimal cyclic policy satisfies τ_d.*

Proof. Assume that $\lambda_u = 2$, $\lambda_d > 2$ and that σ is a cyclic policy satisfying τ_u. If there is no collision, σ has no cost and is therefore optimal.

Assume there is collision at time t, then we know that

$$(a_u(t), a_u(t+1), a_u(t+2)) = (1, 0, 1),$$

$$(a_d(t), a_d(t+1), a_d(t+2)) = (1, 0, 0),$$

$$(\sigma(t), \sigma(t+1), \sigma(t+2) = (U, D, U).$$

Thus, at $t + 3$ the lock is on the downstream side and can process any incoming vessel. Thus, σ has a waiting time of one at each collision and no waiting time at any other time period. This implies that σ is optimal.

Similarly, the symmetric case follows. □

Lemma 3.3.3. *If $\lambda_d > 2$ and $\lambda_u > 2$, then any cyclic policy satisfying τ_u and τ_d is optimal.*

Proof. Assume P_u and P_d are cyclic policies satisfying τ_u and τ_d, respectively. If there is no collision, clearly $C(P_u) = C(P_d) = 0$ and therefore either is optimal.

Assume there is an overlap at time t, then we know that

$$(a_u(t), a_u(t+1), a_u(t+2)) = (1, 0, 0),$$

$$(a_d(t), a_d(t+1), a_d(t+2)) = (1, 0, 0),$$

Since one ship is arriving from either side at t, by assumption no ship arrives at $t+1$ and no ship arrives at $t+2$. Thus, there is enough time to change the lock to either side at position $t+3$. Consequently, P_u and P_d both have a waiting time of one per collision and are thus optimal □

Theorem 3.3.4. *The Cyclic Priority Rule gives a a closed-form formula that creates a feasible and optimal policy in time polynomial in the input size.*

Proof. Follows from Lemmas 3.3.2 and 3.3.3. □

3.4 Dynamic Programming Solution

3.4.1 States and Existence of a cyclic policy

Lemma 3.4.1. *The arrival sequence of vessels is cycling, i.e.* $(a_t^U, a_t^D) = (a_{t+T}^U, a_{t+T}^D)$ *where* $T = lcm(\lambda_1, \lambda_2, \ldots, \lambda_n)$

Proof. Note that for every stream it holds that $a_{1,t} = a_{1,t+\lambda_i}$ for all t. Let $T = lcm(\lambda_1, \lambda_2, \ldots, \lambda_n)$, then it holds for all i and for all t, $a_{1,t} = a_{1,t+T}$. By definition, this implies that $(a_t^U, a_t^D) = (a_{t+T}^U, a_{t+T}^D)$. □

Given that the lock is at state $p \in \{U, D\}$. Let ω_p denote the number of waiting operations W on side p since the last time the lock switched from state $|1-p|$ to the current state p. Let $\omega_{|1-p|}$ denote the number of waiting operations W on side $|1-p|$ since the last time the lock

switched from state p to state $|1-p|$. Given a policy P and a time t, the state of the system can be represented by $S_t = (\omega_p, \omega_{|1-p|}, p)$.

Lemma 3.4.2. *Any policy σ with two consecutive time periods of waiting can be replaced by a σ' without two consecutive time periods of waiting, such that $C(\sigma') \leq C(\sigma)$.*

Proof. Assume we have a policy σ, where $\sigma_t, \sigma_{t+1} = W$ for some t and, without loss of generality, assume that $\sigma_{t-1} = U$, $\sigma_{t+2} = D$. Let σ' be defined by $\sigma'_t = D$, $\sigma'_{t+1} = U$ and $\sigma'_{t'} = \sigma_{t'}$ for $t' \neq t, t+1$. Clearly, σ' is a feasible policy and $C(\sigma') \leq C(\sigma)$. □

It follows that $\omega_p, \omega_{|1-p|} \in \{0, 1\}$. The state space at time t can then be represented by a directed graph. Let there be a node for each state and if one state can be reached in the next time period from another state, let there be a directed edge between then. The state space graph is visualised in Figure 3.1.

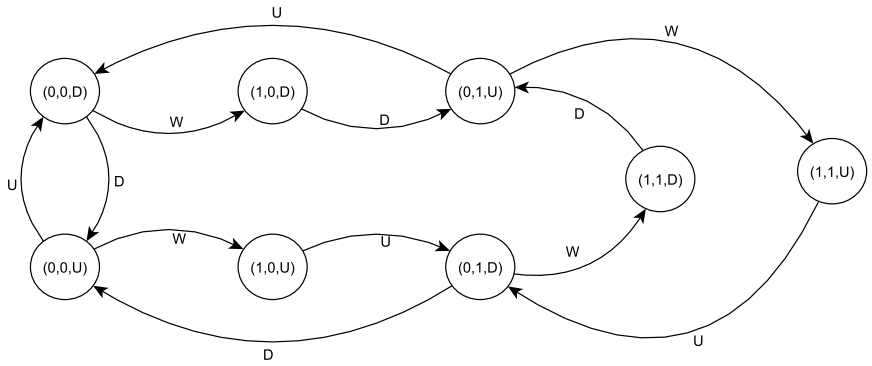


Figure 3.1: State space graph

Lemma 3.4.3. *There exists an optimal policy σ^* that is cyclic.*

Proof. Assume there exists an optimal irregular sequence $\sigma^* = (\sigma_t^*)_{t=0}^{\infty}$, where σ_t denotes the operation according to σ at period t. Let the sequence be separated into intervals i_k of length $T = lcm(\lambda_1, \ldots, \lambda_n)$, i.e. $i_k = (\sigma_t^*)_{t=kT}^{(k+1)T}$ for $k = 0, 1, \ldots$. Let $s(i_k)$ be the first state of the interval.
Due to the finiteness of the state space there must exist an interval i_k, which first state $s(i_k)$ occurs infinitely many times in σ^*. Consider the sequence of intervals between any two occurrences of state $s(i_k)$, $I_j = i_k, \ldots, i'_k$, where $s(i_k) = s(i_{k+1})$. Let I^* be the sequence with the lowest cost in σ^*. Consider then an alternative sequence $\sigma' = \{I^* y, I^*, I^* \ldots\}$. It follows that $C(\sigma') = C(\sigma^*)$ and σ' is a cyclic policy with base sequence I^*. Thus, any optimal irregular policy can be rewritten as a cyclic optimal cost policy. □

Theorem 3.4.1. *There always exists a* cyclic *policy σ^*, such that $L(\sigma^*) \leq 8T$, where $T = lcm(\lambda_1, \ldots, \lambda_n)$.*

Proof. Lemma 3.4.2 implies that there are only 8 different states. Following the construction of Lemma 3.4.3, we know that the length of I^* is at most $8T$, where $T = lcm(\lambda_1, \ldots, \lambda_n)$. □

Note that Theorem 3.4.1 implies that there exists an optimal cyclic policy of size T and therefore we only restrict our algorithms for cyclic policies of that length.

3.4.2 Algorithm

The state-space representation in Figure 3.1 naturally structures the dynamic programming algorithm that is presented in Algorithm 7. For easier notation, we encode upstream U and downstream D as 0 and 1, respectively. In the preliminaries, we defined and computed the number of arrivals per time unit for each side a_t^0 and a_t^1, the length of the longest cycle T and the state space S.

An optimal cyclic policy could have any of the states as its starting state, therefore we iteratively initiate and execute a dynamic program for each s in S. In one iteration, we first initialise a value matrix. The dimensions of the value matrix are the time horizon times the state space for each time point shown in Figure 3.1.

Given a time t and some state $(\omega_p, \omega_{|1-p|}, p)$, the lock is on side p and has not processed vessels from side p since $1 + \omega_p + \omega_{|1-p|}$ time units. The cost of switching from side p to $|1 - p|$ is the accumulated waiting time of vessels that arrived on side p in the last $1 + \omega_p + \omega_{|1-p|}$ time units, i.e. the *cost of processing* is defined by

$$c(t, \omega_p, \omega_{|p-1|}, p|) = \sum_{i=0}^{\omega_p + \omega_{|1-p|}+1} i a^p_{t-i}$$

To ensure that the policy is cyclic, we initiate the values in the last time step of the value matrix in the following way. Given that at time 0 the system is in state $s = (\omega^s_p, \omega^s_{|1-p|}, p^s)$, if $\omega^s_{|1-p|} = 0$ then the lock operator has processed vessels from side $|1 - p|$ at time $T - 1$. Thus, the state that corresponds to the lock being on side $|1-p|$ must be initiated with the respective costs of processing vessels from that side. If $\omega^s_{|1-p|} = 1$, then the lock operator has not processed any vessels and we initiate the respective states with zero waiting cost.

The recursive formula is defined by three cases. In Figure 3.1, for a given state, there are two nodes with multiple outgoing edges, $(0, 0, p)$ and $(0, 1, p)$. If $\omega_p = 0$ and $\omega_{|1-p|} \in \{0, 1\}$, the lock operator has the choice to either wait or to process vessels from side p. In case of waiting, the system will be in state $(1, 1, \omega_{|p-1|})$ at time $t+1$ and no waiting occurs. In case of processing vessels from side p, the system transitions to $(\omega_{|1-p|}, 0, |p - 1|)$ and incurs a cost of $c(t, \omega_p, \omega_{|p-1|}, p|)$. For the other nodes, there is only one outgoing edge. According to Lemma 3.4.2, the lock operator has to process vessels from side p . The system at time $t+1$ transitions to state $(\omega_{|1-p|}, 0, |p-1|)$ at a cost of $c(t, \omega_p, \omega_{|p-1|}, p|)$.

Algorithm 3: Optimal Cyclic Solution

Input: $\mathcal{I}$
Output: Minimal long run average waiting time.
Preliminaries;
Let $T := 8lcm(\lambda_1, \ldots, \lambda_n)$;
Let $a_t^0 := \sum_{i \in \mathcal{I}, d_i = U} a_{i,t}$ and $a_t^1 := \sum_{i \in \mathcal{I}, d_i = D} a_{i,t}$ for $t = 0, \ldots T-1$ and $a_{-t}^0 := a_{T-t}^0$ and $a_{-t}^1 := a_{T-t}^1$ for $t = 1, 2, 3$;
Let $S := \left\{ \left\{ \omega_p, \omega_{|1-p|}, p \right\} \text{ for all } \omega_p, \omega_{|1-p|}, p \in \{0,1\} \right\}$;
for $s \in S$ **do**
  Let $\omega_p^s, \omega_{|1-p|}^s, p^s$ define the values of state s;
  Initialisation;
  Let D_s be a $T \times 2 \times 2 \times 2$ matrix ;
  $D_s[t, \omega_p, \omega_{|1-p|}, p] = \infty$ for all $t = 0, \ldots T-1$ and $\omega_{|p|}, \omega_{|1-p|}, p \in \{0,1\}$;
  if $\omega_{|1-p|}^s = 0$ **then**
    $D_s[T-1, j, \omega_p^s, |1-p^s|] = \sum_{i=0}^{\omega_p^s + j + 1} i a_{t-i}^{p^s}$ for $j = 0, 1$;
  else
    $D_s[T-1, 0, \omega_{|1-p|}^s, p^s] = 0$;
  end
  Recursion;
  $$D_s[t, \omega_p, \omega_{|1-p|}, p] = \begin{cases} \min \left\{ D[t+1, 0, 0, |1-p|] + \sum_{i=0}^{\omega_p + \omega_{|1-p|} + 1} i a_{t-i}^p,\ D[t+1, 1, 0, p] \right\} \\ \text{if } \omega_p = 0 \text{ and } \omega_{|1-p|} = 0, \\ \min \left\{ D[t+1, 1, 0, |1-p|] + \sum_{i=0}^{\omega_p + \omega_{|1-p|} + 1} i a_{t-i}^p,\ D[t+1, 1, 1, p] \right\} \\ \text{if } \omega_p = 0 \text{ and } \omega_{|1-p|} = 1, \\ D[t+1, \omega_{|1-p|}, 0, |1-p|] + \sum_{i=0}^{\omega_p + \omega_{|1-p|} + 1} i a_{t-i}^p \\ \text{otherwise} \end{cases} ;$$
end
Return $\min_{s \in S} D_s[1, \omega_p^s, \omega_{|1-p|}^s, p^s]$

3.5 Approximation Scheme

The complexity of Algorithm 7 is $O(Tn)$ and therefore it is not polynomial in the input size. This implies that for high and/or many relatively prime periodicities, the running time drastically increases and the algorithm loses its appeal to practitioners.

Algorithm 7 uses dynamic programming to compute an optimal (cyclic) schedule of length $8T$. If T is very large, it is not only intractable to compute such a schedule, but even *storing* such a large schedule would be impractical. For practical purposes, it would be very useful to have an algorithm that, in reasonable time, computes a good sequence of actions for the immediate future. In this section, we show how to adapt Algorithm 7 to only look a "short" amount of time into the future to obtain schedules that are close to optimal in a reasonable amount of time.

Our algorithm is a so called incremental polynomial time algorithm, which means that the next set in the list of output sets is generated in time that is polynomial in the size of the input plus the size of the already generated part of the output. Such algorithms have been applied in enumeration and high multiplicity scheduling, see for example [35] and [36].

Our algorithm yields solutions with an approximation guarantee. For any $\epsilon > 0$, looking $O(n^2/\epsilon)$ time units into the future, we obtain solutions that are $1 + \epsilon$-approximations of an optimal cyclic schedule. Our algorithm is thus an *incremental fully polynomial time approximation scheme (IFPTAS)*. To our knowledge, such an approximation scheme has not been considered in literature. The algorithm is given in Algorithm 6, while the approximation guarantee is proven in Theorem 3.5.1. Besides the proposed bound, the incremental construction of policies may be used by practitioners to adjust their schedule to unforeseen delays of some of the vessel streams.

Lemma 3.5.1. *Let σ_i^{opt} be the sequence of actions of an optimal policy in time*

Algorithm 4: $ExactSolution(t_1,\ t_2)$

Input: $\mathcal{I}, t_1, t_2$

Output: Minimal cost over the interval $[t_1, t_2]$.

Preliminaries;

Define $a_t^0 := \sum_{i\in\mathcal{I}, d_i = U} a_{i,t}$ and $a_t^1 := \sum_{i\in\mathcal{I}, d_i = D} a_{i,t}$;

Let $T = t_2 - t_1$;

Initialisation;

Let D be a $T \times 2 \times 2 \times 2$ matrix;

$D[T, \omega_{|p|}, \omega_{|1-p|}, p] = 0$ for all $\omega_{|p|}, \omega_{|1-p|}, p = \{0, 1\}$;

Recursion;

$$D[t, \omega_p, \omega_{|1-p|}, p] = \begin{cases} \min\left\{ D[t+1, 0, 0, |1-p|] + \sum_{i=0}^{\omega_p + \omega_{|1-p|}+1} i a_{t-i}^p,\ D[t+1, 1, 0, p] \right\} \\ \text{if } \omega_p = 0 \text{ and } \omega_{|1-p|} = 0, \\ \min\left\{ D[t+1, 1, 0, |1-p|] + \sum_{i=0}^{\omega_p + \omega_{|1-p|}+1} i a_{t-i}^p,\ D[t+1, 1, 1, p] \right\} \\ \text{if } \omega_p = 0 \text{ and } \omega_{|1-p|} = 1, \\ D[t+1, \omega_{|1-p|}, 0, |1-p|] + \sum_{i=0}^{\omega_p + \omega_{|1-p|}+1} i a_{t-i}^p \\ \text{otherwise} \end{cases} ;$$

$C^* = \min_{i,j,p\in\{0,1\}} D[1, \omega_p, \omega_{|1-p|}, p]$;

Let σ^* be the policy that corresponds to the cost C^* ;

Return C^*, σ^*

Algorithm 5: $ExactSolution_{FixedStart}(t_1,\ t_2, s)$

Input: $\mathcal{I}, t_1, t_2, s$

Output: Minimal cost over the interval $[t_1, t_2]$ given this system is in state s at $t_1 - 1$.

Preliminaries;

Define $a_t^0 := \sum_{i \in \mathcal{I}, d_i = U} a_{i,t}$ and $a_t^1 := \sum_{i \in \mathcal{I}, d_i = D} a_{i,t}$;

Let $T = t_2 - t_1$;

Let $\omega_p^s, \omega_{|1-p|}^s, p^s$ define the values of state s;

Initialisation;

Let D be a $T \times 2 \times 2 \times 2$ matrix;

$D[T, \omega_{|p|}, \omega_{|1-p|}, p] = 0$ for all $\omega_{|p|}, \omega_{|1-p|}, p = \{0, 1\}$;

Recursion;

$$D[t, \omega_p, \omega_{|1-p|}, p] = \begin{cases} \min\left\{ D[t+1, 0, 0, |1-p|] + \sum_{i=0}^{\omega_p + \omega_{|1-p|}+1} i a_{t-i}^p,\ D[t+1, 1, 0, p] \right\} \\ \text{if } \omega_p = 0 \text{ and } \omega_{|1-p|} = 0, \\ \min\left\{ D[t+1, 1, 0, |1-p|] + \sum_{i=0}^{\omega_p + \omega_{|1-p|}+1} i a_{t-i}^p,\ D[t+1, 1, 1, p] \right\} \\ \text{if } \omega_p = 0 \text{ and } \omega_{|1-p|} = 1, \\ D[t+1, \omega_{|1-p|}, 0, |1-p|] + \sum_{i=0}^{\omega_p + \omega_{|1-p|}+1} i a_{t-i}^p \\ \text{otherwise} \end{cases};$$

$$C^* = \begin{cases} \min\left\{ D[1, 0, 0, |1-p^s|] + \sum_{i=0}^{\omega_p^s + \omega_{|1-p|}^s + 1} i a_{t-i}^{p^s},\ D[1, 1, 0, p^s] \right\} \\ \text{if } \omega_p^s = 0 \text{ and } \omega_{|1-p|}^s = 0, \\ \min\left\{ D[1, 1, 0, |1-p^s|] + \sum_{i=0}^{\omega_p^s + \omega_{|1-p|}^s + 1} i a_{t-i}^{p^s},\ D[1, 1, 1, p^s] \right\} \\ \text{if } \omega_p^s = 0 \text{ and } \omega_{|1-p|}^s = 1, \\ D[1, \omega_{|1-p|}^s, 0, |1-p^s|] + \sum_{i=0}^{\omega_p^s + \omega_{|1-p|}^s + 1} i a_{t-i}^{p^s} \\ \text{otherwise} \end{cases};$$

Let σ^* be the policy that corresponds to the cost C^* ;

Return C^*, σ^*;

Algorithm 6: $IFPTAS(k, \varepsilon, s)$

Input: k, ε, s
- Let $\tilde{T} = 40n^2/\varepsilon$;
Let $t_1 = 0$;
if *state s is given* **then**
  Compute $ExactSolution_{FixedStart}(t_i, t + \tilde{T}, s)$;
else
  Compute $ExactSolution(t_i, t + \tilde{T})$;
end
Let C_1^* and σ_1^* be cost and schedule;
for $i = 1, \ldots, k$ **do**
  if *there exist a* $t' = t_i, \ldots, t_i + \tilde{T} - 1$ *s.t.* $a_{t'} = a_{t'+1} = 0$ **then**
    Let t_{i+1} be period of first arrival after t' ;
    Compute $ExactSolution(t_{i+1}, t_{i+1} + \tilde{T})$;
    Let C_{i+1}^* and σ_{i+1}^* be cost and schedule;
    Let σ_i' represent the schedule of empty slots between σ_i^* and σ_{i+1}^*;
    Fill σ_i' arbitrarily such that $\sigma_i^* \cup \sigma_i' \cup \sigma_{i+1}^*$ is feasible;
    Let $\sigma_i = \sigma_i^*[t_i, t'] \cup \sigma_i'$;
  else
    if $C^* \leq 2n/\varepsilon$ **then**
      Let $t_{i+1} = t_i + 20n^2/\varepsilon + 1$;
      Let s be the state of σ_i^* at $t_i + 20n^2/\varepsilon$;
      Compute $ExactSolution_{FixedStart}(t_{i+1}, t_{i+1} + \tilde{T}, s)$;
      Let C_{i+1}^* and σ_{i+1}^* be cost and schedule;
      Let $\sigma_i = \sigma_i^*[t_i, t_i + 20n^2/\varepsilon]$;
    else
      Let $t_{i+1} = t_i + \tilde{T} + 3$;
      Compute $ExactSolution(t_{i+1}, t_{i+1} + \tilde{T})$;
      Let C_{i+1}^* and σ_{i+1}^* be cost and schedule;
      Let σ_i' represent the schedule of empty slots between σ_i^* and σ_{i+1}^*;
      Fill σ_i' arbitrarily such that $\sigma_i^* \cup \sigma_i' \cup \sigma_{i+1}^*$ is feasible;
      Let $\sigma_i = \sigma_i^*[t_i, t_i + \tilde{T}] \cup \sigma_i'$;
    end
  end
end
return $\bigcup_{i=1}^{k} \sigma_i$

interval $[t_i, t_{i+1}]$. *At every iteration* i *in Algorithm 6,*

$$C(\sigma_i) \leq (1+\varepsilon)C(\sigma_i^{opt}).$$

Proof. In the first case, there are at least two consecutive periods of no arrival. Since these periods can be used to position the lock to each side at no cost, we know that the partial exact schedule must be equivalent to the optimal solution. Thus, in this case it holds true that $C(\sigma_i) = C(\sigma_i^{opt})$.

If the cost of the partial exact solution is less or equal than $2n/\varepsilon$, we output the first half of the solution. Since the length of the time window is $40n^2/\varepsilon$, we know that in the second half of the solution there is a consecutive period of length at least $10n$ with zero cost. We claim that the optimal solution is equivalent to the partial exact solution at some point during this zero cost interval. If it does not do the same thing at some point, it means all arrivals during this time period wait at least one unit of time and since we have at least one arrival every 3 time units, it would incur a cost of at least $5n$. If this were the case, we could improve the optimal solution by switching to the partial exact solution at the start of the zero cost period period, and switching back at the end of it. Since the cost of a switch is upper bounded by $2n$, the two switches would occur a cost of $4n$ and thus it would decrease the cost of the optimal solution by at least n, which is a contradiction. Since the partial exact solution reaches the same state as the optimal solution at some point, we know the the cost of the partial exact solution leading up to this point is also optimal. So our choice to output the first half of the solution was optimal, and the configuration we restart in is also a configuration in the optimal solution.

If the cost of the partial exact solution is at least $2n/\varepsilon$, we can simply output it and restart solving at $\tilde{T}+3$. We can use time periods $\tilde{T}+1$ and $\tilde{T}+2$ to link up the σ_i and σ_{i+1} at cost of at most $2n$. Since the partially exact solution is a lower bound on the cost of the part of the optimal solution from t_i to time $\tilde{T}$, we know that the total cost is at most $2n/\varepsilon + 2n \leq (1+\varepsilon)C(\sigma_i^{opt})$.

□

Theorem 3.5.1. *Let* $\lim_{k\to\infty} \sigma_k \to \sigma_\infty$*, then* $C(\sigma_\infty) \leq (1+\varepsilon)C(\sigma^{opt})$

Proof. Follows straightforward from the previous lemma.

$$\lim_{k\to\infty} C(\sigma_k) = \lim_{k\to\infty} C(\bigcup_{i=1}^{k} \sigma_i) = \lim_{k\to\infty} \sum_{i=1}^{k} C(\sigma_i) \leq (1+\varepsilon) \lim_{k\to\infty} \sum_{i=1}^{k} C(\sigma_i^{opt}) \tag{3.3}$$

$$\iff \sigma_\infty \leq (1+\varepsilon)C(\sigma^{opt}) \tag{3.4}$$

□

4

Velocity optimisation on waterway networks

Adapted from: J. Golak, C. Defryn, and A. Grigoriev, "Optimizing fuel consumption on inland waterway networks," *Working Paper*, 2021.

4.1 Our Contribution

To our best knowledge, only [12] minimise overall greenhouse gas emissions by optimising the velocity at which vessels approach the locks. However, the mathematical programming formulation is restricting the applicability of these results in real inland waterway networks, since the large-scale instances quickly become intractable.

In this chapter, we extend and generalise the results from [12] in several ways. First, we allow for a network of waterways with multiple entry and exit points, and any lock configuration. Second, vessels have individual arrival and destination locations in the network. These locations can be entry/exit points of the system or any location along the waterway. Finally, our methods also allow for multi-chambered locks. Most importantly, we design a lock-search based heuristic to tackle real-life large-scale instances of the lock scheduling problem minimising the total greenhouse gas emission of the fleet of vessels.

4.2 The velocity optimization problem

4.2.1 Mathematical notation

Consider an inland waterway network that consists of a set of river segments with given lengths. These segments connect a set of locks and multiple entry/exit points. We refer to this set of locks as $\mathcal{L}$. Note that, in contrast to existing literature, we do not assume that locks are sequentially placed along one river segment, but that any lock configuration is feasible. An example of such a river network is depicted in Figure 4.1. In this figure, the lines represent river segments and the arrows point in the downstream direction. The circles depict the entry/exit points of the river system and the parallel lines show the locks.

Over a time period from 0 to $\mathcal{T}$, vessels arrive, navigate through the network and leave the system. Let $\mathcal{S}$ define the set of vessels. For each vessel $s \in \mathcal{S}$, the time of arrival into and the time of desired departure

out of the system are given and denoted by A_s and D_s, respectively. Vessels may enter and exit the system at entry/exit points, or skippers can anchor their vessel at any location on a river. In the latter case, the arrival/exit location is between the two endpoints of that specific river segment. The trajectory of a vessel is the shortest path from arrival location to destination. Let the sequence of locks a vessel $s \in \mathcal{S}$ needs to cross be given by $\mathcal{L}_s = \{\ell_{s,1}, \ldots, \ell_{s,n_s}\}$ where $\mathcal{L}_s \subseteq \mathcal{L}$. The lengths of the river segments that vessel s needs to cross is defined as $\mathcal{D}_s = \{d_{s,1}, \ldots, d_{s,n_s+1}\}$. In case a skipper starts or ends by anchoring inside the system, the first or the last segment is only partly crossed. Thus, $d_{s,1}$ and d_{s,n_s+1} may only be a fraction of the length of the river segment. The other distances, i.e. between two locks, are the complete lengths of the river segment. Finally, a vessel can approach a lock coming from up or downstream. Let $\mathcal{W}_s = \{w_{s,1}, \ldots, w_{s,n_s}\}$ be the set of directions, where $w_{s,i} \in \{up, down\}$ is the direction from which vessel s approaches its ith lock. Note that, for any vessel, the directions can alternate for each lock in its path. An example for this would be a vessel which would travel from the bottom right to the top right node in Figure 4.1. In such a case, the vessel would approach the bottom lock from downstream and the top lock from upstream.

For each lock $\ell \in \mathcal{L}$, the capacity of a chamber, i.e. the number of vessels that can be leveled up or down simultaneously, is given by C_ℓ. Furthermore, the processing time, i.e. the time it takes to load, unload and process the batch in the chamber, is defined by T_ℓ. We assume that vessels are homogeneous in size and in required processing time. A lock may consist of multiple independently working chambers. The number of chambers for each lock $\ell \in \mathcal{L}$ is denoted by B_ℓ. We assume that every chamber of a lock has the same capacity and processing time.

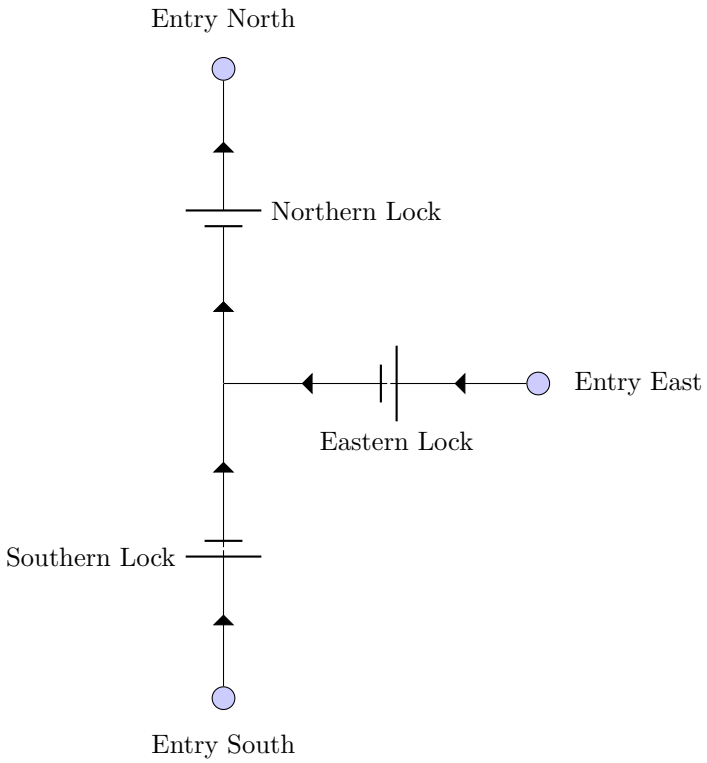


Figure 4.1: A River Network

Example 4.2.1. *Let us clarify some definitions, using the structure of the Amsterdam Rhine Canal case, which is introduced more elaborately in Section 4.4.1. Here, the set of locks is defined as* $\mathcal{L} = \{north, east, south\}$*. The setup is illustrated in Figure 4.1. Consider a vessel* s *that enters the system at the south entry and exits at the northern entry. Thus, it first sails downstream from entry south to the southern lock. Then it continues to sail downstream from the southern lock to the northern lock and finally leaves the system at Entry North. Formally,* $\mathcal{L}_s = \{south, north\}$ *and* $\mathcal{W}_s = \{down, down\}$*. The distances on each segments are respectively* $\mathcal{D}_s = (9.39, 16.20, 18.84)$*. Additionally, assume that this vessel* s *arrives into the system at 7:24 AM*

and desires to leave it at 12:30 AM. Throughout the entire article, we use minutes after midnight as the time unit. This translates to an arrival times of $A_s = 451$ and a departure time of $D_s = 751$. Under the assumption that there are no waiting times at the locks, the vessel has, therefore, a sailing time of 255 minutes, or 4 hours and 15 minutes ($A_s - D_s - T_{north} - T_{south}$).

The cost function $C(v)$ relates fuel consumption to vessel velocity, expressed per unit of distance. Let $v_{s,i}$ denote the velocity of vessel s on the ith segment of the network, and let $\mathcal{V}_s = (v_{s,1} \ldots, v_{s,n_s+1})$. Then the fuel consumption of vessel s is

$$C(\mathcal{V}_s) = \sum_{i=1}^{n_s+1} d_{s,i} C(v_{s,i}), \tag{4.1}$$

and the total fuel consumption of the fleet in a solution is given by

$$C_{total}(\mathcal{V}) = \sum_{s \in \mathcal{S}} C(\mathcal{V}_s), \tag{4.2}$$

which is the expression that we aim to minimize. A widely accepted approximation of the relationship between fuel consumption and velocity, expressed in units of distances is quadratic, see [12].

The solution of this optimization problem is a schedule of lockage starting times (i.e., the time at which the service of a vessel at the lock starts) at each chamber of each lock, an assignment of vessels to specific lockages, and a velocity for each vessel on each river segment between origin and destination. The schedule is feasible if all of the following conditions are satisfied.

- Each vessel must pass all locks along its itinerary in the given order and from the given directions.
- The velocity of each vessel must be consistent with the lockage starting times, the arrival time, and the deadline.

- The velocity of each vessel should be in the interval $[0, V_s^{max}]$ at all times, where V_s^{max} represents the maximum velocity at which vessel s can sail. We assume that this maximum velocity does not exceed the velocity limit of the river segments in the network. We assume that there is no minimum velocity throughout the chapter.

- The lockages of each lock chamber must alternate between downstream and upstream, and every pair of lockages is at least T_l time units apart.

- The number of vessels assigned to one lockage must not exceed the capacity of the lock.

Example 4.2.2. *Let us again consider vessel s on the Amsterdam Rhine Canal. Let $\mathcal{L}_s = \{south, north\}$, $\mathcal{W}_s = \{down, down\}$ and $\mathcal{D}_s = (9.39, 16.20, 18.84)$. Furthermore, assume that $A_s = 451$, $D_s = 751$ and $C(v) = v^2$. Also note that the processing time of a batch at Northern Lock and Southern Lock are 22 and 23 minutes, respectively, i.e. $T_{north} = 23$ and $T_{south} = 22$.*

Vessel s chooses to sail at a constant velocity of 0.3 km per minute on each river section, that is $v_s = (0.3, 0.3, 0.3)$. For reasons of simplicity, we assume that the lockages are fully synchronized with the velocity of vessel s, meaning that vessel s does not have to wait in front of the locks. The arrival time of vessel s at its first lock is then $A_s + d_{s,1}/v_{s,1} = 482.3$ minutes. Similarly, the arrival time of vessel s at the second lock is $482.3 + T_{l_{s,1}} + d_{s,2}/v_{s,2} = 558.3$ minutes. Lastly, the vessel arrives at the destination at time $558.3 + T_{l_{s,2}} + d_{s,3}/v_{s,3} = 644.1$ minutes. Conclusively, vessel s arrives at its destination before its specified deadline. The fuel consumption of vessel s is then calculated as follows:

$$C(\mathcal{V}_s) = d_{s,1}v_{s,1}^2 + d_{s,2}v_{s,2}^2 + d_{s,3}v_{s,3}^2 = 3.99 \tag{4.3}$$

4.2.2 Mixed-Integer Linear Program

In this section, we describe a Mixed-Integer Linear Program (MILP) formulation to compute an optimal velocity for every vessel on each river section as this solution minimizes total aggregated fuel consumption. The MILP is an extension of the model from [12], which aimed at minimizing emissions on a waterway with multiple locks. Our MILP differs from the model by [12] in three ways:

1. The setup is extended to a network of waterways in which locks are placed in any configuration.

2. Vessels are allowed to have individual arrival and destination locations in the network.

3. Locks can have multiple chambers.

Sets The decision variables in the model are based on possible lockages. Assuming an adversary lock operator, each vessel would face the lock position to its opposite side. Therefore, the number of lockages in the optimal solution does not need to be greater than double the number of ships. Thus, the upper bound for the number of lockages is given by $K = 2|S|$. The set of possible lockages is, therefore, defined as $\mathcal{K} = \{1, \ldots, K\}$. For each lock $\ell \in \mathcal{L}$, U_ℓ and D_ℓ represent the set of vessels that approach lock ℓ form upstream and downstream, respectively.

Decision Variables For the decision variables for velocity, consider $\bar{v}_{s,i} = \frac{1}{v_{s,i}}$ for each $s \in S$ and $i \in \{1, \ldots, n_s\}$ and let $\bar{C}(\bar{v})$ express the total fuel consumption as a function of the reciprocal of the vessel velocity, $\bar{v}_{s,i}$. Let $t(\ell, i, k)$ be the starting time of the k'th lockage of the i'th of lock ℓ for all $k \in \mathcal{K}$. Furthermore, for each $s \in S$, $\ell \in \mathcal{L}_s$, $i \in \{1, \ldots, B_\ell\}$ and $k \in \mathcal{K}$, we define $z(s, \ell, i, k)$ as follows:

$$z(s,\ell,i,k) = \begin{cases} 1, & \text{if vessel } s \text{ is handled by the } k\text{th lock movement of chamber } i \text{ of lock } \ell. \\ 0, & \text{otherwise} \end{cases}$$

MILP formulation

$$\min \sum_{s \in S} \bar{C}(\bar{\mathcal{V}}_s) \tag{4.4}$$

subject to

$$A_s \leq t(\ell_{s,1}, i, k) - d_{s,1} v_{s,1} + M_s^a (1 - \sum_{\kappa=0}^{k} z(s, \ell_{s,1}, i, \kappa)) \\ \forall s \in \mathcal{S}, i = 1, \ldots, B_{\ell_{s,1}}, k \in \mathcal{K} \tag{4.5}$$

$$D_s \geq t(\ell_{s,n_s}, i, k) + T - d_{s,n_s+1} v_{s,n_s+1} - M_s^d (1 - \sum_{\kappa=k}^{K} z(s, \ell_{s,n_s}, i, \kappa)) \\ \forall s \in \mathcal{S}, i = 1, \ldots, B_{\ell_{s,n_s}}, k \in \mathcal{K} \tag{4.6}$$

$$z(s_1, \ell, i, k) + z(s_2, \ell, i, k) \leq 1 \\ \forall \ell \in \mathcal{L}, \forall s_1 \in \mathcal{U}_l, \forall s_2 \in \mathcal{D}_l, k \in \mathcal{K} \tag{4.7}$$

$$z(s_1, \ell, i, k-1) + z(s_2, \ell, i, k) \leq 1 \\ \forall \ell \in \mathcal{L}, \forall s_1, s_2 \in \mathcal{U}_l, k \in \mathcal{K} \setminus \{1\} \tag{4.8}$$

$$z(s_1, \ell, i, k-1) + z(s_2, \ell, i, k) \leq 1 \\ \forall \ell \in \mathcal{L}, \forall s_1, s_2 \in \mathcal{D}_l, k \in \mathcal{K} \setminus \{1\} \tag{4.9}$$

$$t(\ell, i, k) - t(\ell, i, k-1) \leq T_\ell$$

$$\forall \ell \in \mathcal{L}, \forall i = 1, \ldots, B_\ell, \forall k \in \mathcal{K} \setminus \{1\} \tag{4.10}$$

$$\sum_{s \in \mathcal{S}} z(s, \ell, i, k) \leq C_l$$
$$\forall \ell \in \mathcal{L}_s, i = 1 \ldots, B_\ell, k \in \mathcal{K} \tag{4.11}$$

$$\sum_{i=0}^{B_\ell} \sum_{k \in \mathcal{K}} z(s, \ell, i, k) = 1$$
$$\forall s \in \mathcal{S}, \forall \ell \in \mathcal{L}_s \tag{4.12}$$

$$t(\ell_{s,j-1}, i, k) \geq t(\ell_{s,j}, i, k) + T_{s,j} + v_{s,j} d_{s,j} -$$
$$M_{k_1,k_2}(2 - \sum_{\kappa_1=1}^{k_1} z(s, \ell, i, \kappa_1) - \sum_{\kappa_2=k_2}^{K} z(s, \ell, i, \kappa_2)$$
$$\forall s \in S, k_1, k_2 \in \mathcal{K} \tag{4.13}$$

$$1/V_s^{max} \leq \bar{v}_{s,i}$$
$$\forall s \in \mathcal{S}, i = 1, \ldots, n_s + 1 \tag{4.14}$$

$$z(s, \ell, i, k) \in \{0, 1\}$$
$$\forall s \in S, \forall \ell \in \mathcal{L}_s, i = 1 \ldots, B_\ell, k \in \mathcal{K} \tag{4.15}$$

$$t(\ell, i, k) \in \mathbb{R}^+$$
$$\forall \ell \in \mathcal{L}, i = 1 \ldots, B_\ell, k \in \mathcal{K} \tag{4.16}$$

The objective function (4.4) minimies the aggregated costs of the individual vessels. Constraint (4.5) ensures that vessels arrive at the lock

before their respective lockage time has started and constraints (4.6) ensure that vessels are leaving the system before their deadline. Constraints (4.5) and (4.6) are based on values M_s^A and M_s^D, which are large constants specified in [12]

Constraints (4.7) to (4.13) are used to model the working of the lockages. Thus, (4.7) ensures that lockages are alternating between upstream and downstream. Constraints (4.8) and (4.9) ensure that only vessels from one side can enter a lockage. The processing time of a lockage is modeled by Constraint (4.10), while the capacity of a chamber is modeled by (4.11). Constraints (4.12) and (4.13) ensure that each vessel is processed by all of its locks and that it arrives at each lock before its assigned lockage time.

Herewith, we extended the MILP formulation provided in [12] but broadened the set notation, such that each vessel has an individual set of locks, set of distances and set of directions its approaching the lock. [12] improve their formulation by adding a set of valid inequalities. In our computational experiments we use these inequalities as well.

4.3 Local search heuristic

Due to the limited scalability of the MILP, it would be impossible to solve realistic instances within an acceptable computation time. To allow our methods to be used in practice, we propose a heuristic to determine high-quality velocity advice for all skippers on the waterway network within a reasonable amount of computation time.

4.3.1 Preliminaries and assumptions

To reduce the solution space and, consequently, the complexity of the velocity optimization problem, we impose a fixed schedule on the starting times of lockages. This is in contrast to the MILP formulation, where the lockage could be scheduled at any feasible time throughout the planning horizon. For each chamber independently, lockages are

set consecutively, such that there is no idle time between two lockages and that they are alternating between up and downstream. Furthermore, the starting times of the first lockage of each chamber is evenly spread on the interval $[0, 2T_l]$. As a result, each lock in each direction has a chamber starting the lockage process every $T_l^* = 2T_l/B_l$ time units.

Formally, let $K_l = \lfloor \mathcal{T}/T_l^* \rfloor$ represent the maximum number of lockages per direction for lock l. Define $t(l, k, w)$ as the starting time of the kth lockage in direction w of lock l. Then, fix the starting times in the following way:

$$\begin{aligned} t(l, k, up) &= 2\ k\ T_l^* && \forall l \in \mathcal{L}, k = \{0, \ldots, K_l/2\} \\ t(l, k, down) &= 2\ (k+1)\ T_l^* && \forall l \in \mathcal{L}, k = \{0, \ldots, K_l/2\} \end{aligned}$$

Similarly, let $h(l, k, w)$ represent the set of vessels assigned to the kth lockage, in direction w of lock l. In the initialisation, every element of that set is declared empty, i.e. $h(l, k, w) = \emptyset$ for $l \in \mathcal{L},\ k = 0, \ldots, K_l,\ w \in \{up, down\}$.

Example 4.3.1. *We apply the initialisation procedure on the Amsterdam Rhine Canal case. As data is expressed in minutes, $\mathcal{T}$ is set to 1440. Given the measured parameters for this case, summarised in Table 4.4, the time between two lockages and the maximum number of lockages for a lock l, i.e. T_l^* and K_l are:*

$$T_{east}^* = \frac{2T_{east}}{B_{east}} = 44,\ K_{east} = \left\lfloor \frac{\mathcal{T}}{T_{east}} \right\rfloor = 32$$

$$T_{north}^* = \frac{2T_{north}}{B_{north}} = 23,\ K_{north} = \left\lfloor \frac{\mathcal{T}}{T_{north}} \right\rfloor = 62$$

$$T_{south}^* = \frac{2T_{south}}{B_{south}} = 22,\ K_{south} = \left\lfloor \frac{\mathcal{T}}{T_{south}} \right\rfloor = 65$$

The resulting schedule of lockages is given in Table 4.1.

Table 4.1: Schedule of starting times

Eastern Lock			**Northern Lock**			**Southern Lock**		
k	up	$down$	k	up	$down$	k	up	$down$
0	0	44	0	0	23	0	0	22
1	44	88	1	23	46	1	22	44
2	88	132	2	46	69	2	44	66
3	132	176	3	69	92	3	66	88
...	...	...	...	...	...	...	...	...
31	1364	1408	61	1403	1426	64	1408	1430
32	1408	—	62	1426	—	65	1430	—

The arrival times of vessels at every lock are determined by their velocity assignment. Assume that a vessel is assigned to lockage k' of lock l' going in direction w'. If the arrival time of that vessel is later than $t(l', k', w')$, the schedule becomes infeasible as the assigned lockage is inconsistent with the assigned velocity. On the other hand, if the arrival time of that vessel is earlier than $t(l', k', w')$, then the vessel has idle time at that lock. This implies that the velocity of vessel s can be reduced on the river segment before lock l'. Thus, the total fuel consumption is reduced while not altering the arrival times at any other lock. Consequently, in an optimal schedule, the arrival time at a lock is synchronised with the starting time of the assigned lockage at that lock. This shows, that the assignment of lockages implies the assignment of velocities for a given vessel and vice versa.

The velocity assignment on the last segment is equivalent to the distance divided by remaining time on that segment. This means that the optimal velocity on this river segment is induced by the velocities on the other river segments or, formally, that

$$v_{s,n_s+1} = \min \left\{ \frac{d_{s,n_s+1}}{\tilde{T}_s - \sum_{i=1}^{n_s} d_{s,i}/v_{s,i}}, V_s^{max} \right\} \tag{4.17}$$

where $\tilde{T}_s = D_s - A_s - \sum_{l=1}^{n_s} T_l$.

4.3.2 Construction of the initial solution

Given the predefined schedule of lockage starting times, vessels are assigned to spots in the following manner. First, for each vessel s, let $\tilde{\mathcal{L}}_s$ represent the set of locks, which have not processed vessel s, yet. Initialise $\tilde{\mathcal{L}}_s = \mathcal{L}_s$ for all s. Second, vessels are iteratively selected according to a *least slack time* priority rule until all vessels are scheduled. Each time a vessel s is selected, it is scheduled to the first lock in $\mathcal{L}_s$, say $l_{s,i'}$. On $l_{s,i'}$, the vessel is *assigned to the earliest feasible lockage* that it can reach. Given the velocities, assigned on all segments before i', $v^0_{s,1}, \ldots, v^0_{s,i'-1}$, this earliest reachable lockage is defined by

$$k^0_{s,i'} = \tau_{s,i'}(v^0_{s,1}, \ldots, v^0_{s,i'-1}, v^{max}) \tag{4.18}$$

The following two scenarios might occur:

1. $|h(l_{s,i'}, k^0_{s,i}, w_{s,i'})| = C_{l_{s,i'}}$
   This means that $k^0_{s,i'}$ is not feasible with respect to the lock capacity. If this is the case, then k^0 is incremented by one until a feasible lockage is attained.

2. $h(l_{s,i'}, k^0_{s,i}, w_{s,i'}) = h(l_{s,i'}, k^0_{s,i}, w_{s,i'}) \cup \{s\}$
   This means that k^0 is feasible. If this is the case, s is added to the set of the respective lockage. Furthermore, the lockage assignment is updated by adding $k^0_{s,i'}$ to $\mathcal{K}_s$ and the velocity assignments are updated by setting $v^0_{s,i'} = \tau^{-1}_{s,i'}((k^0_{s,j})_{j \le i})$. Finally, the set of locks that have not yet processed s is updated by removing $l_{s,i'}$ from $\tilde{\mathcal{L}}_s$. A vessel is labeled as scheduled when $\tilde{\mathcal{L}}_s$ is empty. The velocity on the last segment is then computed by Equation (4.17).

For a vessel s, let the resulting velocities be denoted by $v_s^0 = (v_{s,1}^0 \dots, v_{s,n_s+1}^0)$. Similarly, let $\mathcal{K}_s^0 = \{k_{s,1}^0, \dots, k_{s,n_s}^0\}$, where $k_{s,i}^0$ is the assigned lockage of vessel s at its ith lock.

Example 4.3.2. *We continue our example of the Amsterdam Rhine Canal. Consider again vessel s with following specifications:*

$$\begin{aligned} \tilde{\mathcal{L}}_s = \mathcal{L}_s &= \{south, north\} & A_s &= 451 \\ \mathcal{W}_s &= \{down, down\} & D_s &= 751 \\ \mathcal{D}_s &= (9.39, 16.20, 18.84) & C(v) &= v^2 \text{ (with } v_{max} = 0.41) \end{aligned}$$

Assume vessel s is selected, the process of assigning s to lockages is visualised in Figures 4.2a to 4.3b. In Figure 4.2a, time is plotted on the vertical axis and travel distance is plotted on the horizontal axis. Initially, vessel s is at distance 0 at time $a_s = 451$ and is not yet scheduled on any lock. From this starting point all lockages on $l_{s,1}$ in the green region can be reached, which is constrained by the velocity limit. At maximum velocity, vessel s arrives at the first lock at time $a_s + d_{s,1}/v_{max} = 473.90$. Given the predetermined schedule, the next lockage of the southern lock processing vessels coming from downstream is $k = 21$ with $t(south, down, 1) = 484$. Assuming that the lockage has capacity left, vessel s is assigned to it. Furthermore, the lock is removed from $\tilde{\mathcal{L}}_s$ and s is assigned to the lockage, i.e. $k_{s,1}^0 = 21$. The velocity on the first segment is synchronized to the starting time of the lockage, i.e. $v_{s,1} = 0.284$.

When assigned to lockage $k = 21$, vessel s is processed by the southern lock at time $t = 506$ and is now scheduled on the northern lock as shown in Figure 4.2b. Traveling at maximum velocity would result in an arrival time of $506 + d_{s,2}/v_{max} = 545.51$, and the next available lockage is $k = 23$ with $t(south, down, 4) = 552$. Assuming there is capacity left, the vessel is assigned to the lockage. Similarly, the northern lock is removed from $\tilde{\mathcal{L}}_s$, $k_{s,2}^0 = 23$ and $v_{s,2} = 0.352$. Since $\tilde{\mathcal{L}}_s$ is empty, the vessel is labeled as scheduled.

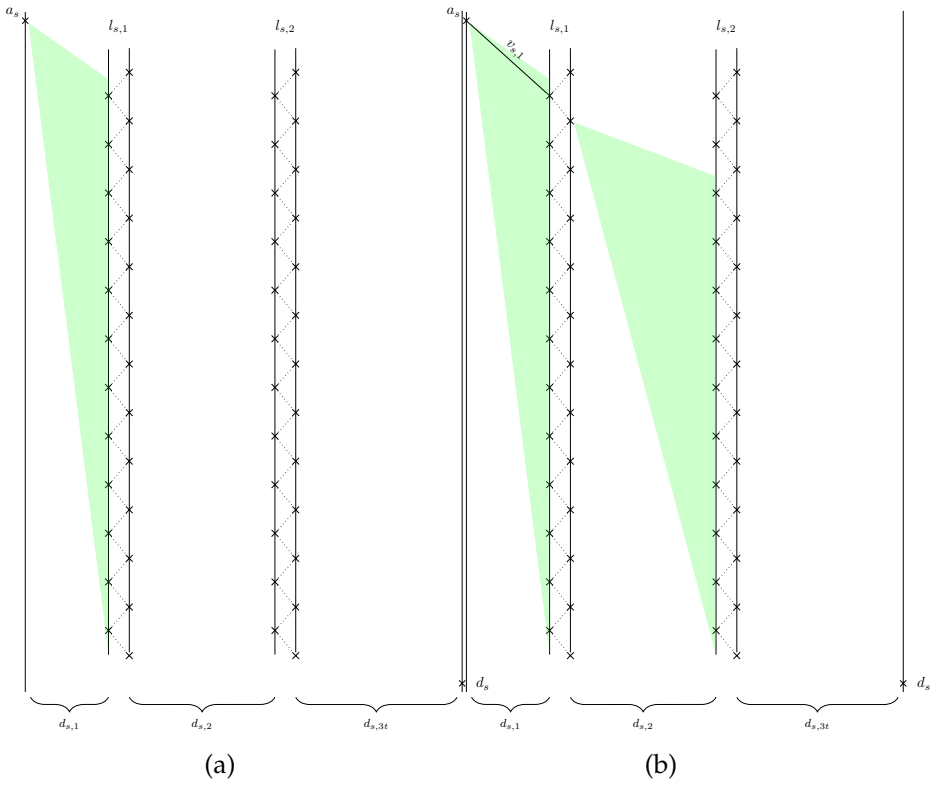


Figure 4.2: Different steps to construct an initial solution for each vessel.

The velocity from northern lock to exit point can be computed by Equation (4.17). *The final assignment of velocities and lockages are then given by* $\mathcal{K}_s^0 = \{21, 23\}$ *and* $v_s^0 = (0.285, 0.352, 0.107)$ *and visualised in Figure 4.3b. The fuel consumption of vessel* s *under the initial schedule is* 2.9871.

Since the vessels are scheduled on the earliest feasible locks available to them, the schedule resulting from the initial solution is costly. By construction, high velocities are advised on earlier segments while skippers can maintain a very low velocity on the last segment. Due to convexity of the fuel consumption function, this unbalanced velocity as-

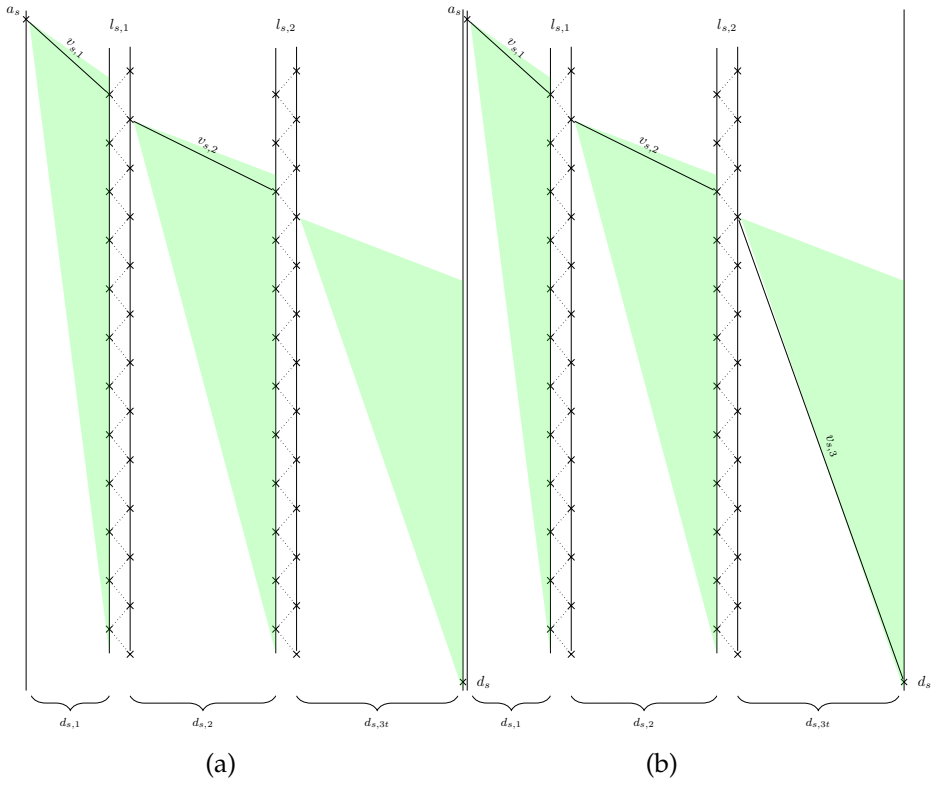


Figure 4.3: Different steps to construct an initial solution for each vessel - continued.

signment results in a high total cost. Although this might seem inconvenient, the proposed procedure performs very well for generating a feasible schedule. The great dependency between decisions for different time intervals and at different locations in the network have forced us to prioritize feasibility over solution quality. Due to high complexity of the system, fixing infeasible operations locally will likely create multiple new conflicts in the schedule.

To further improve the quality of our solution, we propose a dedicated local search operator, which focuses on rebalancing again the velocity assignments.

4.3.3 Balancing velocities using a gradient descent procedure

As discussed above, the required velocity on the final segment of each vessel is deduced directly from its velocity on all previous segments. Therefore, we will focus solely on adjusting explicitly the velocities on segments 1 to n_s.

First, we introduce the gradient of the cost function.

$$\begin{aligned} C_s(v_s) &= \sum_{i=1}^{n_s} d_{s,i} v_{s,i}^2 + d_{s,n+1} \Big(\frac{d_{s,n_s+1}}{\tilde{T}_s - \sum_{i=1}^{n_s} d_{s,i}/v_{s,i}} \Big)^2 \\ \frac{\partial C_s}{\partial v_{s,i}}(v_s) &= 2 d_{s,i} v_{s,i} - \frac{d_{s,i} d_{s,n_s+1}^3}{(\tilde{T}_s - \sum_{i=1}^{n_s} d_{s,i}/v_{s,i})^3 v_{s,i}^2} \\ \nabla C_s(v_s) &= \Big(\frac{\partial C_s}{\partial v_{s,i}} \Big)_{i=1,\dots,n_s} \end{aligned} \tag{4.19}$$

The gradient of the cost function at v_s^0 (i.e., the velocity vector of the initial solution) gives the direction of steepest increase in cost. Consequently, to attain the largest decrease in cost, the velocities of vessel s

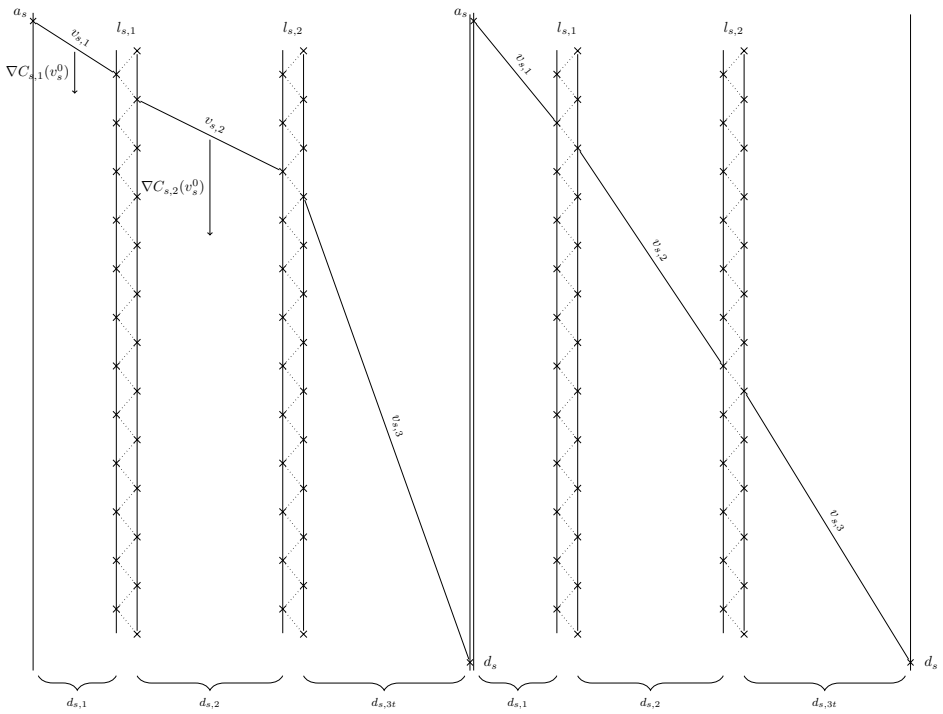
need to be shifted according to $v_s(\gamma) = v_s^0 - \gamma \nabla C_s(v_s^0)$, where γ denotes the magnitude of the shift. For each value of γ there exists a corresponding lockage assignment, which is defined as the latest possible lockage reachable given the adjusted velocities.

Let Γ be the candidate set of unique and feasible lockage combinations for all values of $\gamma \geq 0$. The construction of the candidate lockage assignments Γ is done as follows. For each vessel s, we define $\Delta k_i \in \mathbb{N}$ as the shift of lockages at lock $l_{s,i}$, and $\hat{\gamma}$ as the magnitude of gradient step required for that shift. In other words, $\Delta k_i \in \mathbb{N}$ denotes by how many lock iterations the service of vessels will be advanced (negative Δk_i) or postponed (positive Δk_i). As all lockages were predefined and fixed, we can rewrite $\hat{\gamma}$ as follows:

$$\begin{aligned} \frac{d_{s,i}}{v_{s,i}^0 - \hat{\gamma}_i \nabla C_{s,i}} &= \frac{d_{s,i}}{v_{s,i}^0} + 2\Delta k_i T_{l_{s,i}}^* \\ \hat{\gamma}_i &= \frac{(v_{s,i}^0)^2 \Delta k_i T_{l_{s,i}}^*}{\nabla C_{s,i}(d_{s,i} + v_{s,i}^0 \Delta k_i T_{l_{s,i}}^*)}. \end{aligned} \tag{4.20}$$

The candidate set is now constructed by considering different values for Δk_i. First, we initialise $\Delta k_i = 0$ for all i. Second, Δk_i is incremented (if the gradient is positive) or decremented (if the gradient is negative) by one. The index with the lowest $\hat{\gamma}_i$ is chosen to become permanent, while the other indices are set back to their previous values. $K^c = (K_i^0 + \Delta k_i)_{i=1,\ldots,n_s}$ is added to the set of candidate solutions and the process is repeated. Once a lockage corresponds to an infeasible velocity assignment (e.g., a negative velocity, or a velocity above v_{max}) or the resulting lockage does not exist in the schedule, the iteration stops.

Example 4.3.3. *We continue our example. Consider again vessel s with initial lockage assignment $\mathcal{K}_s^0 = \{21, 23\}$ and velocity assignments*



(a) 1a (b) 1b

Figure 4.4

Figure 4.5: Gradient descent procedure visualised.

$v_s^0 = (0.285, 0.352, 0.107)$. *The balancing procedure is visualised in Figure 4.4.*

First, the gradient is computed using Equation (4.19)*, which results in* $\nabla C_s(v_s) = (1.0, 2.2)$. *We find the candidate assignments given in Table 4.2. From these candidate assignments, the feasible lockage assignment with minimum cost is* $\{22, 27\}$. *Thus, vessel* s *is reassigned accordingly. The new cost of this assignment is* 1.5337*, while the cost of the initial assignment was* 2.9871.

Table 4.2: List of candidate lockage assignments

	Lockage Assignment	$\hat{\gamma}$	Cost
K_1^c	$\{21, 24\}$	0.054	1.9397
K_2^c	$\{21, 25\}$	0.080	1.6583
K_3^c	$\{21, 26\}$	0.096	1.6649
K_4^c	$\{21, 27\}$	0.107	1.9281
K_5^c	$\{22, 27\}$	0.113	**1.5337**
K_6^c	$\{22, 28\}$	0.115	2.2802
...	...	...	

4.4 Computational Experiments

4.4.1 Introduction to the Amsterdam Rhine Canal case

To evaluate the performance of our heuristic, we perform computational experiments on a dataset extracted from an inland waterway network, located at the split of Rhine and the Amsterdam-Rhine Canal (The Netherlands). The network contains three locks, connected by river segments as visualized in Figure 4.6.

The river network connects the *Rhine*, which flows from East to West, with the *Amsterdam Rhine Canal*, which flows from South to North. Furthermore, the Amsterdam Rhine Canal intersects with both the Rhine, and the *Nederrijn*, which flows parallel to and north of the Rhine.

There are three locks:

- The *Prinses Irenesluizen* (referred to as *Northern Lock*). This lock consists of two separate chambers that process vessels independently. The distance from the northern entry point of the river system to the Northern Lock is 22.13 km.
- The *Prins Bernhardsluizen* (referred to as *Eastern* Lock). This lock is used far less than the other two locks and thus consists only out of one chamber. The distance from the Eastern to the Northern

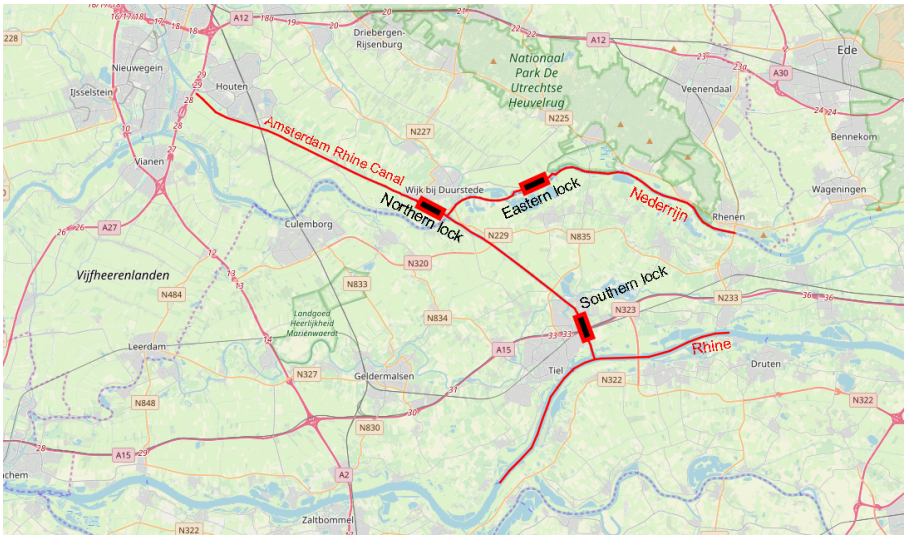


Figure 4.6: Map visualization of the Amsterdam Rhine Canal case.

Lock is 10.22 km, while the distance from the Eastern Lock to the Eastern exit point is 19.576 km.

- The *Prinses Marijkesluizen* (referred to as *Southern Lock*) forms the connection point between the Rhine and the Amsterdam Rhine Canal. This lock consists of two independent chambers. The distances from Southern lock to Eastern and to Northern Locks are 23.11 km and 16.84 km, respectively. Furthermore, the western and eastern entry points into the river system are located at 12.936 km and 13.896 km, respectively.

4.4.2 Preprocessing of the dataset

The heuristic was tested on real-life scenarios extracted from the Automatic Identification System (AIS). We make use of real-life AIS data from the period 2 January 2019 – 31 March 2019, split in instances that

cover a single day on the river. This means that there are 84 observation days/instances.

The dataset contains vessel characteristics, such as vessel ID and type, and the real-time location of the vessel. These locations are given in the form of GPS coordinates and are recorded every two minutes. In the following section, we give a description of the preprocessing steps, applied to the dataset.

First, a graph was constructed of the specified area with sequences of nodes forming the river segments. Second, GPS positions of individual vessels have been mapped onto the network to track their movements. For each vessel, the first and the last node contained in the dataset are defined as entry and exit points. The timestamps of first and last occurrence are used as arrival time and deadline, respectively. A vessel that stops at a point in the network and continues its journey after some time period (e.g., after lunch break) was replaced by two. Then, we computed the trajectory for each vessel in order to define the sequence of locks (and their directions) that it has to traverse. Lastly, the velocity limits were extracted by taking an upper quantile over the measurements of all velocity for each type of vessel.

The parameters defining a lock have been attained in the following way. First, for each lock, a set of nodes located at the position of the lock, were specified. Then, the entry and exit times of vessels on these set of nodes were tracked and used to estimate capacity and processing time of lockages. Vessels that were subject to strong noise have been removed from the estimation process. Finally, the median has been taken over all available observation days for each lock individually.

The deadline for each vessel is determined by the physical time the vessel spent in the system. This enables us to perform the heuristic on the data and estimate the potential, real-life improvement of the system. On the other hand, the estimates for a lock's parameter are subject to high variance. There exist situations in which vessels pass the river segment very quickly by facing a low amount of congestion and a low processing time of locks. By setting the lockage duration

equal to the median of all representative vessels in the real system, the original vessel movements are considered infeasible by the model. As a result, there exists a subset of vessels in every instance that will not be able to pass the river section given these specifications in the theoretical model.

An overview of the average number of vessels processed by each lock on a daily basis is given in Table 4.3. Furthermore, the estimates of the lock parameters are given in Table 4.4.

Table 4.3: Average number of vessels processed by each lock on each day.

Day	**Total**	**East**	**North**	**South**
Monday	131.92	27.58	110.33	27.58
Tuesday	143.17	30.25	122.42	95.50
Wednesday	142.62	25.85	120.77	97.69
Thursday	150.69	28.69	125.76	98.46
Friday	120.17	20.83	83.33	83.33
Saturday	77.08	8.42	64.25	57.75
Sunday	55.00	4.47	45.83	40.50
Total	**117.92**	**98.98**	**80.15**	**21.04**

Table 4.4: overview of all lock parameters.

Lock	**Processing Time (T_l)**	**Capacity (C_l)**	**# Chambers (B_l)**
Eastern	22	3	1
North	23	4	2
South	22	3	2

4.4.3 Limitations of the real-time data

Working with real AIS data has its limitations. As these limitations have an impact on how the numerical results presented in this section should be interpreted, we first take the time to discuss them in more detail.

Due to inaccuracy and missing information, it turned out impossible to accurately determine the waiting and processing time of vessels at the locks. The observed waiting time in front of the locks is unreasonably long. This suggests that skippers also use these areas for their required breaks. This makes it hard to distinguish waiting time related to congestion and time taken for breaks. As we currently do not assume that skippers can take a break when solving the velocity optimization problem, our algorithm uses the additional available time to reduce the velocity on certain river segments. In the results, this will lower total fuel consumption compared to the real-life benchmark, which would give an unfair advantage for our algorithm. Note, however, that these breaks could easily be incorporated in the model formulation if skippers would declare and plan these breaks ahead.

Additionally, we fixed the lockage duration (i.e., the time it takes to process a batch of vessels at the lock) to its median value. The lockage durations for individual vessels in the data, however, deviate substantially from our fixed value. This can clearly be observed in Figure 4.7, which provides a graphical representation of the realised processing times for the northern lock on January 31, 2019. Similar assumptions were also made when determining the maximum sailing velocity on each river segment. As a result, our mathematical model formulation has a higher minimum time to complete a planned itinerary for some vessels than the original travel time, and therefore larger then the imposed deadline. This means that, by construction, the vessels would be flagged as infeasible to schedule by our solution procedure. These observations would thus give an unfair advantage to the real-life benchmark data.

These limitations imply that policy makers should be careful when relying solely on AIS data to support their decision making process. To allow a fair quality assessment of the proposed heuristic, we therefore solely compare the aggregated fuel consumption of the schedule with the optimal solution generated by the MILP presented above.

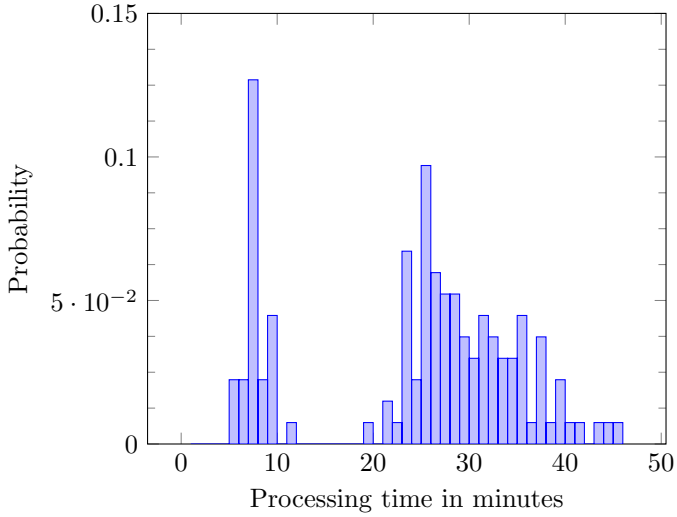


Figure 4.7: Realisation of processing time on January, 31, 2019.

4.4.4 Peformance of the local search heuristic

To evaluate the performance of our local search heuristic, we compare the results to the solutions obtained by the MILP. All numerical experiments have been executed on a Intel(R) Xeon(R) Gold 6126 CPU 2.60GHz (2 CPUs), 2.6GHz with 6144MB RAM memory.

The MILP was implemented in CPLEX 12.6.3.0. We limited the running time of the MILP solver to 7200 seconds (2 hours) per instance. In order to limit the size of the instances, only a fraction of all vessels have been sampled for each day of observation. To ensure the working of the MILP without further modifications, we restricted ourselves to instances that are feasible given the given the parameter settings, as discussed above.

For the heuristic, we apply the initialization phase after which the balancing operator is called at most four time. Extensive numerical experiments on the full fleet of vessels have shown that the total fuel

Table 4.5: Computational performance of the local search heuristic in comparison to the MILP.

# vessels	Cost ratio	Computation time (s)		MILP solution
		MILP	Local Search	found
10	1.046	141.57	2.53	100%
20	1.015	5008.91	4.68	100%
30	0.991	5608.60	7.19	100%
40	0.983	6822.69	9.38	85.7%
full fleet (avg. 117.92)	—	—	17.62	0%

consumption is not decreasing further when increasing the number of iterations of the balancing operator.

The results of the experiments are summarized in Table 4.5. The columns specify the average cost ratio of the heuristic schedule compared to the MILP solution, as well as the running times for both methods. Furthermore, the last column specifies on how many instances the MILP solver found a feasible solution (not necessarily an optimal one). The solver is able to find exact solutions only for instances smaller than 40 vessels, which is far below a realistic instance size. At the same time, the heuristic finds schedules that are of high quality. On the entire fleet, it took on average 17.62 seconds for the local search heuristic to produce a schedule for all vessels. From these 17.62 seconds, 7.14 seconds were required to produce the initial schedule and 10.48 additional seconds were used to apply the four iteration of the velocity balancing operator. The balancing procedure was able to decrease the fuel consumption of the initial schedule by 4.05%. The quality and running time are therefore reasonable for lock planners and skippers to be used in practice.

4.4.5 Efficiency versus service level

The limitations of the real-life dataset imply that only a fraction of all vessels can actually trespass the river segment within their indicated

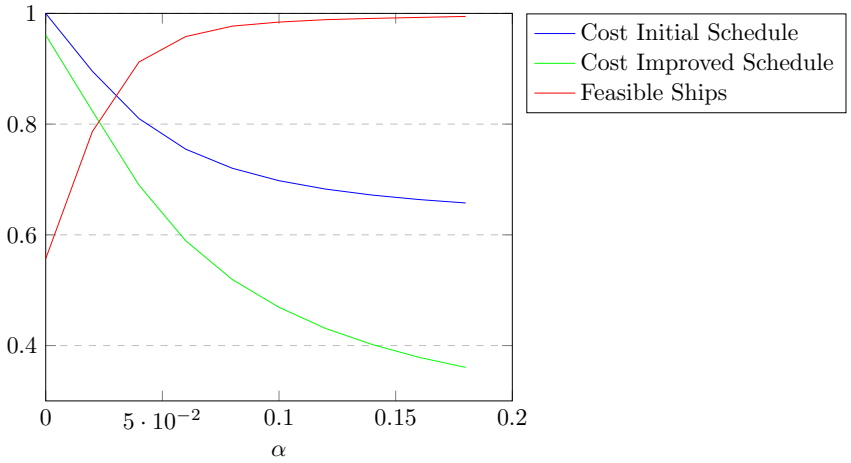


Figure 4.8: Impact of deadline tightness on solution quality.

deadlines, as discussed in Section 4.4.3. In this Section, we investigate the trade-off between fuel consumption and service level, defined as an extension of the deadline by $\alpha\%$. The effect of α on the aggregated fuel consumption is plotted in Figure 4.8.

For each infeasible vessel (i.e., a vessel that cannot be scheduled to arrive before its deadline because of velocity restrictions and fixed lockage durations), we advice a velocity such that the vessel's arrival time is as early as possible.

When the value of α increases, the balancing operator becomes less restrictive as service at each lock can be postponed further for each vessel. As a result, the average advised velocity will also be lower. The strong decay of the fuel consumption (seen in Figure 4.8) implies the increase in sailing time for vessels can be used as a regulatory instrument by policy makers to decrease the total fuel consumption and therefore the total emission considerably.

4.4.6 Impact of lock efficiency on aggregated fuel consumption

Due to the variation in the processing time of the locks in real-life, the question that arises is how the aggregated fuel consumption is influenced by the lock parameters. The aggregated fuel consumption and the fraction of infeasible vessels are plotted against the difference in lockage duration, denoted by Δ_p, in Figure 4.9.

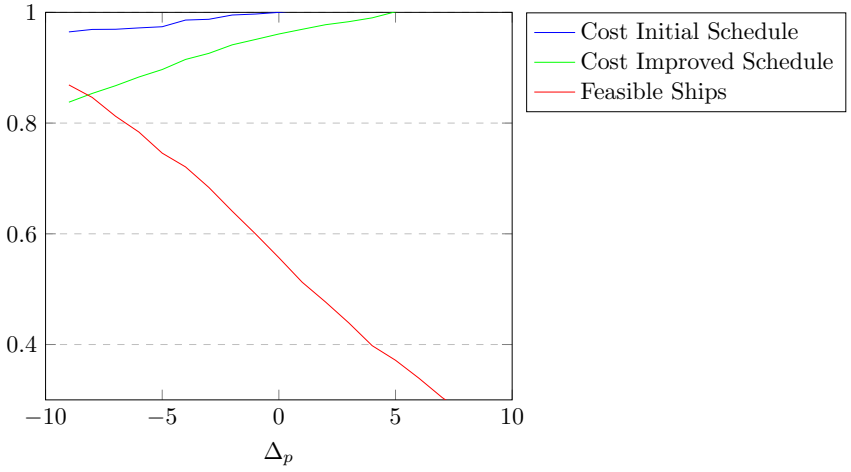


Figure 4.9: Impact of change in lockage duration on solution quality.

Note that a decreasing lockage duration is correlated with decreasing costs and increasing fraction of feasible vessels. A lower lockage duration implies that the local search has more flexibility, and therefore the gap between the initial schedule and an improved schedule increases. If it is possible to improve the efficiency of locks by advising regulators, it would translate to a direct increase in efficiency and a reduction of aggregated fuel consumption of all vessels.

4.5 Conclusion

In this work, we introduced a local-search-based heuristic for the lock scheduling problem on a river network and extended the known mathematical programming formulation of [12] to incorporate more realistic features and conditions of the problem. The heuristic algorithm and a straightforward implementation of the mathematical formulation were compared using computational experiments on real AIS dataset from a section of the Dutch river network. It is evident that the heuristic is able to find schedules for large instances of the lock scheduling problem in very reasonable computation time. It is needless to say that the mathematical programming formulation is incapable of tackling the large real-life instances. Thus, the suggested heuristic has a potential to become a tool for the lock operators and vessel skippers to regulate lock congestions and adjust sailing velocities in order to optimize the fuel consumption and CO_2 emission. Furthermore, we have evaluated the effect of varying the service level and the processing time of lockages on the total fuel consumption of the vessels.

5

Velocity Optimisation under uncertainty

Adapted from: M. Buchem, J. A. P. Golak, and A. Grigoriev, "Vessel velocity decisions in inland waterway transportation under uncertainty," *European Journal of Operational Research*, 2021.

5.1 Introduction

When a skipper approaches a lock today, his information is very limited. The skipper is fully dependent on the operator of the lock to give him an estimate on the time window in which he can enter the lock. Just before the lock is likely to be cleared, he receives a call from the lock operator. On the other hand, the lock operator receives his information directly from approaching vessels. After communication with industry experts of inland waterway transportation[1], the following factors have been identified which influence the processing times of a lockages, see [16].

- Unexpected appearance of new vessels contributing to suboptimal space allocation and unnecessary maneuvering near the lock;
- Pleasure boats which are often not recorded by AIS systems and thus more difficult to track;
- Environmental conditions, e.g. strong winds that prolong the docking procedure of vessels;
- Type and size of ships in the lock;
- Experience of skippers and lock operator;
- Other technical issues.

Note that these factors are only partly observable by the lock operator, which may lead to a very uncertain estimate of the processing time of a specific vessel. A sample of such a lock distribution has been recovered from AIS data and depicted in Figure 5.1.

From a skipper's point of view, not taking this uncertainty into account may lead to waiting time at the lock. By slowing down before, the skipper could traverse the lock at the same time while using less fuel. Fuel consumption plays an important role in two ways: (1) fuel

[1] Communicated to us by our industrial partner Trapps Wise. B.V.

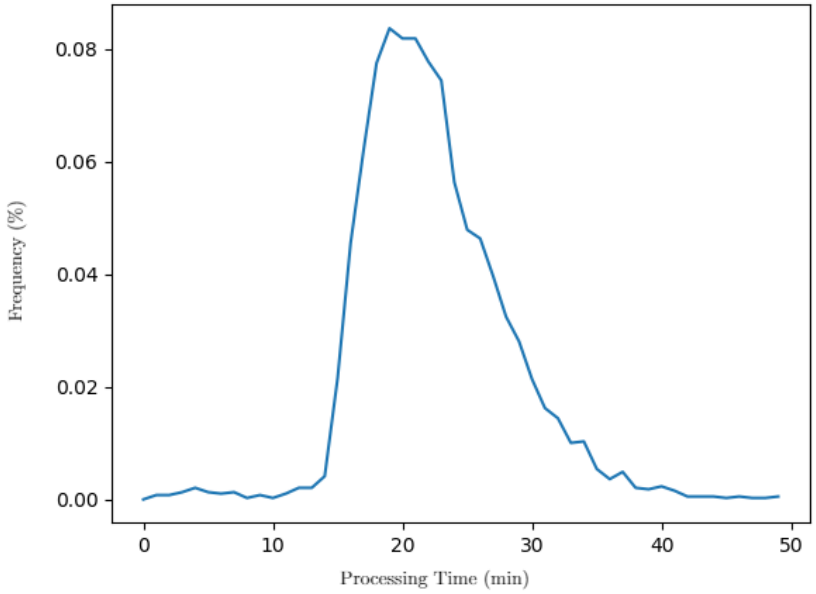


Figure 5.1: Processing time distribution of the Prinses Irenesluizen from 01.03.2019 to 31.03.2019

consumption is directly linked to emission and can therefore be used as an objective to make freight transportation more sustainable and (2) lower fuel consumption implies monetary savings and is a more attractive measure for the skipper. Therefore, fuel consumption is used as an objective hereafter. Equipped with adequate algorithms, a digitalization of locks can lead to a smarter choice of velocity for skippers and thus a reduction in fuel consumption leading to economic and ecological profits. However, the success of implementing such digital system is dependent on participants accepting it, e.g. a system that suggests a high variation of velocity changes may be faced with immediate rejection from skippers.

In this chapter, we focus on the operation of a single vessel on an inland waterway with uncertainty at the lock. We introduce a mathematical optimization model to find the most fuel efficient choice of velocities such that the vessel traverses the waterway within a set time window. We then introduce two solution methods that give advice to skippers for decision making based on an optimal and a heuristic solution to the mathematical optimization problem. The two approaches differ in their efficiency and simplicity for skippers.

Related Work In maritime shipping, container vessels are following a route of ports at which they load and unload cargo. While vessels prefer to slow stream along their route, they are assigned time windows by each individual port. Missing such time window would lead to delay and more uncertainty to when the vessel can be served. This delay may lead to missing deadlines and inefficient speeding behavior of skippers. While the majority of publications considers deterministic service times and implements aspects such as port disruptions like in [39], there has been an increase in literature considering uncertainty in port service times. This uncertainty may lead to unexpected delays if not incorporated into optimization models. [40] consider the ship route schedule design problem with uncertain service times in which late arrivals are not allowed and give a non-linear stochastic programming formulation. [41] relax this assumption and robust schedule design with late arrivals under penalty costs is considered. Most recently, [42] consider a dynamic approach to the velocity optimization in liner shipping by finding a velocity policy, i.e. a rule that for every port and realisation of the finishing time of the ship at the port, states the optimal velocity to travel the next segment with.

Our contribution Following the trend in the maritime shipping research, we introduce the concept of uncertainty in inland waterway transportation. We focus on the operation of a single vessel on an inland waterway with uncertainty at the lock. In Section 5.2 we intro-

duce the mathematical framework and optimization model. In Sections 5.3 and 5.4 we present a dynamic program to obtain a close to optimal solution and a simple heuristic, respectively. Computational experiments analysing the possible savings obtainable by these solution methods and the trade-off between efficiency and simplicity is discussed in Section 5.5. Finally, some concluding remarks are given in Section 5.6.

5.2 Notation and Preliminaries

We consider a waterway with a single lock and two vessels approaching the lock as depicted in Figure 5.2. We assume that we give velocity advice to vessel a, which is approaching the lock from upstream. A second vessel, vessel b, is approaching the lock from downstream.

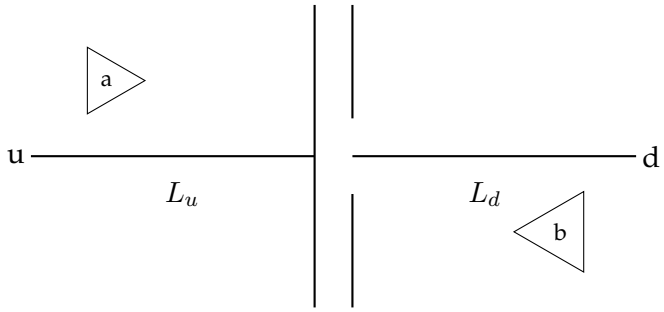


Figure 5.2: The setup of waterway, lock and vessels

The waterway is defined by the lengths of the two segments. Let L_u and L_d be the length of the upstream and downstream segment, respectively. We assume that the lock has unit capacity and a deterministic lockage time, i.e. the time it takes the lock to move from upstream to downstream and vice versa, denoted by P. Furthermore, we assume that the lock operator processes vessel according to first-come-first-serve with ties being broken in favour of vessel a.

Vessel a needs to arrive at the end of the waterway, i.e., at the end of the downstream, before its deadline D. Based on experience, the skipper of vessel a knows the time it takes to dock and undock inside the lock chamber. Therefore, we assume that this time is deterministic and given by η_a. Thus, the total time vessel a spends in the lock when being processed from upstream to downstream is given by $\tau_a = P + \eta_a$.

Vessel b is characterized by its arrival time at the lock, r, and its docking and undocking time, η_b. Today, each vessel is required to be equipped with a automatic identification system (AIS), which provides availability to positional data. Thus, we assume that the arrival time of vessel at the lock is known. On the other hand, the time to dock or undock the vessel is known to the skipper of vessel a. Once vessel b has been processed, the lock operator sends out a notification to the skipper of vessel a. Thus, we assume throughout the chapter that η_b is given by a discrete random variable. Hence, the lock processing time of vessel b defined as $\tau_b = P + \eta_b$ with $p_i = Prob(\tau_b = c_i)$ for $i = 1, \ldots, n$. The system is fully defined by the tuple $\mathcal{I} = (\eta_a, \eta_b, P, r, D, L_d, L_u)$.

The aim of vessel a is to cross the waterway at minimum expected total fuel consumption, which is derived as follows. The resistance R of a vessel is proportional to v^2, i.e. $R(v) = const \cdot v^2$, where the constant is related to actual vessel properties. The fuel consumption per unit of time is proportional to the required power $P(v) = R(v)v$. Therefore, the fuel consumption per unit of time is defined as $F^t(v) = const \cdot v^3$, where the constant related to vessel and engine fuel properties. The required time t for sailing distance d at velocity v equals $t = d/v$. This results in total fuel consumption required for sailing distance d at velocity v as $F(v, d) = F^t(v)t = const \cdot dv^2$. The cost for a skipper is then characterised by the fuel consumption multiplied by the cost per unit of fuel. This is line with conventions from related literature, see [12].

Furthermore, we state the following assumption without loss of generality:

- Vessel a has direct and full control of velocity at every time unit,

- The lock is positioned to the vessel that arrives first, i.e., there is no empty lockage required to process the first vessel,
- Throughout this article, we assume that the ship specific constants, as well as the cost per ton of fuel is equal to one. This implies that fuel consumption and cost are the same and defined by $F^t(v) = v^3$ and $F(v, d) = dv^2$ expressed in unit of time and distance, respectively.
- Vessels can change from one velocity to another instantly,
- Vessel a and b are facing a conflict situation meaning that the optimal arrival time of vessel a if vessel b was not on the water way lies in the interval $[r, r + c_n]$.

We refer to a solution to the problem as a *velocity policy*. A velocity policy v is a decision rule that chooses a velocity for each time t based on the knowledge available up to time t and all the a-priori distributional information. The information given by the lock is defined by $\omega(t)$, which indicates whether the lock is unavailable to vessel a or the time elapsed since the event of processing vessel b. Let t_b denote the time at which vessel b exits the lock, then:

$$\omega(t) = \begin{cases} -1, & \text{if } t < t_b; \\ t - t_b, & \text{if } t \geq t_b. \end{cases}$$

Here it is important to note that $\omega(t) = -1$ for $t < r$ as we only consider conflict cases.

Formally, a velocity policy v defines a velocity $v(t, \omega(t))$ for all t and all realizations of $\omega(t)$. A velocity policy needs to satisfy the following conditions. The velocity needs to satisfy $v_{min} \leq v(t, \omega(t)) \leq v_{\max}$ for all t and $\omega(t)$, where $v_{\max}$ is the maximum velocity allowed on the waterway and $v_{\min}$ is the minimum velocity of vessel a. Furthermore, v needs to ensure that vessel a crosses the whole waterway. We define a random variable $T(v)$ to be vessel a's arrival time under policy v. In order for the policy to be feasible $T(v)$ has to satisfy

$$\int_0^{T(v)} v(t, \omega(t))dt = L_u.$$

The total fuel consumption of vessel a is divided in the two segments of the waterway. On the upstream segment of the waterway the expected total fuel consumption of policy v is given by

$$\mathbb{E}\left[\int_0^{T(v)} v(t, \omega(t))^3 dt\right].$$

Given a policy v, when arriving at the lock at time $T(v)$, vessel a can either enter the lock right away or faces a random waiting $q(T(v))$. Formally, $q(T(v))$ is defined as

$$q(T(v)) = \begin{cases} \max\{r + \tau_b - T(v),\ 0\}, & \text{if } r < T(v) < r + c_n; \\ 0, & \text{otherwise.} \end{cases}$$

Given an arrival at the lock at time $r + c_i < r_a$ for some i we have that $q(r_a) = r + c_j - r_a$ with probability p_j for $j > i$ and $q(r_a) = 0$ with probability $\sum_{k=1}^{i} p_k$.

Note that velocity is non-negative and fuel consumption is convex on non-negative real numbers. This implies constant velocity on a segment has lower cost than increasing and decreasing velocity on arbitrary parts on this segment. Thus it is optimal to have a constant velocity after the vessel leaves the lock. Hence, the total expected fuel consumption on the downstream under policy v is defined by

$$\mathbb{E}\left[\left(\frac{L_d}{D - T(v) - q(T(v)) - \tau_a}\right)^2 L_d\right].$$

An optimal velocity policy is a solution to the following mathematical

program.

$$\min_{v(.)} \mathbb{E}\left[\int_0^{T(v)} v(t,\omega(t))^3 dt + \Big(\frac{L_d}{D - T(v) - q(T(v)) - \tau}\Big)^2 L_d\right] \quad (5.1)$$

$s.t.$

$$\int_0^{T(v)} v(t,\omega(t))dt = L_u \quad (5.2)$$

$$v_{min} \leq v(t,\omega(t)) \leq v_{max} \quad (5.3)$$

The following example illustrates the problem and a simple velocity policy is described.

Example 5.2.1. *We consider a waterway with a total length of* $2km$ *and a centrally located lock, i.e.* $L_u = L_d = 1$. *Vessel* a *needs to cross the waterway within* 6 *time units, i.e.* $D = 6$ *where one time unit corresponds to* 15 *minutes. The lock processing time of vessel* a *is* 15 *minutes meaning that* $\tau_a = 1$. *Vessel* b *arrives at the lock at time* 0 *and has a lock processing time* τ_b *with* $Pr[\tau_b = 1] = Pr[\tau_b = 2] = Pr[\tau_b = 3] = \frac{1}{3}$. *The optimal arrival time at the lock when vessel* b *is not present is at* $t^* = 2.5$. *Hence, the presence of vessel* b *implies a conflict for vessel* b *and velocity updating may lead to fuel savings.*

From a skipper's perspective a natural approach considering possible updating of velocity may be the following. Given the optimal arrival time $t = 2.5$ *without conflicting vessel* b *a skipper may naturally follow the velocity* $\frac{1}{2.5}$ *until time* 2. *If the lock is finished at time* 1 *or time* 2 *the skipper simply continues with the same velocity. If the lock is not finished then the skipper changes the velocity such that he arrives at the lock at time* 3.*This simple dynamic velocity policy yields a total expected fuel consumption of approximately* 0.342. *Not taking into account the uncertainty and simply choosing a constant velocity with arrival time* 2.5 *yields a total expected fuel consumption of* 0.35. *Hence, even a simple dynamic policy leads to fuel savings of approximately* 2.3%. *In Figure 5.3 this dynamic approach is visualized. In the boxes the position at*

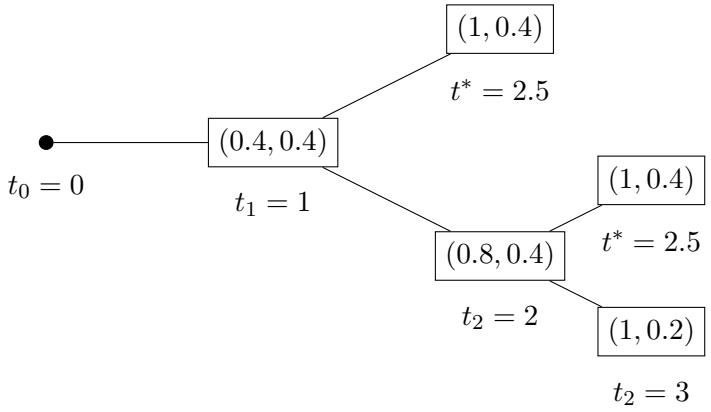


Figure 5.3: Visualization of simple dynamic solution to Example 5.2.1

the time of this specific node and the velocity driven from the previous time to this time are shown.

5.3 Dynamic Programming

In the following section we derive two structural results on optimal velocity policies. We use these results to derive a recursive formulation of finding an optimal velocity policy. Using this recursive formulation we introduce a dynamic programming algorithm that finds a close-to-optimal velocity policy due to discretization of the feasible decisions a velocity policy may choose.

Once the skipper of vessel a received the information that vessel b has been processed, the convexity of fuel consumption implies it is optimal to stay at a constant velocity until the lock. Formally, this is described in the following lemma.

Lemma 5.3.1. *An optimal velocity policy v^* chooses a constant velocity for the remaining distance whenever the lock has finished processing vessel b.*

This means that for any t with $\omega(t) > 0$, the following holds:

$$v(t, \omega(t)) = \begin{cases} v(t - \omega(t), 0), \\ \quad \textit{if vessel } a \textit{ has not reached the lock at time } t; \\ 0, \\ \quad \textit{if vessel } a \textit{ has reached the lock at time } t \textit{ or before.} \end{cases} \tag{5.4}$$

Assume that a receives this information at time t and it is at position X. Thus, it needs to travel an additional $L_u - X$ distance to reach the lock. At this point the cost is uniquely defined *Fixed Cost* (hereafter: FC), which is formulated in (5.5). Note that this cost can be solved analytically by first order conditions.

$$FC(X,t) = \min_{v_{min} \leq v \leq v_{max}} v^2(L_u - X) + \frac{L_d^3}{(D - (t + (L_u - X)/v + \tau_a))^2} \tag{5.5}$$

The position of vessel a at time t is based on the history of velocities and formally given by $\int_0^t (s, -1)ds$. Recall, that the probability that the lock is still processing vessel b at time t is given by $\tilde{p}_t$ and that the probability that the lock finishes processing b at time t is given by p_t. Lemma 5.3.1 implies that an optimal velocity policy has to satisfy equation (5.4). Thus, the total expected fuel consumption can be rewritten as follows:

$$\begin{aligned} &\mathbb{E}\left[\int_0^{T(v)} v(t, \omega(t))^3 dt + \Big(\frac{L_d}{D - T(v) - q(T(v)) - \tau}\Big)^2 L_d\right] \\ &= \int_0^{T(v)} \tilde{p}_t v(t, -1)^3 dt + \int_0^{T(v)} p_t FC(X(t), t) dt \end{aligned} \tag{5.6}$$

The first part is the expected fuel consumption on the upstream before the lock finishes processing vessel b and the second part is the expected fuel consumption on the upstream and downstream after the lock has finished processing vessel b.

Next, because the processing time of b is discrete, an optimal policy changes velocity only at possible realizations of τ_b. On intervals, between possible realisation an optimal policy chooses a constant velocity. From a practical point-of-view this means that the skipper only has a limited number of times at which he may have to change the vessel's velocity. This is described by the following lemma.

Lemma 5.3.2. *An optimal velocity policy v^* only changes its velocity when new information about the processing time of vessel b is available, i.e. at time $t \in \{0, r + c_1, \ldots, r + c_n\}$.*

Using Lemma 5.3.2 we can further reformulate the total expected fuel consumption from Equation (5.6). For notational purposes we reinterpret the idea of a velocity policy as follows: at decision moment i a velocity policy advises the skipper to be at position $X(r + c_{i+1})$ at time $r + c_{i+1}$ if the lock has not finished processing vessel b and a final constant velocity $v(i)$ if the lock has just finished processing vessel b. Note that due to Lemma 5.3.1 these final velocities can be pre-computed for all $X(r + c_i)$ using Equation (5.5) and a velocity policy is uniquely defined by $X(i)$ for $i = 1, \ldots, n$. Given a velocity policy, the vessel travels with velocity $X(r + c_1)/(r + c_1)$ from time 0 to time $r + c_1$ and with velocity $X(r + c_{i+1}) - X(r + c_i)/(c_{i+1} - c_i)$ from time $r + c_i$ to time $r + c_{i+1}$. Then, Equation (5.6) can be reformulated as follows:

$$\int_0^{T(v)} \tilde{p}_t v(t,-1)^3 dt + \int_0^{T(v)} p_t FC(X(t),t)dt =$$
$$\frac{X(r+c_1)^3}{(r+c_1)^2} + \sum_{i=1}^{n-1} \Big(\tilde{p}_{r+c_i} \cdot \frac{(X(r+c_{i+1}) - X(r+c_i))^3}{(c_{i+1}-c_i)^2} \Big) +$$
$$\sum_{i=1}^{n} p_i FC(X(r+c_i), r+c_i) \tag{5.7}$$
$$= \frac{X(r+c_1)^3}{(r+c_1)^2} + \sum_{i=1}^{n-1} \Big(\tilde{p}_{r+c_i} \cdot \frac{(X(r+c_{i+1}) - X(r+c_i))^3}{(c_{i+1}-c_i)^2} +$$
$$p_i \cdot FC(X(r+c_i), r+c_i) \Big) + p_n \cdot FC(X(r+c_n), r+c_n)$$

We next show that an optimal velocity policy can be found by recursively minimizing the expected total fuel consumption after time $r + c_i$ for all $i = n, \ldots, 1$ for all possible positions at time $r + c_i$ and, finally, after time 0 given that we start at position 0, i.e. the start of the downstream segment, at time 0. The expected fuel consumption after time $r + c_i$ can be expressed in two parts: (1) the expected fuel consumption if the lock is still processing vessel b at time $r + c_i$ and (2) the expected fuel consumption if the lock has finished processing vessel b at time $r + c_i$. Let $g_i(X(r + c_i))$ be the minimum expected total fuel consumption after time $r + c_i$ when the policy advises to be at position $X(r + c_i)$ at time $r + c_i$ (given that the lock has not finished processing vessel b before). Let q_i define the probability that vessel b is processed after c_i time steps, given it is not finished after c_i time units, i.e. $q_i = Prob(\tau_b = r + c_i | \tau_b \geq r + c_i)$.

For $i = n$, the probability that the lock is still processing vessel b is zero and the fuel consumption only consists of part (2). Hence,

$$g_n(X(r + c_n)) = p_n \cdot FC(X(r + c_n), r + c_n). \tag{5.8}$$

For $1 \leq i \leq n-1$ we have:

$$g_i(X(r+c_i)) = (1-q_i) \cdot \min_{X \in A_i(X(r+c_i))} \left(\frac{(X - X(r+c_i))^3}{(c_{i+1}-c_i)^2} + g_{i+1}(X) \right) + q_i \cdot FC(X(r+c_i), r+c_i). \tag{5.9}$$

Here, $A_i(X(r+c_i)) = \{X | X(r+c_i) \leq X \leq L_u$ and $v_{min} \leq \frac{X - X(r+c_i)}{c_{i+1}-c_i} \leq v_{\max}\}$ is the set of feasible positions at time $r + c_{i+1}$ given position $X(r+c_i)$ at time $r + c_i$. Then, for $i = 0$ we know that the vessel starts at position 0, so we have:

$$g_0 = \min_{X \in A_0} \{ \frac{X^3}{(r+c_1)^2} + g_1(X) \}, \tag{5.10}$$

where $A(0) = \{X | 0 \leq X \leq L_u$ and $v_{min} \leq \frac{X}{r+c_1} \leq v_{\max}\}$.

The recursive expression implies that the problem exhibits subproblem optimality and that it can be solved by backwards recursion. However, even for only two realizations of τ_b, the first order condition of the sub problems involves finding roots of polynomials of degree at least 6. It is likely that no analytical solution exists. Consequently, we propose a numerical approximation by defining the following dynamic program while discretizing the distance between starting point and lock described in Algorithm 7. The velocity policy corresponding to the

result of the dynamic program can be found using backtracking.

Algorithm 7: Close-to-optimal Dynamic Updating.

Input: $\mathcal{I}$, ε
Output: Minimal cost of traversing the waterway.
Preliminaries;
Let $K := \lceil L_u/\varepsilon \rceil$;
$X_j = j\frac{L_u}{K}$ for all $j = 0 \ldots, K$;
M is $n \times K$ matrix;
Compute $FC(X_j, r + c_i)$ for all $j = 0, \ldots, K$ and all $i = 1, \ldots, n$;
$q_i = Prob(\tau_b = r + c_i | \tau_b \geq r + c_i)$;
Initialization;
$M(n, X_j) = p_n \cdot FC(X_j, r + c_n)$ for all $j = 0, \ldots, K$;
Recursion;
$M(i, X_j) =$
$(1 - q_i)\ \min_{X_{j'} \in A} \left\{ \frac{(X_{j'} - X_j)^3}{(c_{i+1} - c_i)^3} + M(i + 1, X_{j'}) \right\} + q_i \cdot FC(X_j, i)$
for $i = n - 1, \ldots, 2$, for $j = 1, \ldots, K$;
$A_i(X(r + c_i)) = \{X | X(r + c_i) \leq X \leq L_u$ and $v_{min}\}$;
Return $\min_{X_j \in A(0)} \left\{ \frac{X_j^3}{(r + c_1)^2} + M(2, X_j) \right\}$;

Theorem 5.3.1. *For any $\varepsilon > 0$, Algorithm 7 returns a velocity policy v with total expected fuel consumption $f^{\varepsilon}(v)$ such that*

$$f^{\varepsilon}(v) - f(opt) \leq \varepsilon f(opt)$$

in time $O(nL_u^2/\epsilon^2)$, where opt is the optimal velocity policy.

Proof. Let C be the running time of *FC*. The initialization takes $O(CL_u/\varepsilon)$ time. then the recursion computes the minimum over all X'_j, where $j' = 1, \ldots, K$ for all $i = 1, \ldots, n$ and X_j, where $j = 1, \ldots, K$. Thus, the

recursion takes $O(Cn(L_u/\varepsilon)^2$. Thus, the running time of the dynamic program is in $O(CL_u/\varepsilon + Cn(L_u/\varepsilon)^2) = O(n(L_u/\varepsilon)^2)$.

Due to the discretization of the possible positions of the vessel, we know that for any $i \in \{0, \ldots, n-1\}$ the dynamic program finds a position X that deviates from the optimal position X^* by at most ε. Due to the convexity of the fuel consumption function, we know that $|M(X,i) - M(X^*,i)| \leq \varepsilon M(X^*,i)$. From this it follows that

$$f^{\varepsilon}(v) - f(opt) \leq \varepsilon f(opt).$$

□

In the following example we present the optimal velocity policy for Example 5.2.1.

Example 5.3.2. *Recall the instance described in Example 5.2.1 given by the following parameters:*

- $L_u = L_d = 1$
- $D = 6$
- $r = 0$
- $Pr[\tau_a = 1] = 1$
- $Pr[\tau_b = 1] = Pr[\tau_b = 2] = Pr[\tau_b = 3] = \frac{1}{3}$

In Example 5.2.1 we showed a natural dynamic velocity policy and its fuel consumption for this example. Using Algorithm 7 we can find the optimal dynamic solution approach for this instance. Figure 5.4 shows the optimal dynamic solution. In the boxes the position at the time of this specific node and the velocity driven from the previous time to this time are shown. This optimal dynamic policy yields a total fuel consumption of approximately 0.3377 *implying fuel savings of* 3.5% *compared to the naive fixed arrival approach not taking into account any uncertainty and* 1.24% *compared to the simple dynamic velocity policy.*

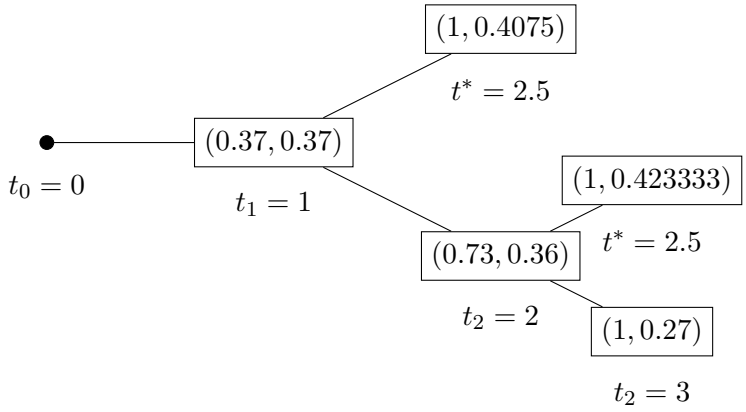


Figure 5.4: Visualization of optimal dynamic solution to Example 5.2.1

5.4 Fixed arrival policies

In this section we introduce a simple class of policies called *fixed arrival* policies. The class of *fixed arrival* policies is attractive for skippers because it advises a single velocity on the upstream segment of the waterway without changes. A policy that offers highly erratic advises at each time step may lead to rejection by skippers. In a fixed arrival policy v the skipper of vessel a is advised a constant velocity up to some deterministic arrival time $T(v)$.

A fixed arrival policy is feasible if $T(v) < D - \tau_a$, meaning that there is still time available to cross the downstream section. Formally, a fixed arrival policy is defined as follows.

Definition 5.4.1. *A velocity policy v is called a fixed arrival if for some $v_{\min} \leq \hat{v} \leq v_{\max}$ we have*

$$v(t, \omega(t)) = \hat{v} \text{ for all } t \text{ and } \omega(t). \tag{5.11}$$

Next, we show how to reformulate Equation (5.1) for a fixed arrival policy v.

Lemma 5.4.1. *Let v be a fixed arrival policy with with constant velocity $\hat{v}$. Then, vessel a reaches the lock with probability 1 at time $T(\hat{v}) = L_u/\hat{v}$. And the total expected fuel consumption of policy v is*

$$\frac{L_u^3}{T(\hat{v})^2} + \sum_{i=0}^{n} p_i \left[\Big(\frac{L_d}{D - (\max\{r_a, r + c_i\} + \tau_a)} \Big)^2 L_d \right]. \tag{5.12}$$

Proof. In terms of Equation (5.1) the total expected fuel consumption of policy v is

$$\mathbb{E}\left[\int_0^{T(\hat{v})} v(t, \omega(t))^3 dt + \Big(\frac{L_d}{D - (T(\hat{v}) + q(T(\hat{v})) + \tau_a)} \Big)^2 L_d \right].$$

The first part of the expectation is a constant and we can reformulate it as follows

$$\hat{v}^2 L_u + \mathbb{E}\left[\Big(\frac{L_d}{D - (T(\hat{v}) + q(T(\hat{v})) + \tau_a)} \Big)^2 L_d \right].$$

As $\hat{v} = L_u/T(\hat{v})$ we obtain

$$\frac{L_u^3}{T(\hat{v})^2} + \mathbb{E}\left[\Big(\frac{L_d}{D - (r_a + q(r_a) + \tau_a)} \Big)^2 L_d \right].$$

Observe that $q(T(\hat{v})) = 0$ if $T(\hat{v}) \geq r + c_i$ and $q(T(\hat{v})) = r + c_i - T(\hat{v})$ if $T(\hat{v}) < r + c_i$. Hence, Equation (5.12) follows:

$$\frac{L_u^3}{T(\hat{v})^2} + \sum_{i=0}^{n} p_i \left[\Big(\frac{L_d}{D - (\max\{T(\hat{v}), r + c_i\} + \tau_a)} \Big)^2 L_d \right].$$

□

The intuition behind the total expected fuel consumption is the following: (1) on the upstream section a fixed arrival policy chooses a fixed

velocity up to a deterministic arrival time and, therefore, the fuel consumption is constant given the velocity and (2) the travel time available on the downstream depends on the arrival time and whether or not vessel a needs to wait at the lock or can enter without waiting.

An optimal *fixed arrival* policy is a solution to the following mathematical program

$$\min_{\hat{v}} \frac{L_u^3}{T(\hat{v})^2} + \sum_{i=0}^{n} p_i \left[\left(\frac{L_d}{D - (\max\{T(\hat{v}), r + c_i\} + \tau_a)} \right)^2 L_d \right] \tag{5.13}$$

$s.t.$

$$L_u / v_{max} \leq T(\hat{v}) < \min\left\{ D - \tau_a, L_u / v_{min} \right\}. \tag{5.14}$$

Finding a closed-form solution for the optimal fixed arrival time and, therefore, the optimal fixed arrival policy is desirable. However, finding such a solution to the above mathematical program requires finding a roof of a polynomial of degree at least 6. Generally, solving a polynomial with degree higher than five is analytically intractable. Therefore, it is unlikely that a local minimum and, therefore, the global minimum can be found in a closed form. Hence, any standard solver can be used to find the optimal fixed arrival policy numerically.

In the following, we illustrate the derivation of an optimal fixed arrival policy for the setting introduced in Example 5.2.1 and compare the fuel consumption of this optimal fixed arrival policy to the fuel consumption of the simple approach introduced in Section 5.2.

Example 5.4.2. *Recall that the instance is characterized by the following values:*

- $L_u = L_d = 1$
- $D = 6$
- $r = 0$
- $Pr[\tau_b = 1] = 1$

- $Pr[\tau_b = 1] = Pr[\tau_b = 2] = Pr[\tau_b = 3] = \frac{1}{3}$

The objective function in the fixed arrival setting for this instance is given by:

$$f(t) = \begin{cases} \frac{1}{t^2} + \frac{1}{3}\frac{1}{(6-(1+1))^2} + \frac{1}{3}\frac{1}{(6-(2+1))^2} + \frac{1}{3}\frac{1}{(6-(3+1))^2} & 0 < t < 1; \\ \frac{1}{t^2} + \frac{1}{3}\frac{1}{(6-(t+1))^2} + \frac{1}{3}\frac{1}{(6-(2+1))^2} + \frac{1}{3}\frac{1}{(6-(3+1))^2} & 1 \leq t < 2; \\ \frac{1}{t^2} + \frac{2}{3}\frac{1}{(6-(t+1))^2} + \frac{1}{3}\frac{1}{(6-(3+1))^2} & 2 \leq t < 3; \\ \frac{1}{t^2} + \frac{1}{(6-(t+1))^2} & 3 \leq t < 5. \end{cases} \tag{5.15}$$

In Figure 5.5 the objective function for this example is plotted.

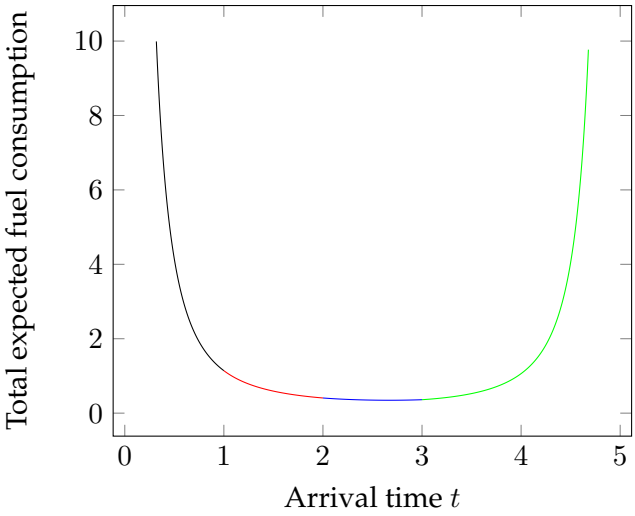


Figure 5.5: Objective function for the illustrative example

Recall that the simple approach in which the skipper arrives at the optimal arrival time assuming that he can enter the lock without waiting yields an arrival time $t^* = 2.5$ with total fuel consumption $f(2.5) = 0.35$.

Taking the conflicting vessel and the possibility of waiting time into account the skipper may choose the optimal fixed arrival time by minimizing Equa-

tion (5.15) *yielding a total fuel consumption of:*

$$f(2.668) = 0.34.$$

Hence, simply being aware of the conflict and possible waiting time of the lock yields fuel savings of 2.86% *in this example.*

5.5 Computational Experiments

In the computational experiments we aim to investigate the fuel efficiency of the optimal dynamic velocity policy and the optimal fixed arrival policy compared to a natural benchmark policy. This natural benchmark policy, referred to a-priori fixed arrival, is given by the optimal arrival time at the lock assuming absence of vessel b. In addition to the economic and ecological aspects we analyse the simplicity of the optimal dynamic policy from a skipper's perspective.

We refer to the optimal dynamic velocity policy as v^*_{dyn}, the optimal fixed arrival velocity as v^*_{fixed} with arrival time $T(v^*_{fixed})$ and the a-priori fixed arrival time as t^*. As discussed earlier this benchmark can be solved in a closed form:

$$t^* = \frac{L_u + L_d}{D - \tau_a} L_u.$$

In order to adequately compare the fuel consumption of these approaches we compute the fuel consumption of t^* in the presence of vessel b, i.e., given t^* we compute the fuel consumption of a fixed arrival policy according to Equation (5.13).

In order to measure the efficiency we consider two criteria: (1) the ratio of fuel consumption of v^*_{dyn} and t^* and (2) the ratio of the fuel consumption of v^*_{dyn} and v^*_{fixed}. To gain valuable insights we approach the following questions regarding the efficiency of the two policies:

E.1 What is the magnitude of fuel savings of the optimal dynamic velocity policy? How do the characteristics of vessel b impact the possible fuel savings?

E.2 What share of these savings can be achieved by the optimal fixed arrival policy? How do the characteristics of vessel b impact the possible fuel savings?

Next we take the perspective of the skipper and look at the simplicity of the optimal dynamic velocity policy. As a measure of simplicity we count how often the skipper may need to change velocity before arriving at the lock. From a skipper's point-of-view the number of velocity changes is a direct indicator on how easy it is to follow a policy. A policy that leads to a few large velocity changes is more attractive than a policy that implies many small velocity changes as more frequent velocity changes are prone to mistakes by the skipper. As an indication of the trade-off between simplicity and efficiency we look at the number of velocity changes and the added value of these changes in terms of the fuel savings of v^*_{dyn} compared to v^*_{fixed}.

With respect to the simplicity of the optimal dynamic velocity policy we aim to answer the following questions:

S.1 To what extent is the optimal dynamic velocity policy implementable by skippers? What is the trade-off between fuel savings and simplicity of the policy for skippers?

S.2 How do the characteristics of vessel b impact the simplicity?

5.5.1 Instance and Scenario parameters

Throughout the experiments we fix the physical situation surrounding the lock, the deadline of vessel a as follows:

- $L_u = L_d = 20\text{km}$
- $D = 150\text{min}$

To create a significant set of scenarios we vary the following parameters:

- τ_a: Lock processing time of our own vessel
- r: Arrival time of vessel b at the lock
- τ_b: Lock processing time of vessel b, i.e., the probability distribution underlying the processing times.

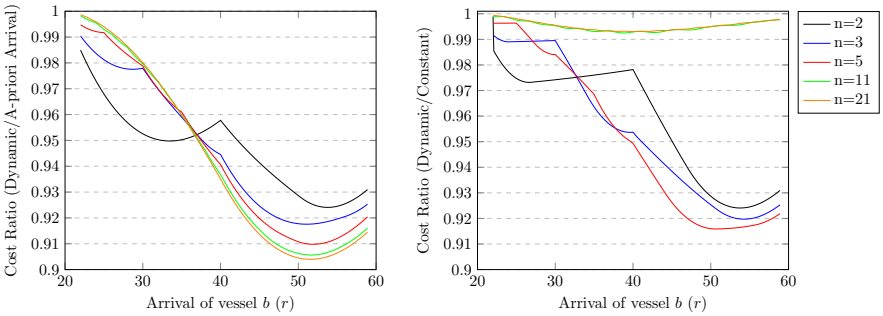
We distinguish between two groups of experiments. We first consider uniform distributions underlying the processing time of vessel b. From a practical point of view, uniform distributions with different time windows between realizations can be used to model different types of vessels and from a theoretical point of view uniform distributions are simple to model and analyse the impact of the size of the time windows between realizations. We assume that the uniform distributions have a minimum value 20 and maximum value 40. We parameterize the distribution by the number of realizations n, i.e. number of possible processing times. For example, for $n = 3$, there are three processing times possible, namely $c_1 = 20$, $c_2 = 30$, $c_3 = 40$. And let the lock processing time of vessel a be $\tau_a = 30$.

The second group of experiments considers discretized and truncated geometric distributions underlying the lock processing time of vessel b. Investigations of AIS data have shown that the process time follows a geometric distribution, see Figure 5.1. Furthermore, geometric distributions are among the commonly chosen distribution for stochastic processing times [43]. In this setting the processing time τ_b lies in the interval $[20, 39]$ and follows a truncated geometric distribution with parameter $p \in \{0.1, 0.2, ..., 0.9\}$. The processing time of vessel a is constant with $\tau_a = 29.5$.

In both settings we choose $r \in [22, 59]$ in steps of size 0.1. Table 5.1 summarizes the settings chosen for the computational experiments. The benchmark arrival times for the two groups are given by

τ_b	*Parameters*	τ_a	r
Uniform	$n \in \{2, 3, 5, 11, 21\}$	30	$[22, 59]$
Geometric	$p \in \{0.1, 0.2, ..., 0.9\}$	29.5	$[22, 59]$

Table 5.1: Scenarios



(a) Fuel consumption compared to a-priori fixed arrival

(b) Fuel consumption compared to optimal fixed arrival

Figure 5.6: Fuel consumption ratios for uniformly distributed processing times for τ_b

$$t^*_{uni} = \frac{L_u + L_d}{D - \tau_a} L_u = 60$$

and

$$t^*_{geo} = \frac{L_u + L_d}{D - \tau_a} L_u = 60.25.$$

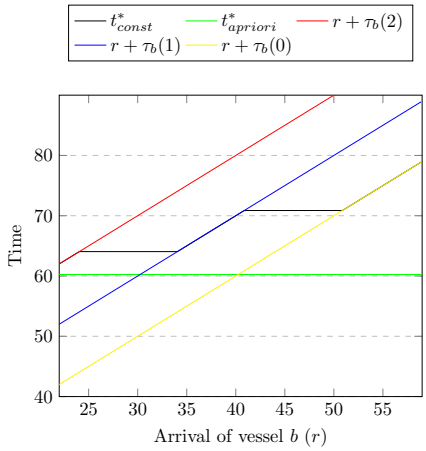
5.5.2 Efficiency

We first consider the scenarios with τ_b following a uniform distribution. Figure 5.6 shows the fuel consumption ratios plotted as functions of r for the different values of n. With respect to Question **E.1**, one can observe that v^*_{dyn} yields significant fuel savings up to approximately 9% as r becomes larger for all values of n. In fact, the fuel savings achievable by v^*_{dyn} follow a piece-wise convex relation with respect to the arrival of vessel a. The different parts are due to the fact that at

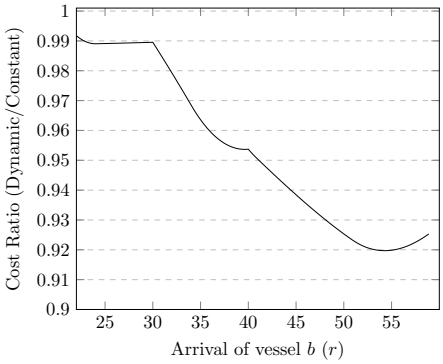
these points one more realization moves past t^*_{uni} (see Figure 5.7 for an example). We give a short intuitive explanation for this. Consider the setting with $n = 2$ and the interval $[21, 40]$ for r. The fuel consumption ratio follows a parabolic relation with respect to r. As r increases to 40 the first possible time at which the lock has finished processing vessel b moves closer toward t^*_{uni} while the second realization moves further away. As t^* is the optimal a-priori fixed arrival we want to arrive close to t^*_{uni} if the lock finishes before and as close to t^*_{uni} as possible if the lock finishes afterwards (due to convexity). However, the closer t^*_{uni} is to the second realization the lower the impact on the fuel consumption of the velocity change at time $r + c_1$ (the first realization).

With respect to Question **E.2**, Figure 5.6b shows that for small values of n the fuel savings are mainly due to the added value of changing velocities, i.e. v^*_{dyn} should be preferred. However, for $n = 11$ or $n = 21$ the fuel consumption of v^*_{dyn} differs from the fuel consumption of v^*_{fixed} by less than 1%. This could be explained intuitively as follows. As the number of scenarios increases size of the time interval between scenarios decreases. This implies that the savings that can be achieved with the dynamic velocity policy decrease as arriving at the lock between two realizations does not lead to long waiting times. Additionally, the fixed arrival policy may account for more uncertainty for higher values of n. Hence, for these settings the simple fixed arrival policy yields a large amount of the savings possible when adding the uncertainty assumption. The relation between r and the fuel consumption ratio is piece-wise convex as well for the same reason as before. Next, we take a look at the efficiency in the setting with τ_b following geometric distributions (Figure 5.8).

With respect to Question **E.1**, one can observe that the fuel savings of v^* again follow convex relation with respect to r and savings up to approximately 14% are possible. An interesting observation is that up to some value of r the savings are higher for lower values of p (more uncertainty in the lock processing time of vessel b) and afterwards the savings are lower for higher values of p (less uncertainty). This may be related to the probability of entering the lock before the optimal a-

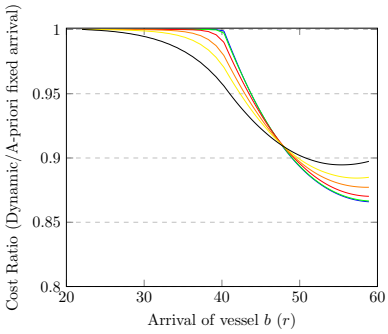


(a) A-priori arrival time compared to realizations of lock finishing time

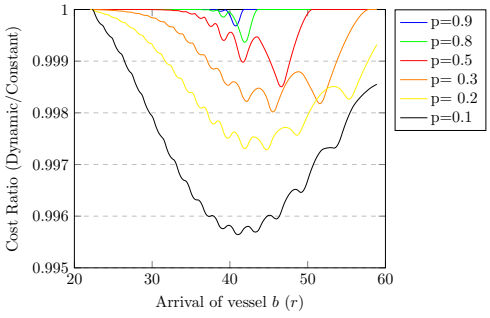


(b) Consumption ratio: role of a-priori fixed arrival

Figure 5.7: Detailed look at relation between Consumption ratio and r with uniformly distributed processing times and $n = 3$



(a) Consumption ratio compared to a-priori fixed arrival



(b) Consumption ratio compared to optimal fixed arrival

Figure 5.8: Consumption ratios for geometrically distributed processing times for τ_b

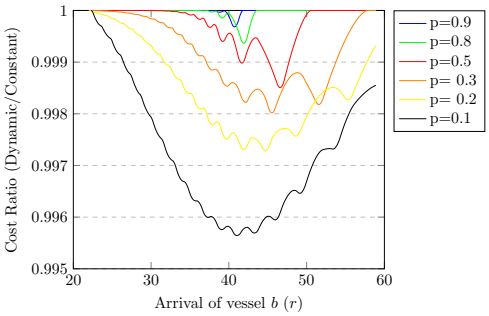
priori arrival time. For low values of r this can still be achieved with a probability depending on p. Hence, the lower p the lower the probability of this event and the dynamic policy can, naturally, adjust velocity when necessary whereas the fixed arrival policy may incur long waiting times at the lock. For higher values of r the event becomes less likely and, therefore, an early as possible arrival is desirable.

Similar to the results of Figure 5.6b for high values of n, Figure 5.8b gives a clear answer to Question **E.2**. The fuel savings possible for these scenarios are mainly due to the optimal fixed arrival policy v^*_{dyn}. This highlights the attractiveness of this simple type of policy in real-life settings as it is both efficient and simple. However, small additional savings can still be achieved with v^*_{dyn}. The piece-wise convex relation seen in Figure 5.8b can be explained in similar fashion as the piece-wise convex relation in the setting of uniform distributions.

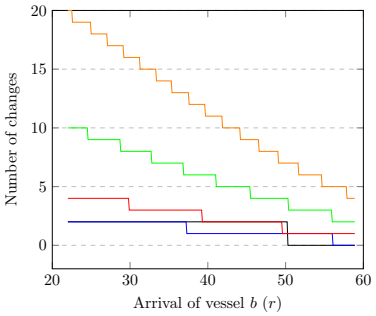
5.5.3 Simplicity

In Figures 5.9 and 5.10 the trade-off between simplicity of v^*_{dyn} and additional fuel savings compared to the optimal fixed arrival is depicted for the setting with uniform and geometric distributions, respectively.

Both figures give a clear answer to Question **S.2**. The number of changes decreases as r increases for both uniform and geometric distributions. This can be explained by the increased consumption of crossing the lock at a later time due to the fuel consumption on the downstream segment. Therefore, an early arrival with possible waiting time is more fuel efficient than a later arrival. An early arrival also implies that vessel a has more opportunities to enter the lock as soon as it is finished without having to travel a remaining distance to the lock. Clearly, for uniform distributions the number of changes depends on n (due to Lemma 5.3.2). Furthermore, the size of the steps, i.e. the number of different values of r with the same number of changes, increases as r increases. For geometric distributions, the number of changes goes



(a) Number of velocity changes in optimal dynamic velocity policy



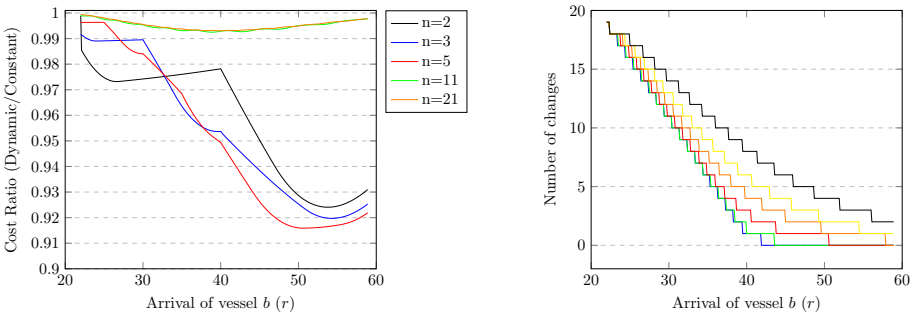
(b) Fuel consumption compared to optimal fixed arrival

Figure 5.9: Simplicity versus Efficiency with uniformly distributed τ_b

towards 0 and reaches 0 at an earlier r for higher values of p. Arriving at the lock early for high values of p implies a low negative impact of waiting time on fuel consumption as high realizations of τ_b are less likely.

With respect to Question **S.2**, one can see that for the setting with uniform distributions the high additional savings for low values of n can be achieved with a relatively small amount of velocity changes. For high values of n as well as for the case of geometric distributions a significant number of velocity changes has to be made to obtain relatively low additional fuel savings over the optimal fixed arrival policy.

To summarize, for probability distributions of τ_b with more frequent realizations the optimal fixed arrival time policy accounts for a large share of the fuel savings possible by implementing uncertainty in the process of finding the most fuel efficient way of crossing the water way for vessel a. This is interesting as these distributions occur more often in practice. If, the time between the realizations of τ_b is large, then a dynamic velocity policy can yield additional fuel savings as velocity changes are made over larger time intervals and, therefore, have larger impact on fuel consumption.



(a) Number of velocity changes in optimal dynamic velocity policy

(b) Fuel consumption compared to optimal fixed arrival

Figure 5.10: Simplicity versus Efficiency with uniformly distributed τ_b

5.6 Conclusion

We introduced a mathematical optimization problem modelling the uncertainty at locks in inland waterways when considering the operation of a single vessel. In real life, this uncertainty is present and caused by many factors but has so far not been considered in theory and the problem introduced in this chapter can be used as a first step towards more understanding of how to model uncertainty in inland waterway transportation and how to solve corresponding mathematical optimization problems.

For the problem considered in this chapter we show how to find a close-to-optimal velocity policy and introduce a simple heuristic based on the notion of fixed arrival. Our computational experiment underlines the importance of taking uncertainty into account as relevant fuel savings can be achieved, but also shows a clear trade-off between efficiency and simplicity of the considered policies. In future research the computational experiment could be extended to different problem settings such as different types of processing time distributions.

The methods can be implemented and supplied to skippers, which decreases their operational cost and decrease ecological footprint. Fur-

thermore, the insight can be used by lock operators and policy makers to value a digitalization of the inland waterway transportation.

Bibliography

[1] European Commission, *The European Green Deal*, https://ec.europa.eu/info/strategy/priorities-2019-2024/european-green-deal_en, Accessed 27 January 2021, 2020.

[2] ——, *Inland Navigation in Europe, Market Observation. Central commission for the navigation of the Rhine, Annual Report 2019*, https://inland-navigation-market.org/wp-content/uploads/2019/11/ccnr_2019_Q2_en-min2.pdf.pdf, 2019.

[3] Eurostat, *Navigable inland waterways, by horizontal dimensions of vessels and pushed convoys*, http://appsso.eurostat.ec.europa.eu/nui/show.do?dataset=iww_if_hordim&lang=en, Accessed 09 September 2020, 2020.

[4] European Commission, *Digital Inland Waterway Area: Towards a Digital Inland Waterway Area and Digital Multimodal Nodes*, https://ec.europa.eu/transport/sites/transport/files/studies/2017-10-dina.pdf, Accessed 09 September 2020, 2017.

[5] European Statistical Office, *Freight transport statistics - modal split*, Extracted from https://ec.europa.eu/eurostat/statistics-explained/pdfscach 2020.

[6] W. Passchyn, "Scheduling locks on inland waterways.," 2016.

[7] W. Passchyn, S. Coene, D. Briskorn, J. L. Hurink, F. C. R. Spieksma, and G. V. Berghe, "The lockmaster's problem," *European Journal of Operational Research*, vol. 251, no. 2, pp. 432–441, 2016.

[8] W. Passchyn, D. Briskorn, and F. C. Spieksma, "No-wait scheduling for locks," *INFORMS Journal on Computing*, vol. 31, no. 3, pp. 413–428, 2019.

[9] J. Verstichel, P. De Causmaecker, F. C. R. Spieksma, and G. V. Berghe, "Exact and heuristic methods for placing ships in locks," *European Journal of Operational Research*, vol. 235, no. 2, pp. 387–398, 2014.

[10] J. Verstichel, P. De Causmaecker, F. C. R. Spieksma, and G. V. Berghe, "The generalized lock scheduling problem: An exact approach," *Transportation Research Part E: Logistics and Transportation Review*, vol. 65, pp. 16–34, 2014.

[11] N. Pinson and F. C. Spieksma, "Online interval scheduling on two related machines: The power of lookahead," *Journal of Combinatorial Optimization*, vol. 38, no. 1, pp. 224–253, 2019.

[12] W. Passchyn, D. Briskorn, and F. C. R. Spieksma, "Mathematical programming models for lock scheduling with an emission objective," *European Journal of Operational Research*, vol. 248, no. 3, pp. 802–814, 2016.

[13] M. Prandtstetter, U. Ritzinger, P. Schmidt, and M. Ruthmair, "A variable neighborhood search approach for the interdependent lock scheduling problem," in *Evolutionary Computation in Combinatorial Optimization - 15th European Conference, EvoCOP 2015, Copenhagen, Denmark, April 8-10, 2015, Proceedings*, G. Ochoa and F. Chicano, Eds., ser. Lecture Notes in Computer Science, vol. 9026, Springer, 2015, pp. 36–47.

[14] W. Passchyn and F. C. Spieksma, "Scheduling parallel batching machines in a sequence," *Journal of Scheduling*, vol. 22, no. 3, pp. 335–357, 2019.

[15] E. R. Petersen and A. J. Taylor, "An optimal scheduling system for the welland canal," *Transportation science*, vol. 22, no. 3, pp. 173–185, 1988.

[16] L. D. Smith, D. C. Sweeney, and J. F. Campbell, "Simulation of alternative approaches to relieving congestion at locks in a river transportion system," *Journal of the Operational Research Society*, vol. 60, no. 4, pp. 519–533, 2009.

[17] L. D. Smith, R. M. Nauss, D. C. Mattfeld, J. Li, J. F. Ehmke, and M. Reindl, "Scheduling operations at system choke points with sequence-dependent delays and processing times," *Transportation Research Part E: Logistics and Transportation Review*, vol. 47, no. 5, pp. 669–680, 2011.

[18] R. M. Nauss, "Optimal sequencing in the presence of setup times for tow/barge traffic through a river lock," *European Journal of Operational Research*, vol. 187, no. 3, pp. 1268–1281, 2008.

[19] C.-J. Ting and P. Schonfeld, "Control alternatives at a waterway lock," *Journal of Waterway, Port, Coastal, and Ocean Engineering*, vol. 127, no. 2, pp. 89–96, 2001.

[20] E. Lübbecke, M. E. Lübbecke, and R. H. Möhring, "Ship traffic optimization for the Kiel Canal," *Operations Research*, vol. 67, no. 3, pp. 791–812, 2019.

[21] C. Defryn, J. A. P. Golak, A. Grigoriev, and V. Timmermans, "Inland waterway efficiency through skipper collaboration and joint speed optimization," *European Journal of Operational Research*, 2020.

[22] J. Nash, "Non-cooperative games," *Annals of mathematics*, pp. 286–295, 1951.

[23] V. V. Vazirani, *Approximation algorithms*. Springer Science & Business Media, 2013.

[24] N. Nisan, T. Roughgarden, E. Tardos, and V. Vazirani, *Algorithmic game theory, cambridge univ*, 2007.

[25] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. New York, NY, USA: W. H. Freeman & Co., 1979.

[26] J. Golak, A. Grigoriev, F. van Lent, and T. van der Zanden, "Periodic lock scheduling problem," *Working Paper*, 2021.

[27] S. Anily, C. A. Glass, and R. Hassin, "The scheduling of maintenance service," *Discrete Applied Mathematics*, vol. 82, no. 1-3, pp. 27–42, 1998.

[28] A. Bar-Noy, R. Bhatia, J. Naor, and B. Schieber, "Minimizing service and operation costs of periodic scheduling," *Mathematics of Operations Research*, vol. 27, no. 3, pp. 518–544, 2002.

[29] C. A. Glass, "Feasibility of scheduling lot sizes of three products on one machine," *Management Science*, vol. 38, no. 10, pp. 1482–1494, 1992.

[30] E.-S. Kim and C. A. Glass, "Perfect periodic scheduling for three basic cycles," *Journal of Scheduling*, vol. 17, no. 1, pp. 47–65, 2014.

[31] C. A. Glass, "Feasibility of scheduling lot sizes of two frequencies on one machine," *European Journal of Operational Research*, vol. 75, no. 2, pp. 354–364, 1994.

[32] C. Glass, J. Gupta, and C. Potts, "Lot streaming in three-stage production processes," *European Journal of Operational Research*, vol. 75, no. 2, pp. 378–394, 1994.

[33] S. Patil and V. K. Garg, "Adaptive general perfectly periodic scheduling," *Information processing letters*, vol. 98, no. 3, pp. 107–114, 2006.

[34] Z. Brakerski, A. Nisgav, and B. Patt-Shamir, "General perfectly periodic scheduling," *Algorithmica*, vol. 45, no. 2, pp. 183–208, 2006.

[35] N. Brauner, Y. Crama, A. Grigoriev, and J. Van De Klundert, "Multiplicity and complexity issues in contemporary production scheduling," *Statistica Neerlandica*, vol. 61, no. 1, pp. 75–91, 2007.

[36] P. A. Golovach, P. Heggernes, D. Kratsch, and Y. Villanger, "An incremental polynomial time algorithm to enumerate all minimal edge dominating sets," *Algorithmica*, vol. 72, no. 3, pp. 836–859, 2015.

[37] J. Golak, C. Defryn, and A. Grigoriev, "Optimizing fuel consumption on inland waterway networks," *Working Paper*, 2021.

[38] M. Buchem, J. A. P. Golak, and A. Grigoriev, "Vessel velocity decisions in inland waterway transportation under uncertainty," *European Journal of Operational Research*, 2021.

[39] B. D. Brouer, J. Dirksen, D. Pisinger, C. E. Plum, and B. Vaaben, "The Vessel Schedule Recovery Problem (VSRP)–a MIP model for handling disruptions in liner shipping," *European Journal of Operational Research*, vol. 224, no. 2, pp. 362–374, 2013.

[40] S. Wang and Q. Meng, "Liner ship route schedule design with sea contingency time and port time uncertainty," *Transportation Research Part B: Methodological*, vol. 46, no. 5, pp. 615–633, 2012.

[41] ——, "Robust schedule design for liner shipping services," *Transportation Research Part E: Logistics and Transportation Review*, vol. 48, no. 6, pp. 1093–1106, 2012.

[42] N. Aydin, H. Lee, and S. A. Mansouri, "Speed optimization and bunkering in liner shipping in the presence of uncertain service times and time windows at ports," *European Journal of Operational Research*, vol. 259, no. 1, pp. 143–154, 2017.

[43] D. Trietsch, L. Mazmanyan, L. Gevorgyan, and K. R. Baker, "Modeling activity times by the Parkinson distribution with a lognormal core: Theory and validation," *European Journal of Operational Research*, vol. 216, no. 2, pp. 386–396, 2012.

[44] European Commission, *Roadmap to a Single European Transport area – Towards a competitive and resource efficient transport system*, `https://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:52011DC0144&from=EN`, Accessed 09 September 2020, 2011.

Summary

In this thesis, we investigate problems of managing the arrival of vessels on locks in order to increase efficiency on inland-waterways. We explore problems in which we give advise on velocity decisions to skippers, such that the overall fuel consumption is minimized. Lastly, one problem considers a schedule of lock operations to minimize the waiting time of streams of periodic arrival of vessels.

In the first paper provides an introduction, motivation and a literature review. In the second chapter, we consider a river with a single lock and put the lock management into a game theoretic setting. That is, we consider skippers as agents that try to attain the best outcome for themselves while disregarding the outcome of other skippers. We show that this selfish behavior may lead to situations that are not stable, meaning that skippers may keep deviating their velocity to achieve a better outcome. Next, we introduce supervision methods of the lock operator that guarantee stability, but the aggregated fuel consumption can be high. Finally, we propose a payment mechanism, guaranteeing a stable situation while keeping the aggregated fuel consumption at a minimum.

In the next chapter, we assume that vessels arrive in periodic streams at the lock. This means that, in each stream, the time between the arrivals of two vessels is the same. The aim is to construct a schedule of lock operations, such that the average waiting time over a long period of time is small. For a small instance of only two streams, we provide a method that computes an optimal schedule instantly. For the general problem, we provide a method to compute the optimal schedule that is relatively slow. However, we propose an alternative method that gives a part of the schedule in a short time.

In the third chapter, we consider an entire fleet of vessels in a setup with multiple locks on river crossings. We propose a heuristic, which

is a method that quickly constructs a schedule without any guarantee of the gap to the optimal solution. With numerical experiments, we show that an optimal solution solver is not applicable on instance that are comparable to a real-world problem. The numerical experiments provide relevant managerial insights that can support decision making in lock scheduling.

In the final chapter, we consider a waterway with a single lock. We assume that there is uncertainty in the time that vessels requires to get processed through the lock. We show how to find a close-to-optimal strategy for a skipper. Furthermore, we propose a method to compute a simple strategy instantly in short computation time. Lastly, we provide numerical experiments that emphasize the advantages and disadvantages between the two methods.

Impact paragraph

The transportation system as is not yet sustainable [44]. Oil is becoming a scarcer resource as it is sourced from uncertain supplies. Due to this growing uncertainty and price increase in oil, the economic security could be severely impacted due to its dependence on oil. Furthermore, the increasing demand for transport caused by globalization puts a high pressure on the existing transportation network. The congestion on the road network continues to increase and thus poses new challenges to the transportation sector and our society.

The EU proposed the Green Deal in order to define new environmental goals and strategies on achieving these [1]. Among others, they claim that their ambitious climate goals require a shift to more sustainable transport modes such as rail and inland waterways. In comparison to road transportation, the use of inland waterways is more environmentally friendly because of lower greenhouse gas emissions per volume or weight unit. It is also relatively cheap due to economies of scale and increased bundling opportunities. However, the inland waterway network is less dense than the road network.

Another strategy is the focus on multi-modal transport, i.e. combining rail and waterborne transport, including short-sea shipping in a supply chain. This may increase the coordination and administrative cost, but has a high potential to increase the use of more sustainable transport modes and thus to decrease overall emission.

Furthermore, the Green Deal calls for increased efficiency across the transportation sector. Digital technologies, smart applications and other new innovations will play an important role in increasing the capacity of current infrastructures and in reducing the cost of sustainable transportation.

In this research, we focus on problems that practitioners in inland-waterway transportation are currently facing. The real-world problems are reduced to mathematical formulation and solution concepts for the mathematical problems are proposed. For each of the solution concepts, we provide theoretical results or numerical experiments to give further insights.

Practitioners and regulators may use the designed algorithms and extend them to the specific real-life situation that they are facing. Thus, the results this thesis can be used to evaluate and design new business models that create profit and efficiency in inland-waterway transportation. Furthermore, the ideas may help regulators to evaluate the effectiveness of investments to digitalize and controlling measures.

The second chapter concerns a payment mechanism for an inland waterway system such that the total fuel consumption is minimized, without the opportunity for vessels to profit by deviating from the proposed solution. The applicability of these concepts is constrained by the computational complexity of the methodology used. Therefore, practitioners need to extend these ideas and provide computationally solvable solutions for real-life scenarios.

In the third chapter, we introduced and proposed different algorithms for the Periodic Lock Scheduling Problem. The enforcement of regular schedules could potentially be a management strategy for the locks. Vessels would then adapt their velocities according to this schedule and congestion would reduce drastically. The algorithms could be extended to scheduling tools for lock operators.

In the third chapter, we consider an entire fleet of vessels on a setup with multiple locks on river crossings. The algorithms are designed to provide a fast solution, which enhances the practicability. Additionally, they were designed for a general problem setting, and can be extended and adjusted to specific situations. Therefore, the proposed algorithms can be used by policy makers and practitioners to implement a central scheduling system on inland-waterways.

In the final chapter, strategies and algorithms for passing through a lock under uncertainty are presented. If the required data is accessible to a skipper, the algorithms can potentially be used in scheduling tool that helps skippers individual to save fuel while passing a lock. Therefore, the need for centralisation and cooperation is low, which then enhances attractiveness and acceptances by skippers.

There are many ways research on inland-waterway transportation can be extended even further. As claimed in the Green Deal, the success of the climate goals depends on multi-modal transportation. Much research on inland-waterway, harbor management and truck transportation has been conducted. Further research could focus on problems that occur if multiple modes of transportation are combined. Furthermore, there are many open question stated in this thesis that would lead to further refinement and extensions. All this together increases attractiveness, awareness and acceptance of practitioners.

About the author

Julian Arthur Pawel Golak was born on January 31, 1993 in Ratingen, Germany. He obtained his BSc degree in Econometrics and Operations Research in August 2015 and a MSc degree in Economic and Financial Research in August 2017, both at Maastricht University. After his studies, he joined the department of Quantitative Economics at Maastricht University as a PhD candidate under the supervision of Prof. Dr. Alexander Grigoriev and Dr. Christof Defryn. Subsequently, he joined the department of Data Analytics and Digitalisation from September 2020 to July 2021. The findings of his research are partly published and have been presented at several international conferences.

In August 2021, Julian joined the Institute of Algorithms and Complexity in the Hamburg University of Technology as a postdoctoral researcher with Prof. Dr. Matthias Mnich.