Submitted to *Operations Research*
manuscript

Ship Traffic Optimization for the Kiel Canal

Elisabeth Lübbecke
Workforce Management, INFORM Software GmbH, Pascalstr. 23, 52076 Aachen, Germany,
elisabeth.luebbecke@inform-software.com

Marco E. Lübbecke
Lehrstuhl für Operations Research, RWTH Aachen University, Kackertstr. 7, D-52072 Aachen, Germany,
marco.luebbecke@rwth-aachen.de

Rolf H. Möhring
Institut für Mathematik, Technische Universität Berlin, Str. d. 17. Juni 136, D-10623 Berlin, Germany,
rolf.moehring@tu-berlin.de
Beijing Institute for Scientific and Engineering Computing, Beijing University of Technology, No. 100 Ping Le Yuan,
Chaoyang District, Beijing 100124, China, rolf.moehring@bjut.edu.cn

We introduce a hard practical optimization problem, the ship traffic control problem (STCP). We study it at the Kiel Canal which is the most frequented artificial waterway in the world. The canal is operated bi-directionally, but large ships can pass each other only in sidings. Decisions must be made about who is waiting for whom, where, and for how long, subject to a multitude of operational constraints. The objective is to minimize the total waiting times of all ships. This problem generalizes train scheduling on a single-track network. We integrate algorithmic ideas from collision-free routing of automated guided vehicles. This leads to a unified view of scheduling and dynamic routing that may serve as a prototype for scheduling bi-directional traffic with passing conflicts. We implement a traffic control tool that uses our combinatorial algorithms and perform a computational study on traffic data from the Kiel Canal. Our tool produces schedules that significantly improve over manual planning and were approved by expert planners.

As traffic volume and vessel sizes are projected to significantly increase, the canal is planned to be enlarged in a huge project. Our tool was used to select from a variety of enlargement options. The enormous level of detail in our model ensured that the chosen construction scenario actually constitutes a remedy to the impending in-operability of the canal.

Key words: dynamic routing, conflict-free routing, job-shop scheduling, local search, rolling horizon

1. Introduction

With more passages than the Panama and Suez Canals together, the Kiel Canal is the world's most frequented artificial waterway (Brockmann et al., 2008). Located in Northern Germany, it links the North and Baltic Seas. The route through the canal is 250 nautical miles (460km) shorter than the

way around Skaw, saving fuel and reducing CO_2 emissions. The canal is also the safer route, and became the basis for the trade between the countries of the Baltic area with the rest of the world.

Figure 1 The Kiel Canal connects the North and Baltic Seas, cf. `www.wsa-kiel.wsv.de/Nord-Ostsee-Kanal/`.

The canal is operated bi-directionally, but large ships can pass each other only in *sidings*. Decisions must be made about who is waiting for whom, where, and for how long, subject to a multitude of operational constraints. This *traffic control* is done by a higher authority, the Waterways and Shipping Board with a team of nautically experienced expert navigators. The objective is to minimize the total waiting and thus transit times of all ships. In this paper we develop the mathematical and algorithmic foundation for ship traffic control in a canal and build an optimization tool for this operational planning task that respects all the details at the Kiel Canal. Our computational study uses real data from the Kiel Canal, too, and shows that our schedules improve upon manual plans.

Since the end of the 1990s, the Kiel Canal has seen a tremendous growth of traffic demand, and worse, a growth of the total gross tonnage per ship (Brockmann et al., 2008). This development continues, and the Kiel Canal may become inoperable when no countermeasures are taken: a larger share of large vessels complicates scheduling and challenges the navigators. Our tool supported the decisions about enlarging the canal in a huge project, see the discussion section at the end.

Algorithmic Overview. The ship traffic control problem (STCP) has a combinatorial and a geometric flavor. The geometry (Section 2.2) is able to capture every detail about feasibility of itineraries in space and time. The combinatorics reflects scheduling decisions that resolve precedence conflicts. To understand these, we first focus on a relaxation which ignores capacities in sidings (Section 3.1). Ship itineraries (geometry) imply scheduling decisions (combinatorics), but the converse is not true in general. However, for the relaxation, we show that itineraries can be constructed from a feasible schedule in polynomial time by a dynamic routing algorithm that treats a single ship at a time (Theorem 1). This successive quickest path style algorithm extends to the general situation to insert a single route into an existing plan, respecting all constraints imposed by the geometry, when this

is possible (Section 4). Yet, we show that any such sequential routing procedure may yield a total travel time arbitrarily far from optimum (Lemma 2). The reason lies in the inherent interdependence of routing and scheduling. We thus devise a local search on the scheduling decisions (Section 5.2) which uses a routing algorithm to incorporate the geometric part. We reflect the problem's online character (only a limited amount of future routing requests is known) by embedding the algorithm into a rolling horizon planning (Section 5.3). We demonstrate the effectiveness and practicability of our heuristic in a computational study on real data in Section 6.

Our Contributions. Two methodological aspects are of broader interest.

1. We merge two algorithmic areas that deal with collision-avoidance: Train scheduling on a single-track railway network, that has a common interpretation as a *job-shop scheduling* problem; and collision-free routing of automated guided vehicles that spawned the development of elaborate successive *quickest path* algorithms. Integrating scheduling and dynamic routing aspects, we extend the applicability of such quickest path algorithms by "teaching" them to deal with combinatorial decisions *en route*. Use cases are e.g., in vehicle routing with synchronization constraints. This enables us to produce collision-free routes for a whole fleet of vehicles.
2. *Time-expanded* networks are a standard model for routing over time. A fine time discretization lets these networks quickly grow beyond any manageable size. Dynamic routing algorithms avoid this at the expense of a more elaborate algorithm. Some applications additionally call for a *space discretization*, like in traffic assignment, or when the exact location of a node is unknown. We non-trivially extend dynamic routing to also avoid space discretization in such situations. We develop a Dijkstra-like shortest path algorithm that *implicitly* handles both discretizations with an *arbitrary* precision.

From the point of view of applications, our work also extends well beyond the Kiel Canal.

3. Collision-free routing is a research topic of wide practical interest and we give many examples in the literature review where our algorithms could be used. In order to avoid collisions it is customary to block an entire arc in the network for other vehicles for the whole time interval it is in use, even if this is not required in practice. We only forbid *entry times* into arcs which allows having several vehicles on the same arc simultaneously. This resolves an algorithmic problem which hitherto wasted non-negligible optimization potential.
4. Specifically, for the Kiel Canal, we solve a very complex practical problem to the fullest satisfaction of the expert planners. Routing over time, scheduling, and packing aspects are to be considered simultaneously, and in an online manner. We developed a traffic scheduling tool that works with current and future traffic scenarios, and that respects all operational constraints at an enormous level of detail. The practical relevance was verified by experts when they used our tool to assess the operational impact of various canal enlargement strategies.

2. The Ship Traffic Control Problem (STCP)

2.1. The Planning Context

Any two ships, no matter what size, can pass each other in sidings. Outside sidings, on *transit segments*, legal regulations and nautical parameters precisely define the circumstances under which ships are allowed to meet. Ships are categorized into *traffic groups* 1–6, mainly depending on their dimensions and charge; transit segments are classified into *passage numbers* 6, 7, or 8, reflecting physical dimensions. In simplified terms, two ships may pass each other on a transit segment only if the sum of their traffic group numbers does not exceed the respective passage number, see Figure 2. Also, overtaking a moving ship is considered a dangerous maneuver and thus forbidden.

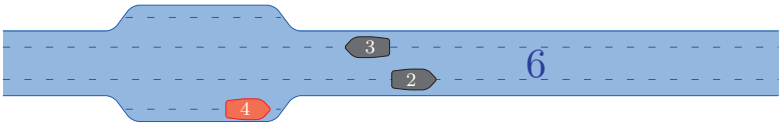


Figure 2 A transit segment of passage number 6. The ship of traffic group 4 has to wait in a siding until the ship of traffic group 3 has released the segment (reason: $3+4>6$). The ship of traffic group 2 can pass (reason: $3+2\leq 6$).

To protect the river bed, velocities of most of the largest ships are limited to 12km/h, that of all others to 15km/h. We assume constant full speed for each ship, as planners do. Because of non-uniform velocities we also need to obey location dependent safety distances.

In each of the 12 sidings along the Kiel Canal, in each direction, there is a *passage track* and a *waiting track*, i.e., four tracks altogether, see Figure 2. A ship travels on the passage track to its waiting position, changes over to the waiting track where it stays for the full duration of waiting, then moves back to the passage track. In addition to the ship's length safety distances to other ships must be kept, both, when waiting and moving. Overtaking a waiting ship is possible. The capacity of a siding is limited, and the choice of waiting positions influences how well this capacity is used. As a rule of thumb, for each ship, the expert planners try not to exceed a waiting time of three hours during the whole journey and one and a half hours individually in each siding. For ships of traffic group 6 this reduces to two hours in total and one hour per siding.

A journey through the canal typically takes seven to nine hours. It may be preempted on many occasions, into side arms or berths. Hence, ships can (and do) appear or disappear for planning literally on arbitrary positions within the canal.

Current traffic control is aided by a *distance-time diagram* (also: time-space or string-line diagram, known from traffic timetabling, see Ceder, 2007). Figure 3 illustrates an example, a screen-shot of the diagram actually in use is given in Figure 4.

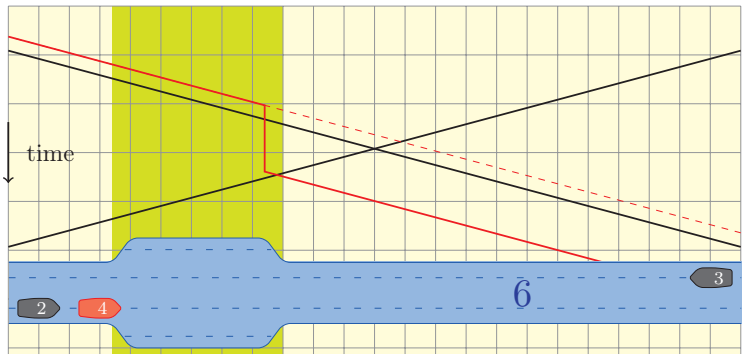


Figure 3 Simplified distance-time diagram for the example of Figure 2. The x-coordinate corresponds to a position within the canal, the y-coordinate to a point in time. The location of the siding is shaded. Each itinerary of a ship is represented by a line plot. Thus, a vertical line within a siding illustrates waiting.

We finally stress the *online* character of ship traffic control. There is a limited look-ahead, as ships register for a journey through the canal only about two hours before arrival. Because of a scheduled locking process at the canal's boundaries, arrival times at the first siding are known and preliminary itineraries can be planned. However, these need to be updated upon arrival of new ships.

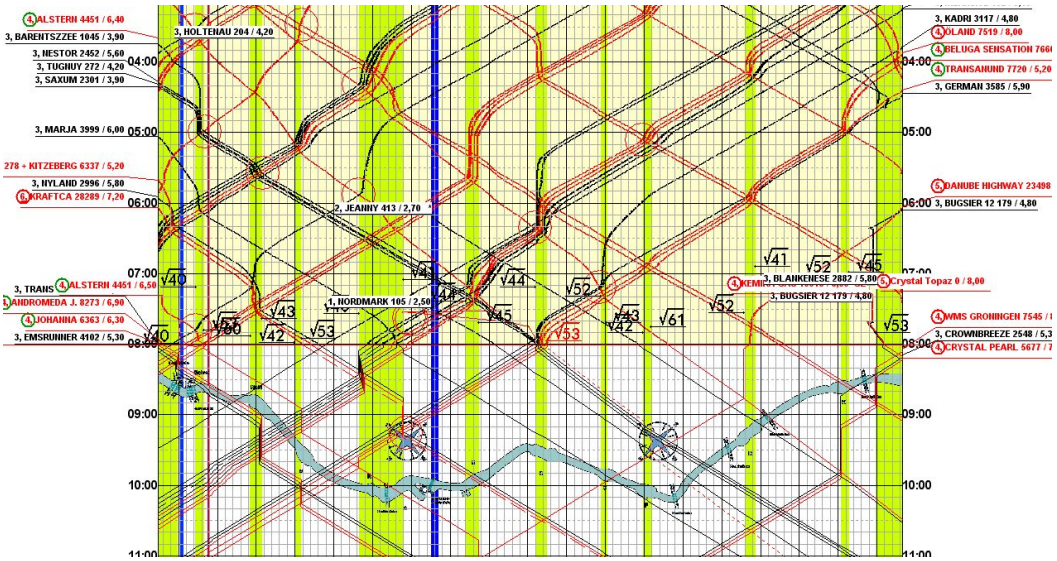


Figure 4 Screen shot of the distance-time diagram which is used for traffic control. The lower part below the current time of 08:00 allows for an interactive traffic control in the near future, like signaling ships to wait in sidings etc. Straight line itineraries reflect future plans.

2.2. A Precise Geometric Model

A *canal* is represented as an interval $C \subset \mathbb{R}$, partitioned into a set $\mathcal{E}$ of m intervals, called *segments*, i.e., $\dot{\cup}_{E\in\mathcal{E}} E = C$, see Figure 5. Elements of a subset $\mathcal{T} \subset \mathcal{E}$ are called *sidings*; we refer to all other

segments as *transit segments*. Sidings and transit segments do not necessarily alternate; there may be consecutive transit segments with different properties. A set of *ships* $S = \{1, \dots, n\}$ induces a set of requests $R = \{(s_i, t_i, v_i, r_i, h_i) \mid i \in S\}$ with *start* and *target* $s_i \neq t_i \in C$ at arbitrary positions in the canal, a *velocity* v_i, a *release time* r_i, and a *parking distance* h_i (see below). The travel direction of a ship (its *heading*), or simply the ship $i \in S$ itself is called *upstream* if $t_i > s_i$ and *downstream* otherwise. Two ships heading in the same direction are called *aligned*, otherwise *opposed*. The velocity of a ship $i \in S$ and the length of a segment $E \in \mathcal{E}$ define the transit time θ_{iE} of ship i along segment E. For each pair of ships $(i, j) \in S \times S$ and each segment $E \in \mathcal{E}$ we are given a *vertical distance* $v_{ijE} \geq 0$. It ensures a sufficient headway between two (aligned or opposed) ships, translated to a duration via the velocities. Since the safety distance between two aligned ships can depend on the size of the ship following behind, $v_{ijE} \neq v_{jiE}$ is possible. The *horizontal parking distance* $h_i \geq 0$ of each ship $i \in S$ defines the required space when the ship is waiting in a siding.

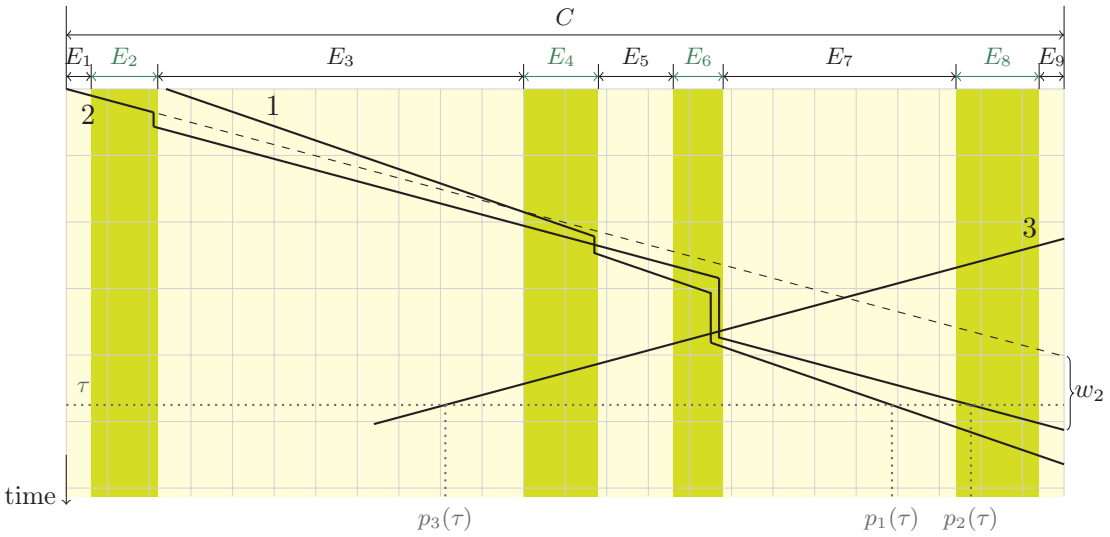


Figure 5 An instance together with a feasible solution. The partition of the canal C into segments $E_1, \dots, E_9$ is shown at the top. The shaded segments correspond to the subset $\mathcal{T} = \{E_2, E_4, E_6, E_8\}$ of sidings. The three solid polygonal chains ("polylines") represent three ships navigating the canal. The start point of each polyline is given by the corresponding ship's start position and release time; the velocity defines the slope. By a dashed line we signify a ship's theoretical onward journey if there were no constraints which enforced waiting. For ship 2 it shows how the waiting time w_2 is calculated. The dotted lines illustrate how positions of ships are obtained from a given point in time by the functions $p_i(.)$.

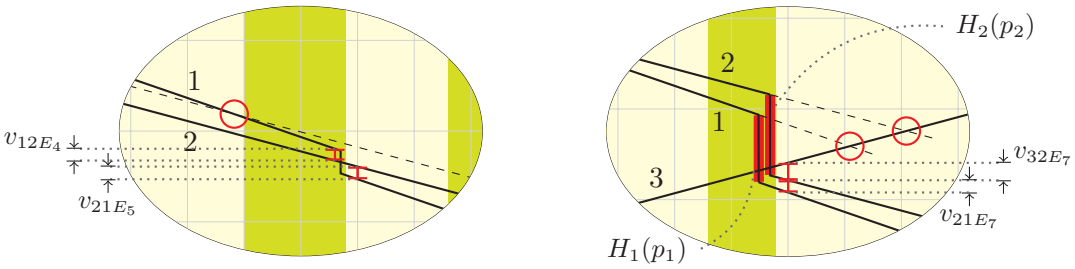
When speaking of a ship's position we refer to its center. Tracing this position over time gives what we call the ship's *dynamic route*. As each ship always moves at constant speed or waits, its dynamic route is fully defined by the locations/times where/when the speed changes. Formally, the dynamic route of ship i is a simple polygonal chain in $C \times \mathbb{R}$, or *polyline*, $P_i = ((p_{i,1}, \tau_{i,1}), \dots, (p_{i,k_i}, \tau_{i,k_i}))$. Each

breakpoint $(p_{i,\kappa}, \tau_{i,\kappa})$, $\kappa = 1, \ldots, k_i$, states that at time $\tau_{i,\kappa}$ ship i is at position $p_{i,\kappa}$. The polyline is defined from the ship's start position and release time to its target position, i.e., $p_{i,1} = s_i$, $\tau_{i,1} = r_i$, and $p_{i,k_i} = t_i$. The straight connection of consecutive breakpoints represents either the movement of ship i from $p_{i,\kappa-1}$ to $p_{i,\kappa}$ with its velocity corresponding to its heading, i.e., $\frac{|p_{i,\kappa} - p_{i,\kappa-1}|}{\tau_{i,\kappa} - \tau_{i,\kappa-1}} = v_i$ with $(p_{i,\kappa} - p_{i,\kappa-1})(t_i - s_i) > 0$, or waiting in a siding, i.e., $p_{i,\kappa} = p_{i,\kappa-1} \in \cup_{T \in \mathcal{T}} T$ with $\tau_{i,\kappa} > \tau_{i,\kappa-1}$, for each $\kappa = 2, \ldots, k_i$. A ship must not wait more than once per siding, thus, a polyline has at most 2 breakpoints per segment. The *waiting time* of ship $i \in S$ is the difference of i's actual passage time in P_i and i's minimum possible passage time, i.e., $w_i := (\tau_{i,k_i} - \tau_{i,1}) - |t_i - s_i|/v_i$. A *solution* $\mathcal{P} = \{P_i \mid i \in S\}$ is a set of such polylines, one for each ship. A solution is *feasible* (or synonymously *collision-free*) if it satisfies the following two properties, cf. Figure 6.

Passing rules: Polyline segments corresponding to two *moving* ships $i, j \in S$ (i.e., of finite slope) are not allowed to be too close to each other (or even cross) if i and j are aligned (in particular, no overtaking) or if i and j are opposed and their passing is not allowed on E. In both cases this will be ensured by the minimal vertical distance v_{ijE} between the line segments.

Parking distance: The required space of a waiting ship defined by the horizontal parking distance h_i must lie completely within the siding. Furthermore, this space must not be occupied by a second ship which is waiting concurrently.

The *Ship Traffic Control Problem (STCP)* is to find a feasible solution that minimizes the *total waiting time* $\sum_{i \in S} w_i$. Due to its geometric character we refer to a feasible solution also as *geometry*.



(a) Details at siding E_4. (b) Details at siding E_6.

Figure 6 Details of Figure 5 to illustrate collision-freeness. In part (a) it is not allowed for the faster ship 2 to overtake the slower ship 1 outside sidings. The circle indicates the conflict which would result from violating the required positive vertical distance. To avoid this, ship 2 has to wait in siding E_2 so that ship 1 can wait in siding E_4 to let it pass. There, the required vertical distances indicated by small intervals are respected. Likewise, for conflicting opposed ships in part (b), positive vertical distances forbid the crossing of the respective line segments. Also in part (b), the required space for waiting of ship 1 and 2 in the siding is depicted by the rectangles which have to be disjoint.

In order to express feasibility formally, we introduce the map $p_i : [\tau_{i,1}, \tau_{i,k_i}] \to C$. It assigns each point in time $\tau \in [\tau_{i,1}, \tau_{i,k_i}]$ to the corresponding position of ship $i \in S$ within the canal at time τ as

defined by polyline P_i. For each position $p \in C$, the preimage $p_i^{-1}(p) =: \tau_i(p)$ gives a time interval, possibly empty or a single point. The open set $H_i(p) := \{[p-h_i, p+h_i] \times \tau_i(p)\}^\circ$, which is the interior of the rectangle occupied for potential waiting at position p, is non-empty only if ship i actually waits at p. Let $\overline{\tau_i}(p) := \max \tau_i(p)$ denote the latest point in time at which ship i stays at position p, and similarly $\underline{\tau_i}(p) := \min \tau_i(p)$ denotes the earliest such point in time. Abusing notation, we refer to the interval $[\min\{a,b\}, \max\{b,a\}] \subset C$ by $[a,b]$ when both, $a \le b$ or $b \le a$ is possible, depending on the considered heading. A solution is *feasible* or *collision-free* if the following holds:
Passing rules: For all $i \neq j \in S$, $E \in \mathcal{E}$, and $p \in E \cap [s_i, t_i] \cap [s_j, t_j]$ with $\overline{\tau_i}(p) \le \underline{\tau_j}(p)$:

$$\underline{\tau_j}(p) - \overline{\tau_i}(p) \ge v_{ijE} \ .$$

Parking distance: For all aligned ships $i \neq j \in S$, $T \in \mathcal{T}$, and $p_1, p_2 \in T$:

$$H_i(p_1) \subset T \times \mathbb{R} \qquad \text{and} \qquad H_i(p_1) \cap H_j(p_2) = \emptyset \ .$$

Remark 1. It would be simpler to require that the sum of lengths of parking ships does not exceed the length of a siding at any point in time. For later reference, we call this the *knapsack relaxation* (one can assume that all ships wait at the end of a siding). This relaxation is weaker than our above model of parking in sidings. Our stronger model is related to orthogonal *rectangle packing* on a strip (Baker et al., 1980). If all ship dimensions are identical the weaker knapsack condition suffices to ensure a corresponding rectangle packing. In general, it is strongly NP-hard to decide if a feasible solution to the knapsack relaxation implies a corresponding feasible rectangle packing; see Błądek et al. (2015), Duin and Van Der Sluis (2006), Günther et al. (2014) for discussions and proofs.

In what follows, we compare times relative to a reference point. Given a time τ_1 at position p_1 for an upstream (resp. downstream) ship i we define $\tau_2 := \tau_1 + (p_2 - p_1)/v_i$ (resp. $\tau_1 - (p_2 - p_1)/v_i$) to be the corresponding time *translated to position* p_2. This is illustrated by the dashed straight line through (p_1, τ_1) with slope defined by the heading and velocity of i, in e.g., Figures 5 and 6.

2.3. Related Work and Applications

Other *waterways with sidings* for passing ships are the Suez Canal (Griffiths, 1995) in Egypt before 2015, the Göta River in Sweden (Holm and Grundevik, 2015), the Świnoujście-Szczecin fairway in Poland (Gucma, 2016), or the Strait of Istanbul in Turkey (Ulusçu et al., 2009). All have fewer sidings, thus the combinatorial complexity of the ship traffic management does not match that of the Kiel Canal. Some *maritime container terminals* are connected to the open sea via an entrance waterway (like the Yangtze Delta in Shanghai, see Lalla-Ruiz et al., 2016). The port's performance heavily depends on the throughput of such a single transit segment. This is a special case of our

situation. Moreover, one may consider such entrance waterways together with the *berth allocation problem* in the port (Bierwirth and Corry, 2018). We would model a berth as a transit segment (in a more general network) where the loading/unloading time gives the transit time.

Many waterways, in particular in the US, are wide enough that two ships may pass at any place, but level differences require using locks. *Lock scheduling* (e.g., Passchyn et al. 2016, Petersen and Taylor 1988, Verstichel and Vanden Berghe 2009) has a combinatorial component: which groups of ships are together in the same lockage? Since these groups alternate in directions, lock scheduling has a bi-directional character, in particular when locks are arranged in sequence like at the Mississippi River (Campbell et al., 2007). Hence, even though important differences in the concrete models occur, the high level structure exhibits similarities.

Strategic or tactical questions in waterway transportation, e.g., distributing investments over time, are often tackled by simulations, see e.g., Carroll and Bronzini (1973), Mitchell et al. (2013), Schonfeld and Ting (1998), or Schonfeld and Wang (2005). Righini (2016) uses a network flow model on the Northern Italy waterway system to analyze the maximum freight transportation capacity. The author emphasizes that long stretches operated bi-directionally cannot be covered by the model.

There are several applications of *conflict-free routing* like scheduling industrial robots (Rambau and Schwarz, 2010, Skutella and Welz, 2011). We mentioned already that we do not require reserving network arcs entirely, but only entry times. This unlocks optimization potential in such applications.

Last, but certainly not least, our work relates to scheduling trains on a stretch of single-track segments, or *single-track train scheduling*. However, there are crucial differences, e.g., trains can never (single-track) or always (double-track) meet at segments while our passing rules are general. Models, methods, and results from train scheduling do not directly apply to our combinatorially richer situation. Conversely, our work could be particularly useful for freight trains that do not operate according to a timetable, see e.g., Jaumard et al. (2013). Harbering et al. (2015) survey train scheduling in different variants and study the computational complexity. The interpretation as a job-shop scheduling problem, with trains as jobs and segments as machines, dates back to Szpigel (1973). From the machine scheduling point of view the STCP generalizes scheduling with family setup times (Potts and Kovalyov, 2000) and flow-shop or restricted job-shop scheduling (Lawler et al., 1993). The routing component in train scheduling, if relevant, is typically handled by selecting (alternative) routes from a pre-computed set according to some priority rule (D'Ariano et al., 2008). The general idea of alternately solving a (simpler) scheduling and a (simple) routing problem is also considered e.g., by D'Ariano et al. (2008). An online situation similar to ours arises in real-time train dispatching when one has to immediately deal with delays and disruptions, see e.g., D'Ariano et al. (2008), Lamorgese and Mannino (2015), and references therein. Lusby et al. (2011) give an overview on conflict-free allocation of railway track capacity over time in general networks. None of

the existing approaches handles the presence of arbitrary start and target positions on segments, nor an arbitrary conflict graph between trains/ships, nor continuous/precise halting positions in sidings. In particular, we systematically avoid loss of precision and error propagation in computations by not using a time/space discretization commonly seen already at the modeling stage. It is important to note that the computational complexity strictly increases from polynomial time solvable (for train scheduling, with all trains in conflict) to APX-hard (for ship scheduling, with arbitrary pairs of ships in conflict) when considering only a single segment (Disser et al., 2015). This suggests that the STCP is considerably harder than train scheduling.

3. Scheduling on Transit Segments

In this section, we focus on the combinatorial decisions to be taken on transit segments. We ignore all constraints in sidings, in particular capacities. This is called the *combinatorial relaxation*. In this relaxation, all parking distances and all vertical distances for sidings are zero, i.e., $v_{ijT} = 0$ for all $i, j \in S$ and $T \in \mathcal{T}$, and $h_i = 0$ for all $i \in S$. Already this relaxation is NP-hard by a reduction from single-machine scheduling (Disser et al., 2015).

3.1. Understanding the Combinatorics

Ignoring all constraints in sidings, we can derive a solution just from the *time* a ship enters each segment, as any waiting position is feasible. A mixed integer program (MIP) describes the combinatorial relaxation, see Figure 7. Let E_i^+ denote the next segment after $E \in \mathcal{E}$ in ship i's heading, if it exists. *Visit time variables* d_{iE} specify when ship $i \in S$ enters segment $E \in \mathcal{E}$. On transit segments, visit times are linked by the transit time θ_{iE} in Equation (2). On siding segments $T \in \mathcal{T}$, waiting is allowed which is reflected by *waiting time variables* $w_{iT} \geq 0$ for each ship $i \in S$. Equation (3) links visit and waiting times in sidings. Lower bounds (6) on visit times enforce release times.

When (opposed or aligned) ships i and j are not allowed to be concurrently at the same position of segment E, they have a *conflict* on E (this is equivalent to $v_{ijE} > 0$). Denote by $C_E \subseteq S \times S$ the set of all conflicting pairs of ships on E. We say that ship i *precedes* ship j on segment E if i passes *each position* of E before j, i.e., $\overline{\tau_i}(p) < \underline{\tau_j}(p)$ for all $p \in E \cap [s_i, t_i] \cap [s_j, t_j]$. For a conflicting pair $(i, j) \in C_E$, it must be decided, which ship precedes the other. This *scheduling decision* is done by binary z_{ijE}-variables: A value of one reflects that i precedes j on E. Figure 8 shows that precedence can be involved. Define $\Delta(i, j, E)$ (or simply $\Delta(i, j)$ if the segment is obvious) as the smallest value such that the condition

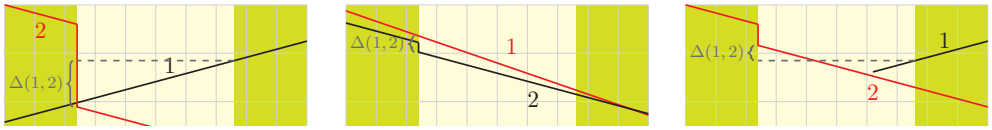
$$d_{iE} + \Delta(i, j, E) \leq d_{jE} \tag{9}$$

ensures the required vertical distance v_{ijE} when ship i precedes ship j on E. The Δ need not be symmetric, see Figures 8(b) and 8(c). Figure 8(c) also shows that the visit time d_{iE} of the preceding

$$\begin{aligned}
&\text{minimize} && \sum_{i\in S,\, T\in\mathcal{T}} w_{iT} && && (1)\\
&\text{s.t.} && d_{iE}+\theta_{iE} &= d_{iE_i^+} && \forall i\in S,\, E\in\mathcal{E}\setminus\mathcal{T}, E_i^+\in\mathcal{E} && (2)\\
& && d_{iT}+\theta_{iT}+w_{iT} &= d_{iT_i^+} && \forall i\in S,\, T\in\mathcal{T}, T_i^+\in\mathcal{E} && (3)\\
& z_{ijE}=1 \;\Rightarrow\; && d_{iE}+\Delta(i,j,E) &\le d_{jE} && \forall E\in\mathcal{E}\setminus\mathcal{T},\, (i,j)\in C_E && (4)\\
& z_{ijE}=0 \;\Rightarrow\; && d_{jE}+\Delta(j,i,E) &\le d_{iE} && \forall E\in\mathcal{E}\setminus\mathcal{T},\, (i,j)\in C_E && (5)\\
& && \underline{d}_{iE} &\le d_{iE} && \forall i\in S,\, E\in\mathcal{E} && (6)\\
& && 0 &\le w_{iT} && \forall i\in S,\, T\in\mathcal{T} && (7)\\
& && z_{ijE} &\in \{0,1\} && \forall E\in\mathcal{E}\setminus\mathcal{T},\, (i,j)\in C_E && (8)
\end{aligned}$$

Figure 7 Mixed integer program for the combinatorial relaxation in which all feasibility conditions concerning passing and waiting in sidings are relaxed. Constraints (4) and (5) model Equation (9) and can be linearized using a "big M" formulation.

ship i can even be *later* than the visit time d_{jE} of the succeeding ship j. Hence, $\Delta<0$ is possible. Also in this case, Condition (9) defines a lower bound for d_{jE} depending on the value of d_{iE}.



(a) (b) (c)

Figure 8 Examples for calculating Δ: Since ship 1 of Figure (b) is slower than ship 2, $\Delta(1,2)>\Delta(2,1)$. Special attention is necessary when a request starts or ends on a segment: This also yields asymmetric Δ values that can even be negative, see Figure (c).

According to the decision which ship precedes on a segment, either Condition (9) must hold as stated, or with the roles of ships i and j reversed. Implications (4) and (5) model this dichotomy. An assignment of the z variables defines for each transit segment $E\in\mathcal{E}\setminus\mathcal{T}$ a partial order $\rho(E)\subset S\times S$ on the set of ships where $(i,j)\in\rho(E)$ if and only if $(i,j)\in C_E$ and ship i precedes ship j on segment E, i.e., $z_{ijE}=1$. We refer to the collection (not the union) $\{\rho(E)\mid E\in\mathcal{E}\setminus\mathcal{T}\}:=\rho$ of these partial orders as the *combinatorial structure* of a solution or its *combinatorics* for short. Visit times d_{iE} for all ships and segments that respect Implications (4) and (5) yield a geometry which we call a *realization* of ρ. If such a geometry exists ρ is *realizable*. This can be checked using MIP (1)–(8): Once all precedences are decided, i.e., all z_{ijE} variables are fixed, optimal visit times are obtained by solving the resulting linear program—or the combinatorics is proven to be unrealizable.

Remark 2. The LP relaxation of MIP (1)–(8) is very weak. Since $z_{ijE}=0.5$ for all i,j,E is a feasible fractional solution, one obtains the trivial lower bound of zero. The model is still well

suited to describe and understand the combinatorial structure of the problem and good enough for experiments on small instances. We therefore do not discuss models with stronger relaxations here.

REMARK 3. We noted that only certain pairs of ships are in conflict and that ships can start and end virtually everywhere in the canal. This entails a variety of configurations of precedence, see again Figure 8. The conflict sets and the Δ values enable us to give a compact formulation by using just one unified statement, namely Condition (9), to ensure precedence between conflicting ships. Moreover, this gives us a way to consider only a segment's boundary to ensure precedence *on the whole segment*. Note that we can simply define visit time variables d_{iE} for each ship $i \in S$ and for each segment $E \in \mathcal{E}$ even if the start point of E is not included in the request interval $[s_i, t_i]$. This is done via translation of the time values between the interval boundaries and the start point of E (w.r.t. i's velocity and heading). That is, we think of each ship as traversing the whole canal, even though conditions concerning conflicts are active precisely on the ship's request interval only.

3.2. Realizing a Combinatorial Structure via Iterated Routing

We have just seen that optimal dynamic routes can be constructed by linear programming, once the scheduling decisions are fixed. However, as we show next, this *global* view, i.e., *simultaneously* deciding about all visit times is not necessary. Rather, *local* decisions for a single ship at a time suffice to create optimal collision-free dynamic routes for the combinatorial relaxation.

Assume we are given dynamic routes $\mathcal{P}^*$ for a subset $S^* \subseteq S$ of ships, and we would like to construct a dynamic route for another ship $i \notin S^*$, respecting $\mathcal{P}^*$. This can be done *greedily*. Define the visit times of i consecutively for each segment as early as possible after its arrival without violating vertical distances to dynamic routes in $\mathcal{P}^*$. This yields a collision-free dynamic route with minimum waiting time for ship i w.r.t. $\mathcal{P}^*$ and takes polynomial time. When a conflict occurs on a transit segment E, traversing E as early as possible implies waiting at the end of a siding, directly in front of E. This is feasible as constraints on parking and vertical distances are relaxed in sidings.

We focus on solutions with a structure motivated by this greedy routing. We call a dynamic route $P_i \in \mathcal{P}$ to be *earliest arrival* w.r.t. the other dynamic routes in $\mathcal{P} \setminus \{P_i\}$, if the arrival time $\underline{\tau_i}(p)$ of P_i is at each position $p \in [s_i, t_i]$ as early as possible with respect to $\mathcal{P} \setminus \{P_i\}$ preserving collision-freeness and the solution's combinatorics. Thus, ship i waits at no position longer than necessary, and all waiting of i takes place as late as possible. Note that the earliest arrival route of a ship with respect to the others and their combinatorics is unique. A solution $\mathcal{P}$ is called *earliest arrival* if each dynamic route of $\mathcal{P}$ is earliest arrival w.r.t. the other dynamic routes of $\mathcal{P}$. A solution can be turned into an earliest arrival solution by iteratively turning dynamic routes into earliest arrival ones in an arbitrary order (that may contain cycles) without increasing the waiting time of any ship. Observe

that greedy routing produces an earliest arrival route for a single ship with respect to the already planned ones. We next modify greedy routing to additionally respect given predecessors of the current ship on each segment by defining the visit times of each segment as early as possible without violating them. We use this fact to produce earliest arrival solutions given only the combinatorics.

THEOREM 1. *For our combinatorial relaxation, there is a polynomial time algorithm using greedy routing that, given an instance (C, S, R) and a combinatorial structure ρ on S, either creates a feasible earliest arrival solution $\mathcal{P}$ realizing ρ or proves that no realization exists.*

We constructively prove the theorem by describing Algorithm 1. In Section 5.2 we generalize it to respect siding constraints. Define a *partial dynamic route* of ship $i \in S$ to be a dynamic route of i with the relaxation that the first and the last x-coordinates can be any points in $[s_i, t_i]$. A partial dynamic route of ship $i \in S$ is called a *prefix route* if the first x-coordinate equals s_i and a *suffix route* if the last x-coordinate equals t_i. A prefix route that is not a suffix route is *incomplete*. Consider some $(j, i) \in \rho(E)$ for a transit segment E, i.e., ship j has to precede ship i on E. To ensure this precedence the visit time of j on E must be known before deciding about the visit time of i on E via greedy routing. This can be achieved by successively extending prefix routes by partial dynamics routes until a complete dynamic route is obtained, instead of producing it in one step. This iteration is described in Line 4 of Algorithm 1:

1. Let p_i be the last x- and τ_i be the last y-coordinate of i's current prefix route P_i.
2. Find a collision-free suffix route P_i' for i with start position p_i and start time τ_i that respects the current partial routes $\{P_j \mid j \in S\}$ and the given combinatorics ρ.
3. Let E be the first transit segment on P_i' with an *open predecessor* j of i, i.e., $(j, i) \in \rho(E)$ and E is not yet contained in the current prefix route P_j of ship j.
4. Extend the prefix route P_i by the part of P_i' that starts in p_i and ends in p' ("route extension"), where p' is determined as follows: (a) if E does not exist, let $p' = t_i$; (b) if $p_i \in E$, let $p' = p_i$; (c) otherwise, set p' to the beginning of the siding before E on P_i.

This yields a proper extension of P_i if and only if $p' \neq p_i$. In this case we call ship i *free*. This can be tested in polynomial time.

LEMMA 1. *Algorithm 1 produces an earliest arrival solution realizing ρ if and only if ρ is realizable for the instance (C, S, R).*

Proof. If Algorithm 1 returns a solution, it is collision-free, earliest arrival, and realizes ρ by construction. Consequently, ρ is realizable.

For the converse direction think of Algorithm 1 as timing the events of ships passing a segment. Assume that there is a realization $\mathcal{P}$ of ρ. W.l.o.g. $\mathcal{P}$ is an earliest arrival solution. We prove the following loop invariant for Algorithm 1:

input: canal C, requests R for ships S, combinatorics ρ

output: an earliest arrival solution $\mathcal{P}$ realizing ρ or the information that none exists

for each $i \in S$ initialize a partial dynamic route $P_i := ((s_i, r_i))$
while there are ships S' with incomplete dynamic route **do**
  if there is a free ship $i \in S'$ **then**
    extend P_i by greedy routing of i as far as possible with respect to ρ
  else
    return "there is no realization for ρ"
return $\{P_i \mid i \in S\}$

Algorithm 1: Realizing the given combinatorics in our combinatorial relaxation.

Claim 1. *As long as $S' \neq \emptyset$ there is at least one free ship $i \in S'$ in each iteration and the events that yield the route extension of P_i are timed in the same way as in $\mathcal{P}$.*

This implies that the algorithm terminates with a complete collision-free dynamic route for each ship which finally equals the one of $\mathcal{P}$.

The proof is by induction over the number of iterations. The base case is obvious. For the inductive step assume that the claim holds for all previous iterations and $S' \neq \emptyset$. By contradiction, suppose that there is no free ship. Both, ρ and the ships' headings impose precedences on the passing events of ships on segments. Thus, we can assign a time to an event only if all its predecessors are timed. If there is no free ship, all un-timed events must have un-timed predecessors, implying a cyclic dependency on events. However, since ρ is realizable, this is a contradiction. Thus, there must be a free ship i in this iteration.

We now want to time events of ship i. Since ship i is free, all preceding events are timed in earlier iterations, and, by the inductive hypothesis, identically to those in $\mathcal{P}$. Since we use greedy routing, the route extension of ship i is earliest arrival and unique. Hence, the events of ship i are timed identically to those in $\mathcal{P}$. Q.E.D.

For a realizable combinatorial structure of a given instance the output of Algorithm 1 is well-defined; we thus conclude from the above induction hypothesis:

Corollary 1. *For any feasible solution to our combinatorial relaxation there exists a unique earliest arrival solution which can be constructed by Algorithm 1.*

The local view on routing that we have adopted in this subsection will greatly help us in the following to incorporate feasibility constraints concerning passing and waiting in sidings when dealing with the original, unrelaxed problem.

4. Collision-free Routing for a single Ship

Given collision-free dynamic routes $\mathcal{P}'$ for a subset of ships $S' \subseteq S$, we want to construct a dynamic route for *a further ship* $i \in S \setminus S'$ which is collision-free and minimizes its waiting time w.r.t. the already given $\mathcal{P}'$. With siding conditions relaxed (Section 3.2) this insertion was easy via greedy routing. In the original setting, more work is needed: First, if a conflict occurs, the siding directly in front of it may be fully occupied such that the waiting must take place at an earlier siding. It is thus not always possible to use the earliest available time to traverse a transit segment. Second, the routing algorithm must now take spatial decisions concerning collision-free waiting within sidings.

Our setting is related to the successive routing of automated guided vehicles (AGVs) where transportation requests arise over time in a network. The routing algorithm by Gawrilow et al. (2008) answers the respective next request, avoiding collisions to previously routed AGVs. We extend their idea to plan ship i optimally with respect to the dynamic routes in $\mathcal{P}'$. Shortest path problems with time windows are widely studied in the vehicle routing literature, see Desrosiers et al. (1995), Solomon and Desrosiers (1988) for overviews. The permanent labeling algorithm developed by Desrochers and Soumis (1988) (see also Desrosiers et al., 1995) is the basis for the algorithm of Gawrilow et al. (2008) which runs in polynomial time for the objective of finding a *quickest* path respecting time windows, where quickest refers to elapsed time, i.e., travel time plus waiting.

Like in Dijkstra's algorithm (Dijkstra, 1959) we maintain labels representing arrival times of the ship. However, instead of referring to a single point in time, our labels correspond to feasible arrival time *intervals*. Further, for each arc, we store *forbidden time windows* during which it is not allowed to enter that arc in order to avoid collisions with conflicting ships. Together, this gives an *implicit* time-expansion of the network and avoids the traditional way of guaranteeing collision-freeness by deleting arcs from a time-expanded network (Ford and Fulkerson, 1958, 1962, Skutella, 2009). When exploring the outgoing arcs of a node the label intervals are adapted according to the forbidden time windows. The collection of labels then covers all possibilities to traverse a transit segment, not only the earliest one. This resolves the first algorithmic issue stated in the introduction of this section.

Algorithm 2 gives an overview of our quickest path computation with time windows. A *label* $lab = (a, [epat, lpat], pred, \ell)$ represents a *set* of collision-free dynamic routes for ship i, from its start position s_i to the target node of arc a, with an interval of earliest ($epat$) and latest ($lpat$) possible arrival times, and a reference to a predecessor label $pred$ of lab on all represented routes. Using $pred$, a route with arbitrary arrival time in $[epat, lpat]$ can be reconstructed recursively. Since the length ℓ of some arcs will be adapted dynamically by the algorithm, each label holds ℓ as a further component. It is possible that a label interval degenerates to a single point in time. The precise construction of the network $G(C)$ and the maintenance of forbidden time windows $\mathcal{F}(.)$ is described later.

input: canal network $G(C)$, routing request $r_i \in R$, time windows $\mathcal{F}(.)$
output: quickest path P_i w.r.t. $\mathcal{F}(.)$

```
1  enqueue start-labels for s_i in priority queue PQ
2  while PQ is not empty do
3      cur := best label w.r.t. ≺ dequeued from PQ
4      if cur corresponds to t_i then
5          return reconstructed dyn. route for cur with earliest possible arrival time
6      foreach successor arc succ of label cur do
7          foreach label next resulting from propagation of cur along succ w.r.t. F(.) do
8              if next is not dominated by an active label of succ then
9                  enqueue next in PQ
10                 remember next as an active label of succ
11                 delete active labels of succ that are dominated by next also from PQ
```

Algorithm 2: Calculate a quickest path for ship i avoiding given forbidden time windows $\mathcal{F}(.)$ within a canal network $G(C)$.

Like in Dijkstra's algorithm, the set of all labels is organized in a priority queue. We use a natural order $\prec$ on the labels which is induced by the earliest possible arrival times *epat*. For labels which are identical according to this order, preference is given to certain properties, see below. Controlled by this earliest-label-first order, the arcs are explored starting from labels corresponding to the arc that contains the requested start position until an arc containing the target position is reached. Two important sub-tasks distinguish the algorithm from Dijkstra's classic (details follow):

1. The forbidden time windows are respected during the exploration of an arc resulting in a set of split labels. Furthermore, for arcs corresponding to a waiting track, the label interval is expanded to represent the waiting. Together, this is called *propagation*.
2. Comparing two labels corresponding to one arc, we cannot always determine which one will produce no better solutions than the other. In this case, both must be remembered for further consideration. Otherwise, a *dominated* label can be deleted.

In the remainder of this section we specialize to our canal setting: define a network, develop time windows guaranteeing collision-freeness, and describe the distinct steps of Algorithm 2. In particular, this answers the second algorithmic challenge concerning waiting decisions within sidings. For more details we refer to Lübbecke (2015).

4.1. Network for collision-free Routing

The quickest path computation must take place in an appropriate network. We have just seen that we can avoid a time-discretization. The next question is how to model sidings and their waiting

tracks. Recall that a ship traverses a siding on the passage track, changes to the waiting track at the defined waiting position, and later continues on the passage track. It seems natural to partition the waiting track into a discrete set of "parking lots," or waiting arcs. However, with a fixed length, short ships consumed more waiting capacity than necessary. If we partitioned the waiting track in many short waiting arcs, a parking ship would use several arcs, which is not compatible with our local label propagation (Gawrilow et al. (2008) have a workaround for AGVs which have identical length, contrasting our setting with individual ship lengths varying from 50 to 250 meters). Finally, no partitioning could ensure that a ship waits at most once per siding. For these reasons, we do not use a spatial discretization of the waiting tracks at all. Instead, we let our algorithm dynamically consider waiting positions as necessary.

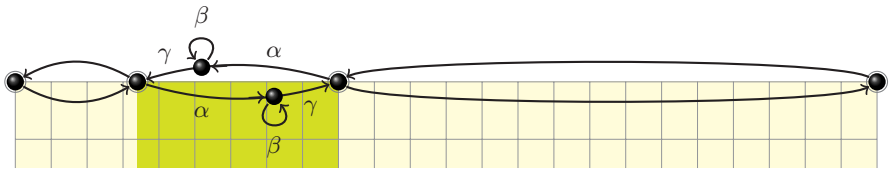


Figure 9 Network $G(C)$ representing canal C. In a siding, different types of arcs reflect time/space before (type α), while (type β), and after (type γ) waiting.

Figure 9 displays how we translate a canal into a network. A node is introduced on the boundary points of each segment. For each transit segment, these two nodes are connected by an up- and a downstream arc, respectively. In order to model the possibility to wait on the waiting track, we introduce a further node per direction within each siding. The waiting track itself is represented by a loop adjacent to this node. In total, there are three arcs per siding and direction, one of *type* α for the passage before waiting, one of *type* β for waiting on the waiting track, and one of *type* γ to continue the passage after waiting. In contrast to the nodes at the segment boundaries the node representing waiting in a siding has no explicit embedding: The waiting arc does not correspond to a fixed waiting position but will hold information about all ships of $\mathcal{P}'$ waiting within that siding. As part of the propagation process along arc α all necessary but not more positions for the waiting node will be checked depending on this information. Given a canal C, we denote this network as $G(C)$. All the potential complexity of this network—we neither have time nor space discretization—is avoided by entirely shifting the information management to the algorithm.

4.2. Forbidden Time Windows

Consider a transit segment $E \in \mathcal{E} \setminus \mathcal{T}$. As suggested by Equation (9) the dynamic route of ship $j \in S'$ (in conflict with the new ship i on E) defines an upper bound $u_j := d_{jE} - \Delta(i, j, E)$ on the visit time

of i on E before the passage of ship j. Similarly, a lower bound $\ell_j := d_{jE} + \Delta(j, i, E)$ for visiting E after j arises. This results in a *time window* $F_j := [u_j, \ell_j]$ implied by the dynamic route of j during which it is *forbidden* for ship i to enter E, i.e., we have to ensure that $d_{iE} \notin F_j$. As before, the interval is with respect to the start position of segment E to have a reference point independent of s_i. In this sense, we can determine for each arc and each dynamic route in $\mathcal{P}'$ time windows that restrict the use of this arc to ensure a collision-free dynamic route for i.

Now consider a siding $[x, y] =: T \in \mathcal{T}$. A ship that waits in T does not block the passage track of T. Therefore the passage of that track before waiting and the passage after waiting must be considered separately. Consider a ship $j \in S'$ with given dynamic route that waits at position $p_j \in T$ and has a conflict with i. If ship i's waiting position p_i is, say, in front of p_j we must ensure that the passage of ship i before its waiting does not collide with the passage of j before its waiting and furthermore with the passage of j after its waiting. Hence, depending on the considered waiting position p_i ship j can induce two forbidden time windows for the corresponding arc of type α or two for the corresponding arc of type γ. For more details confer Figure 10. Additionally, the waiting ship j produces a forbidden time window on the waiting arc of T (type β) for the interval $\tau_j(p_j)$ if the waiting positions p_j and p_i have a distance smaller than $h_i + h_j$.

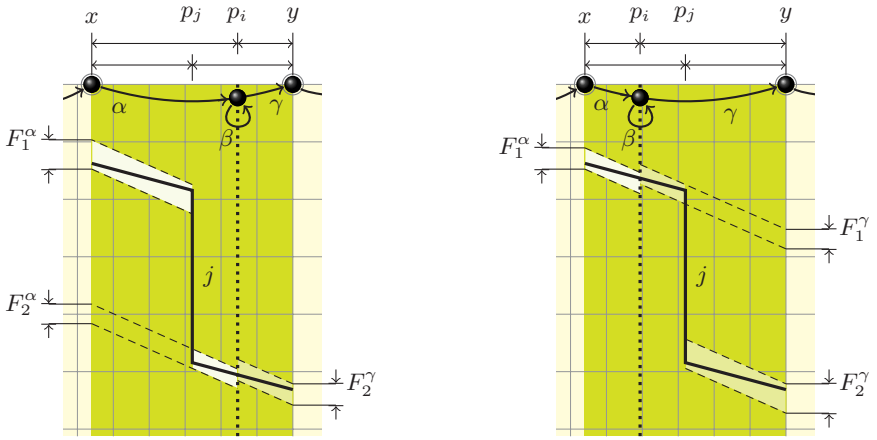
To summarize, the set of dynamic routes $\mathcal{P}'$ yields sets of forbidden time windows $\mathcal{F}(a, i, p_i)$ for siding arcs a, and sets of forbidden time windows $\mathcal{F}(a, i)$ for transit arcs a. These restrict the use of a for ship i as described, possibly depending on the given waiting position p_i.

As already mentioned, we want to have fast local access to the set of time windows of an arc a during the propagation along a. To this end, each arc holds for each given dynamic route $P_j \in \mathcal{P}'$ the respective local information to define a time window (the visit time of the segment, possibly the waiting position within the siding, and a link to the properties of the corresponding ship j). Since each time window additionally depends on properties of ship i it can be calculated in constant time.

4.3. Routing Details for the Canal

Propagation. The core task of propagating a label *cur* along an arc *succ* with respect to $\mathcal{F}(.)$ (Line 7 of Algorithm 2) is the splitting of a label interval corresponding to the given forbidden time windows into a set of new labels for arc *succ* guaranteeing collision-free dynamic routes. This is described in Algorithm 3 whose input must be chosen according to the type of the arc *succ*.

If *succ* corresponds to a transit segment E, the algorithm is called with $newend := lpat(cur)$, F as union of $\mathcal{F}(succ, i)$, label length $|E|$, and transit time θ_{iE}. Since F contains each point in time at which ship i must not enter E, each time value of $\mathcal{I}$ is a feasible entering time for i. Since the new labels correspond to the target node of *succ*, the time intervals must be translated to this position.



(a) The tested waiting position p_i is behind the fixed waiting position p_j.

(b) The tested waiting position p_i is in front of the fixed waiting position p_j.

Figure 10 A given dynamic route of ship j defines a set of time windows in sidings. If ship i is supposed to wait behind ship j (part (a)) there are two time windows where i is not allowed to enter T. Otherwise (part (b)) ship j induces two time intervals where ship i is not allowed to continue its passage after waiting. To determine the boundaries of the forbidden time windows we make use of $\Delta(\cdot)$ as above. Since it is possible that ships i and j have different velocities, we must restrict the domains of i and j valid for the calculation of $\Delta(\cdot)$ for the distinct cases (indicated by the light areas): F_1^α corresponds to $[x, p_i]$ and $[x, p_j]$, F_2^α to $[x, p_i]$ and $[p_j, y]$, F_1^γ to $[p_i, y]$ and $[x, p_j]$, and finally F_2^γ to $[p_i, y]$ and $[p_j, y]$ if the respective intervals are not disjoint (cf. Figure 8). If the start or target position of i or j is situated within this siding, the request intervals must be further reduced appropriately. For clarity's sake we located the time windows of the arcs of type γ in this picture w.r.t. the end of T. Note that we actually define them translated to the start of T such that they are compatible with all other definitions.

Is *succ* of type α for a siding T, we split several times, once for each *interesting* waiting position p_i: Each iteration calls Algorithm 3 with $newend := lpat(cur)$, $F :=$ union of $\mathcal{F}(succ, i, p_i)$, label length p_i, and the resulting transit time for this length as input. To determine all interesting waiting positions, we consider each maximal time interval with invariant waiting conditions within T from the earliest arrival of *cur*. In this, each free waiting area will be filled with waiting positions respecting the horizontal parking distances with h_i as step size from the end of the siding to its beginning until one waiting position yields a label who's interval would not be reduced by the splitting process on any of the three siding arcs. All further waiting positions would not yield further possible arrival times at the end of the siding and hence, would not yield a better solution. This often reduces the number of constructed labels drastically and is important for an acceptable running time.

The input value *newend* becomes important if *succ* is of type β. In this case, the waiting will be modeled by an extension of the interval to $newend := \infty$. After this extension it will be split by F as union of $\mathcal{F}(succ, i, \ell(cur))$ since $\ell(cur)$ is the tested waiting position. To provide this information also for the next arc, the label length is also set to $\ell(cur)$ (and not 0). But the transit time is 0.

input: arc *succ*, label *cur*, time value *newend*, union of time intervals F, label length ℓ, transit time θ

output: set of feasible labels for *succ* of length ℓ with *cur* as predecessor

1 $\mathcal{I} := [epat(cur), newend] \setminus F$ `// open time windows`

2 $\mathcal{L} := \emptyset$

3 **foreach** maximal interval $[epat', lpat']$ of $\mathcal{I}$ with $epat' \leq lpat(cur)$ **do**

4 $\quad \mathcal{L} := \mathcal{L} \cup (succ, [epat' + \theta, lpat' + \theta], cur, \ell)$

5 **return** $\mathcal{L}$

Algorithm 3: Splitting a predecessor label along an arc *succ* into a set of feasible labels for *succ*.

Finally, *succ* could be of type γ. In this case, label *cur* can correspond to arcs of both types α and β. The start node of *succ* then has the corresponding position and hence, the given label length must be $|T| - \ell(cur)$ and the transit time, respectively. Furthermore, each time window of $\mathcal{F}(a, i, \ell(cur))$ must be translated by $\ell(cur)$ w.r.t. the velocity of i for the construction of F. The value *newend* is again set to $lpat(cur)$.

Dominance. A label *lab* is said to *dominate* another label *lab*′ if both represent the same position in the canal, and *lab*′ cannot lead to any better routes than *lab*. Dominated labels can be deleted. More precisely, label *lab* dominates *lab*′ if and only if both correspond to the same arc a, both correspond to the same length ℓ if a is of type α or β, and

$$lab \preceq lab' \qquad \text{and} \qquad [epat(lab'), lpat(lab')] \subseteq [epat(lab), lpat(lab)].$$

Testing the dominance of labels (Lines 8 and 11 of Algorithm 2) helps to reduce the number of labels that must be propagated without losing any dynamic route of minimum waiting time.

Resulting Dynamic Route. The first constructed label that corresponds to an arc containing the target position of ship i (Line 5 of Algorithm 2) represents a *set* of dynamic routes with smallest possible arrival time and hence, with minimum waiting time. It may still be open in which of the sidings in front of an occurred conflict to spend the waiting time. Each route in the set can be constructed recursively from the last label in reverse order. Hence, we have the freedom to choose among these optimal dynamic routes and we decide to let waiting always take place within the latest possible siding. Also, the ordering $\prec$ of the priority queue prefers late waiting positions within each siding. Consequently, in the case of relaxed siding conditions, Algorithm 2 produces the same earliest arrival solutions as greedy routing. After the reconstruction of the dynamic route of i we update the corresponding information on each arc for future calculations of time windows.

When there is not enough capacity available for waiting within the sidings, the priority queue runs out of labels before a label corresponding to the target position is reached. In that case we conclude that a dynamic route for ship i does not exist.

Running Time. The running time of Algorithm 2 depends on the number of waiting positions that need to be tested, which is strongly related to the maximum ratio ρ_{h_i} of siding length and horizontal distance of ship i. Extending the argumentation by Gawrilow et al. (2008), we can show that the running time is polynomial in the maximum number of time windows η per arc, the number of segments, and ρ_{h_i}. Note that η is bounded by $2n$ and is often much smaller. For the real world instances coming from the Kiel Canal each ratio ρ_{h_i} is bounded by a constant.

Knapsack Relaxation. In the combinatorial relaxation, a solution can be realized from the given combinatorics via greedy routing (Theorem 1). When we additionally impose the siding capacity as a knapsack constraint, i.e., in the knapsack relaxation, it might be impossible for each ship to wait directly in front of a conflict. Thus, greedy routing may produce infeasible solutions. Even so, Theorem 1 can be extended to this stronger knapsack relaxation (which is relevant in the case of all identical ship dimensions) when using collision-free routing instead of greedy routing: only time windows and propagation on arcs of type β need to be slightly adapted.

COROLLARY 2. *For the knapsack relaxation, there is a polynomial time algorithm using collision-free routing that, given an instance (C, S, R) and a combinatorial structure ρ on S, either creates a feasible earliest arrival solution $\mathcal{P}$ realizing ρ or proves that no realization exists.*

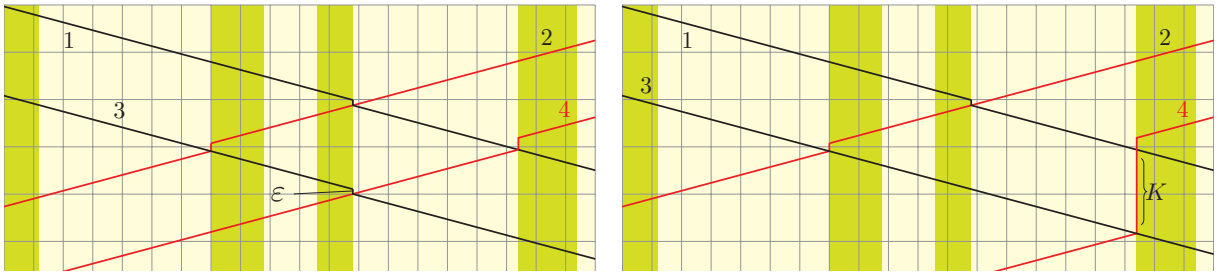
5. A Heuristic for the STCP

5.1. Constructing Solutions by Sequential Routing

To construct an initial solution we iteratively route ships through the canal, one after another, using collision-free routing (a natural idea used e.g., by Gawrilow et al., 2008). We sort ships by non-decreasing times they (would have) entered the canal (i.e., the translation of each release date from the start position to the canal boundary, if needed). This is a canonical ordering but it is somewhat arbitrary, and there may be better orderings which give better overall solutions. However, the question for a *best* ordering of such a *sequential routing* is void in view of the following lemma.

LEMMA 2. *Any sequential routing procedure can yield arbitrarily bad solutions, even if all horizontal distances between ships are zero.*

Proof. Figure 11 sketches an instance where in each optimal solution *each* ship has to wait for a short time of, say, at most ε. Such a solution contains a cyclic waiting pattern which cannot be achieved via any sequential routing procedure: at least the ship which is routed first will not wait anywhere. However, such a ship causes another ship to wait for at least $K \gg \varepsilon$. With $\varepsilon \to 0$, one cannot avoid an arbitrarily large deviation from the optimum when routing sequentially. Q.E.D.



(a) Each ship is waiting for a small amount.

(b) One ship without waiting induces high waiting time for another.

Figure 11 Example where sequential routing yields high waiting time independent of the chosen routing sequence.

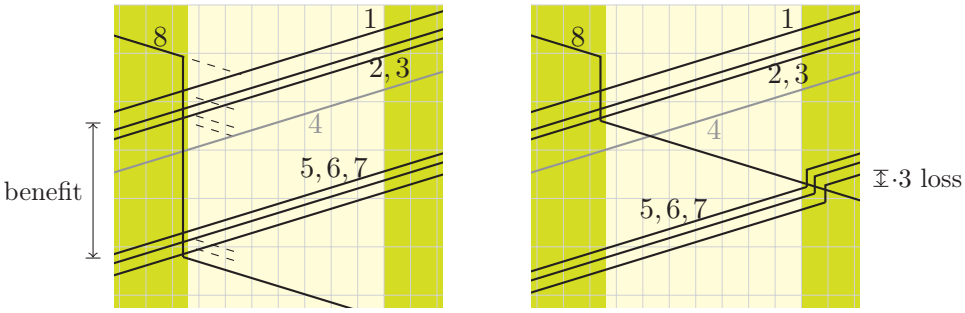
5.2. Improving Schedules by Local Search on the Combinatorics

We have seen earlier that scheduling decisions, i.e., who is waiting for whom and where, constitute the core difficulty of our problem. The most important insight from Lemma 2 is that, regardless of the ordering of ships, sequential routing is limited in the structure of schedules that it can produce. Actually, it does not actively work on a schedule at all, i.e., the combinatorics of the solution comes as a by-product. We thus need to find ways to produce different, *richer* combinatorial structures, and we need to be able to deal with this in conflict-free routing. We address the second item first.

Algorithm 1b. A given combinatorial structure may be impossible to realize because of the siding capacities. The following modification of Algorithm 1 succeeds in producing a realization if this is possible. However, we allow to alter or even eliminate some scheduling decisions if that should be necessary. When in Algorithm 1, Line 4, a route cannot be extended for ship i due to insufficient space for waiting in the preferred siding, we backtrack: Try the route extension from one siding earlier, dismissing the already fixed onwards part of the extension. This is iterated until we find a dynamic route for ship i or no earlier siding is available. In the latter case, *no* dynamic route can be assigned to ship i. Note that this backtracking does not touch other ships: a ship which originally succeeded ship i along a re-considered transit segment may change sequence with ship i. We therefore cannot guarantee that the route extension of Line 4 respects a given combinatorial structure in general. However, barring the mentioned complications it does. The backtracking version of Algorithm 1 is called Algorithm 1b. We show below that it always produces feasible solutions.

Combinatorial Neighborhood. We would now like to improve existing solutions by local search, as summarized in Algorithm 4. Instead of working with our geometric encoding of an incumbent solution $\mathcal{P}$ directly, we define a neighborhood on the induced combinatorics $\rho(\mathcal{P})$, slightly abusing notation. This fits well with our conclusions from Lemma 2 since it enables us to produce a much larger variety of schedules in the first place. A neighbor of $\rho(\mathcal{P})$ is constructed by altering scheduling between consecutive sidings. Improvement heuristics on scheduling decisions or conflicts have also

been used for train scheduling on a single track, see e.g., Carey and Lockwood (1995), Higgins et al. (1997). Figure 12 motivates how this is done: every ship i waiting in a siding could switch precedence on the onward transit with (some of) the k conflicting ships it is waiting for. Let $j_1, \ldots, j_k$ be the respective ordering of these ships. All of them currently precede i at this point. Every reordering $\{i, j_1, \ldots, j_k\}$, $\{j_1, i, j_2, \ldots, j_k\}$, $\ldots$, $\{j_1, \ldots, j_{k-1}, i, j_k\}$ induces new precedences. The union of all these, over all waiting ships in all sidings, describes the neighborhood of $\rho(\mathcal{P})$. When several ships wait in a siding for the *same* set of conflicting ships, it makes sense to insert *all* of them simultaneously in $j_1, \ldots, j_k$. This is done in our implementation.



(a) Current waiting situation. (b) New realized schedule.

Figure 12 A waiting ship induces several neighbors in (a); a new incumbent solution in (b). This also illustrates how our local estimate of a neighbor is computed. Assume that ship 4 is not in conflict with ship 8. Each dashed line in (a) represents an opportunity for ship 8 to continue its journey. Realizing one of these would require swapping the scheduling relation for all succeeding ships on this transit segment. The most promising opportunity as in (b) yields a large benefit for ship 8 and three small increased waiting times for the ships 5, 6 and 7.

The described neighborhood may contain combinatorial structures that are not realizable, not even in the combinatorial relaxation of Section 3.1 that ignores siding capacities. Theorem 1 ensures that we detect this defect, and Algorithm 1b can repair it. In Line 7 of Algorithm 4 we ensure that there are free ships as long as there are ships with an incomplete route by heuristically deleting a relation on a cyclic dependency (similar to Figure 11a) when this is detected.

Locally Evaluating the Quality of Neighbors. Representing a solution only by its combinatorics adequately reflects our wish to improve scheduling decisions. At the same time, we have to evaluate the quality of a neighbor ρ': It would be too expensive to use Algorithm 1b only to calculate the waiting time of a solution produced from ρ'. Instead, we only use a simple estimate, see again Figure 12: We compare the decreased waiting time (benefit) of a ship, or a group of ships, with the increased waiting time (loss) of the conflicting ships we delay. This is very myopic, ignoring that reversing scheduling decisions locally can have global effects: because of siding capacities, waiting

time can be saved or produced in a siding distant from the altered scheduling. Only neighbors with best such local estimates are actually evaluated by trying to realize ρ' via Algorithm 1b. Recall that we cannot guarantee that all altered scheduling decisions are respected, and that it may even happen that a ship cannot be routed at all. A solution with best actually realized improvement becomes the next incumbent. In order to escape local optima we allow worsening iterations. A parameter limits the total number of such failures of improvement. We avoid cycling by tabooing bad incumbents.

input: canal C, requests R for ships S
output: solution $\mathcal{P}$
1 construct initial solution $\mathcal{P}$ by sequential routing
2 **repeat**
3 $\quad\rho(\mathcal{P}) :=$ combinatorics induced by $\mathcal{P}$
4 $\quad$estimate benefits/losses of all neighbors of $\rho(\mathcal{P})$
5 $\quad\mathcal{N} :=$ subset of neighbors with best local estimates
6 $\quad$**foreach** neighbor $\rho' \in \mathcal{N}$ **do**
7 $\quad\quad$ensure/repair (combinatorial) realizability of ρ' using Theorem 1
8 $\quad\quad$construct a solution from ρ' using Algorithm 1b
9 $\quad\mathcal{P} :=$ best found solution for candidates in $\mathcal{N}$
10 **until** there is not enough improvement
11 **return** globally best found solution

Algorithm 4: The general scheme of our local search on the combinatorics

Heuristically Encouraging Certain Properties of a Solution. For practical acceptance of our solutions we need to take care of a few additional properties. We therefore do not evaluate solutions only by their total waiting times for several reasons: (a) Algorithm 1b may leave ships unrouted which nominally decreases waiting time; (b) very large waiting times of single ships may decrease that of many other ships, however in practice, waiting in a siding (and also the total waiting time per ship) should not exceed certain soft limits; (c) a large waiting time of a single ship counts as much as the sum of short waiting times for several ships; (d) it is a good idea to give priority to slow ships as they generally produce more scheduling complications in particular for opposed ships. These problems can be lessened by assigning individual weights to ships and by penalizing large individual waiting times. In our implementation this is done by replacing the objective of total waiting time by $\sum_{i\in S}(a_i w_i)^f$ with priorities a_i and $f > 1$. This applies to both, estimating the quality of a neighbor in Line 4 and choosing a best new incumbent in Line 9 of Algorithm 4.

5.3. Rolling Horizon

Finally, we embed the local search in a rolling horizon framework, i.e., we consecutively plan for shorter, overlapping time horizons. There are two good reasons for doing so—an algorithmic and a conceptual one. Algorithmically, this supports the local search as more neighbors can be evaluated when dealing with a smaller instance i.e., a shorter time horizon. Conceptually, we more realistically emulate the current situation at the Kiel Canal where requests arrive over time.

We denote the *horizon length* by T_Δ, the *horizon start time* by T_0, and the *step size* by $T_\delta \leq T_\Delta$. In each iteration we consider only a partial instance of those ships with release time between T_0 and the *horizon end time* $T_0 + T_\Delta$. Note that it is important to construct each dynamic route until the actual target position of the ship even if it will arrive there after $T_0 + T_\Delta$. After constructing a solution for this partial instance, we declare everything before the horizon end time as fixed and define the intersection point of each calculated dynamic route with the corresponding time axis as new start position and release time. Now, T_0 increments to $T_0 + T_\delta$ and the process repeats until all released ships have been considered.

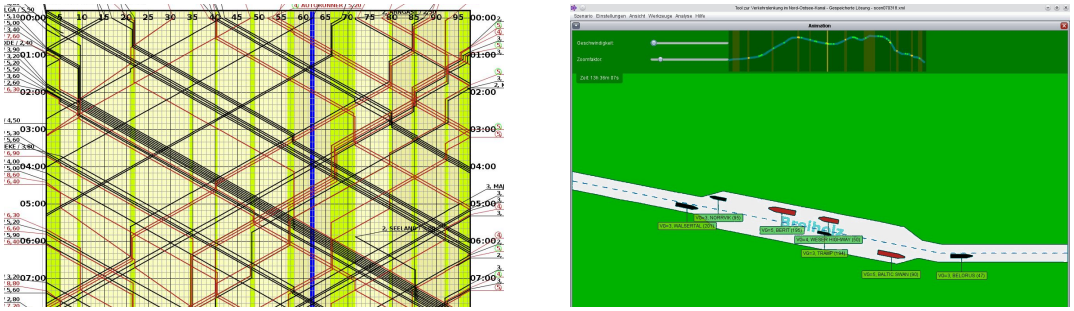
Fixing everything before the horizon start time bears the risk that we may not find a dynamic route for some ships since these cannot be adapted if siding capacities are exceeded later. To decrease this risk, we construct dynamic routes even beyond the current horizon length, and transmit the already taken scheduling decisions ρ from one time horizon to the next.

Experiments in Section 6.1 will show that this results in better objective values, running times and even in fewer ships without assigned dynamic route compared to plain local search.

6. Computational Study

GPS position data (of about 43,000 ships) of actually traveled itineraries for the busy year 2007 were made available to us. From these we generated 365 instances, one for each day with a planning horizon of 24 hours. They contain 185 requests on average (minimum 83, maximum 247). All implementation is in Java. All computations were performed on a commodity desktop PC running Linux. When evaluating the quality of a solution we refer to average waiting time per ship.

We experimented with an explicit discretization using the TS-OPT tool discussed by Schlechte (2012). Even with a coarse discretization of one minute, computation times were exorbitant and we observed a considerable loss in accuracy. Since, by contrast, our algorithm is not affected by the precision with which we measure time, all our calculations can be made exact to the second. We strictly obey all the rules (and a few more details) as given in the problem description, even though in manual planning small violations may occur. It thus happens that an algorithm is unable to find a feasible route for a ship which is then disregarded. This incurs a (penalty) waiting time of 2 hours. Two visualization options in our tool (Figure 13) greatly helped in validating solutions, fine tuning, and getting approval from the expert planners.



(a) Distance-time diagram. (b) Animation of ship movements.

Figure 13 Screen shots from our tool displaying solutions as (a) an interactive distance-time diagram or (b) an animation of the movement of ships on a map of the canal.

6.1. Algorithmic Components

We stressed the interplay of geometry and combinatorics several times. Our first suite of experiments aims at demonstrating that this understanding of the problem also pays off computationally. We evaluate the "purely geometric" approach of sequential routing (Section 5.1, denoted SeqR); sequential routing integrated with re-scheduling decisions in a local search on the combinatorics (Section 5.2, denoted LS-240); and the rolling horizon heuristic which additionally reflects the problem's online character (Section 5.3, denoted RH-1). The numbers refer to the following parameter settings that we chose empirically. LS-240 is allowed a maximal number of 240 worsening steps within the local search over the full 24-hour planning horizon. In RH-1 the horizon length is $T_\Delta := 2$ hours and the step size is $T_\delta := 1$ hour. As the local search acts only on horizons of an hour, we allow a maximum number of 10 ($= 240/24$) worsening steps to make LS-240 comparable to RH-1 in that respect. Furthermore, we greedily evaluate only one promising neighbor within an improvement step, i.e., $|\mathcal{N}| = 1$ in Algorithm 4. The variant with $|\mathcal{N}| = 3$ is denoted by RH-3.

We present four pairwise comparisons of algorithms. In each comparison we compute, for each instance, the ratio of average waiting times obtained by the respective two algorithms and give the whole distribution of ratios as a box-and-whiskers plot in Figure 14. The box marks the range of the mid 50% of the ratios where the central line indicates the median. The whiskers mark the range of 95% and the smallest and largest 2.5% are considered as outliers, marked by circles. The average (marked with an asterisk $*$) and the standard deviation (arrows) are given below the boxes.

Both, the integration of routing and scheduling (LS-240) and the rolling horizon heuristic (RH-1) significantly improve over sequential routing (SeqR) alone. It is remarkable that the local search improvement performs considerably better when it works in small chunks "over time" instead of more globally on the whole day, as shown by the comparison RH-1/LS-240. The last comparison RH-3/RH-1 indicates that it does not pay much to evaluate neighbors in a more elaborate way than

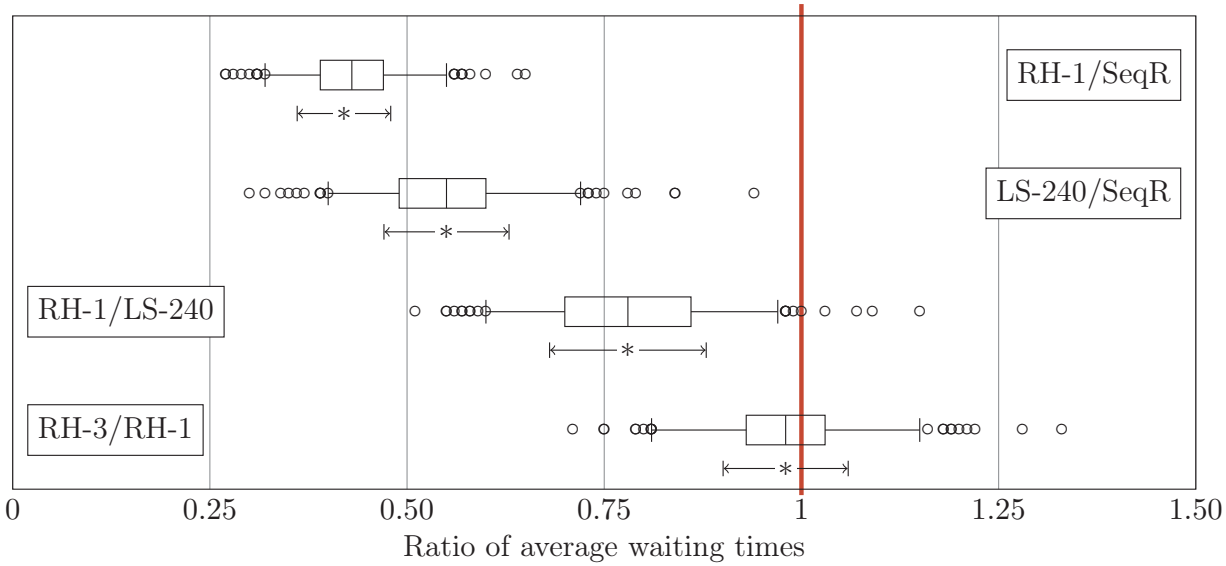


Figure 14 Four pairwise comparisons of algorithmic approaches: Each box-and-whisker plot shows the distribution of instance-wise ratios of average waiting times achieved by the respective two algorithms. A ratio $X/Y < 1$ means that X performs better than Y.

greedily. The most important qualitative conclusion is that taking care of scheduling decisions in addition to sequential routing is imperative for obtaining satisfactory solutions.

algorithm	# ships without route				running time (CPU min)			
	avg.	std. dev.	min	max	avg.	std. dev.	min	max
SeqR	1.26	1.36	0	9	0.01	0.01	0.00	0.05
LS-240	1.10	1.30	0	8	7.97	3.76	0.21	27.80
RH-1	1.05	1.22	0	6	1.34	1.42	0.23	19.65
RH-3	1.05	1.22	0	7	3.56	2.77	0.15	34.40

Table 1 Performance of algorithmic variants, averaged over all instances: Number of ships for which no dynamic route could be found and the running time in CPU minutes.

Table 1 summarizes two further aspects: the number of ships for which no dynamic route could be found (a little more than one ship per instance) and computation times (which are generally very low). RH-1 yields the best combination of quality and running time.

6.2. Combinatorial Relaxation

In Section 3.1 we discussed the combinatorial relaxation that ignores siding constraints. We use its MIP formulation (1)–(8) to estimate the quality of solutions obtained by our rolling horizon heuristic RH-1. This MIP is computationally very demanding: a still unsolved instance (called `shipsched`) with 87 ships belongs to the *challenge* benchmark set of MIPLIB2010 (Koch et al., 2011). We therefore prepared smaller instances comparable to a single time horizon in RH-1 as follows. Start

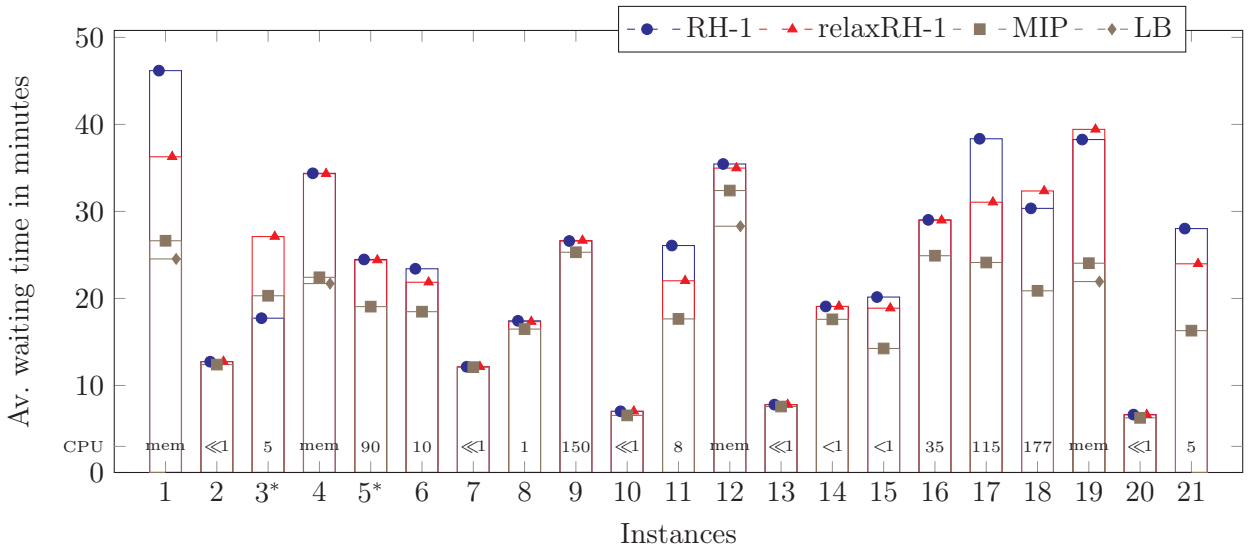


Figure 15 Results for 21 small instances with at most 35 ships over a 2h horizon. The columns indicate the respective average waiting times per instance for the rolling horizon heuristic (RH-1), this heuristic with relaxed siding conditions (relaxRH-1), the value of the best known integer solution of MIP (1)–(8), and (if not identical to MIP) the largest known lower bound (LB). MIP/LB constitute *lower* bounds on the optimum. For instances 3 and 5 RH-1 was not able to find a dynamic route for all ships of the corresponding instance due to capacity problems within sidings. The `CPLEX` computation time to solve the MIP is stated in CPU minutes, or "mem" when the computation hit the memory limit.

from a given solution for a whole day; filter all ships that are present or enter the canal during a given time horizon of two hours; and further reduce the number of ships to at most 35 by random removal, biased to prefer a balance between upstream and downstream ships and to keep ships satisfying a minimum distance value between start and target position.

Figure 15 shows results for 21 instances. Label "MIP" marks the value of the best known feasible integer solution of MIP (1)–(8). For instances not solved to optimality we additionally state the largest known lower bound indicated with the label "LB." The objective values labeled "RH-1" refer to the rolling horizon heuristic. A version of the rolling horizon heuristic where siding constraints are relaxed as in the MIP is labeled "relaxRH-1." The computational efforts of the MIP solver `CPLEX` (version 12.3) indicate that instances with 35 ships vary from very easily solvable to intractable.

The average difference of waiting times between LB/MIP and relaxRH-1 is about 5 minutes (maximum 18). These gaps increase a bit if the siding conditions must be respected (average below 7 minutes, maximum below 22; ignoring the two instances marked with $*$). Different decisions in the local search occasionally produces less waiting time for RH-1 than for relaxRH-1. For the larger 24-hour instances (Figure 16) RH-1 loses on average only 8% when compared to relaxRH-1. This increases drastically (to about 30%) when we consider the forecasted traffic demand for 2025 (not reported here). This shows that a detailed modeling of siding constraints is important.

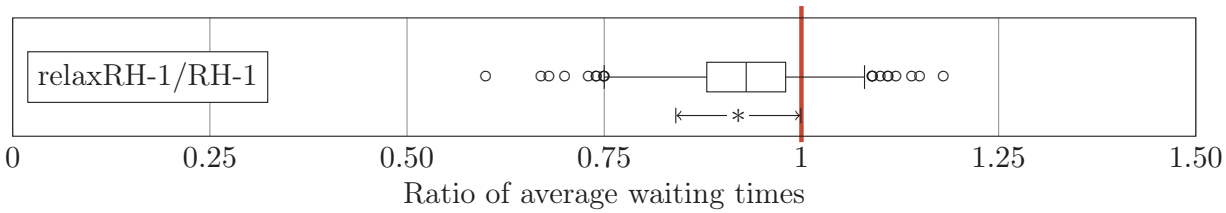
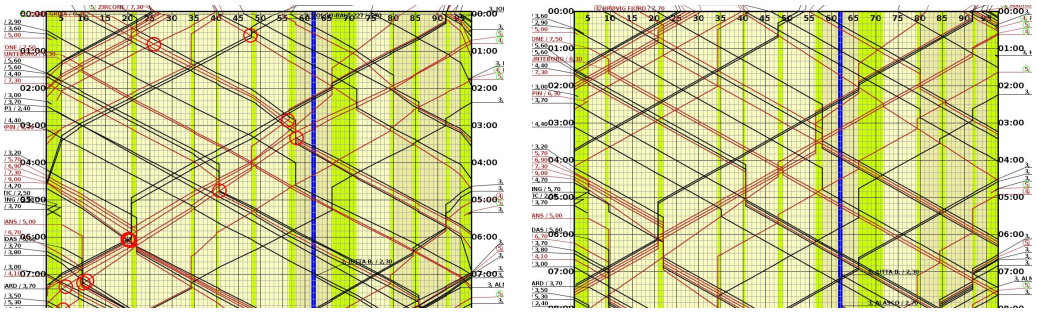


Figure 16 Comparison as in Figure 14 (all instances of 2007): our rolling horizon heuristic (RH-1) vs. its variant with relaxed siding constraints (relaxRH-1).

6.3. GPS Data Realized

We finally evaluate the practical usefulness of our algorithms. The GPS data contain actual entry/exit times at sidings for each ship. Figure 17(a) shows the linear interpolation between them. Small circles indicate infeasibilities of such a solution: large ships pass each other (slightly) outside sidings; velocities do not exactly match the actual ones (some ships exceed the speed limit in reality, others move below full speed, in particular when waiting in front of the lock chambers at the canal boundaries); also constraints in sidings may be violated.



(a) Linear interpolation of GPS data. (b) GPS data realized with Algorithm 1b.

Figure 17 Actually traveled itineraries as per GPS data with infeasibilities on the left (a) and a feasible solution derived using Algorithm 1b on the right (b).

We deduce a combinatorial structure from this solution, resorting to the most plausible scheduling decision in case of ambiguities, and use Algorithm 1 to construct a routing. The latter may not *exactly* realize the original solutions since these need not be earliest arrival ones in general. However, we consider this proceeding a closest-to-reality re-construction of actual itineraries which at the same time fit the precise feasibility definitions. The result is called "GPS data realized," GPS-re.

Figure 18 shows the improvement of our rolling horizon heuristic over manually planned solutions. Average waiting times are down by some 25% on average. Again, precise values should be taken with care, but a picture is clearly emerging. In Figure 19 we sort instances according to average waiting

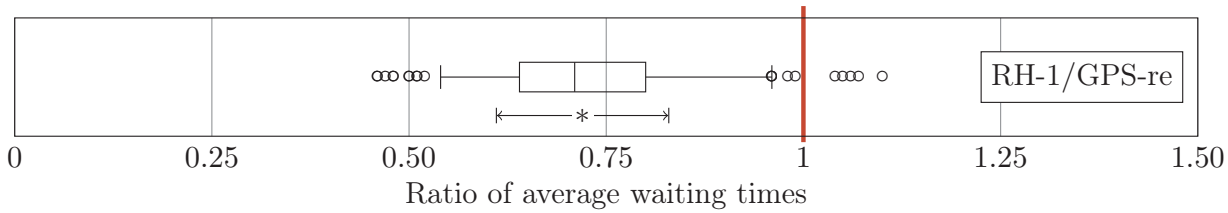


Figure 18 Distribution of the instance-wise improvement w.r.t. average waiting time of our rolling horizon heuristic (RH-1) over realized GPS data (GPS-re).

Figure 19 Average waiting time achieved by realizing the GPS data (GPS-re) and the rolling horizon heuristic (RH-1) for each day of 2007, sorted non-decreasingly according to GPS-re.

time of GPS-re which is considered as a measure of the *planning complexity*. Instances which are harder under this measure in reality are harder for our heuristic, too.

7. Discussion and Perspectives

We introduced the combinatorial optimization problem of ship traffic control at the Kiel Canal. To solve it, we integrated algorithmic ideas from train scheduling on a single-track network and collision-free routing of automated guided vehicles. Blending dynamic routing and scheduling, or sequential (local) and simultaneous (global) solution approaches, enabled us to allot scarce network resources to a fleet of vehicles in a collision-free manner.

We developed a fast heuristic, with an average running time of less than two minutes to plan all ships of one day. Our solutions are approved by the expert planners and significantly improve upon manual plans. This laid the ground for a computer aided traffic control. Also, our planning

in rolling horizons integrates well with a heuristic (Luy, 2010) that schedules the locking process at each boundary of the Kiel Canal since entering, passing, and exiting the canal are interdependent.

Having a computer aided scheduling tool is even more interesting as the number and size of vessels is projected to grow, as we mentioned in the introduction. This was actually the reason why we were approached by the Waterways and Shipping Administration: A variety of options of dredging and widening of the canal and moving or enlarging sidings was considered to meet future traffic demand. Our tool was used to assess the *operational impact* of each of these 128 enlargement scenarios (with projected traffic data for the year 2025) and the strategic decision was based on these computations. The enormous level of detail in our model ensured that the chosen construction scenario actually constitutes a remedy to the impending in-operability of the canal. Figure 19 suggests that our tool is able to reliably compare the planning complexity of two different enlargement scenarios.

Shih et al. (2014) were in a similar situation to take strategic decisions by evaluating the operational impact, when optimally locating sidings in single-line railway tracks.

The officials considered using our tool during several potentially difficult years of construction work and even to decide about the construction schedule itself: Different orders of construction and the selection of different excavating machines (several of which significantly hinder regular traffic) directly influence the traffic flow.

We are confident that our ideas can be transferred to other application areas like train scheduling and collision-free routing, and thus provide a prototype for scheduling bi-directional traffic, possibly in more general networks like in Jaumard et al. (2013). This may entail extensions like considering variable speed, e.g., with a speed discretization, and thus, parallel arcs on transit segments.

Acknowledgments. This work was done while all three authors were with the Institut für Mathematik at Technische Universität Berlin, Germany. We are grateful to our cooperation partner "Planungsgruppe für den Ausbau des NOK" of the Waterways and Shipping Administration, in particular to Martin Abratis, Ulrich Bösl, Martin Bröcker, and Sönke Meesenburg for their openness to mathematical methods, for their efforts and patience at an intense communication process, and in general for a very delightful cooperation. We thank Felix G. König for stimulating discussions about algorithmic ideas and intensive support within this project. We acknowledge support by the German Research Foundation (DFG) which enabled our production of a series of educational short movies for high school students, based on this project (Lübbecke and Höhn, 2009).

References

Baker, B.S., E.G. Coffman Jr., R.L. Rivest. 1980. Orthogonal packings in two dimensions. *SIAM Journal on Computing* **9**(4) 846–855. doi:10.1137/0209064.

Bierwirth, C., P. Corry. 2018. Integrating ship scheduling and berth allocation for container seaports with channel access. M. Freitag, H. Kotzab, J. Pannek, eds., *Dynamics in Logistics*. Lecture Notes in Logistics, Springer, 144–147. doi:10.1007/978-3-319-74225-0_18.

Błądek, I., M. Drozdowski, F. Guinand, X. Schepler. 2015. On contiguous and non-contiguous parallel task scheduling. *Journal of Scheduling* **18**(5) 487–495. doi:10.1007/s10951-015-0427-z.

Brockmann, J., A. Heeling, M. Pohl, K. Uliczka. 2008. The Kiel Canal. *Die Küste* **74 ICCE** 317–332. URL `https://hdl.handle.net/20.500.11970/101614`.

Campbell, J.F., L.D. Smith, D.C. Sweeney II, R. Mundy, R.M. Nauss. 2007. Decision tools for reducing congestion at locks on the Upper Mississippi River. *Proceedings of the 40th Hawaii International Conference on System Sciences*. 56–65. doi:10.1109/HICSS.2007.164.

Carey, M., D. Lockwood. 1995. A model, algorithms and strategy for train pathing. *Journal of the Operational Research Society* **46** 988–1005. doi:10.1057/jors.1995.136.

Carroll, J.L., M.S. Bronzini. 1973. Waterway transportation simulation models: Development and application. *Water Resources Research* **9**(1) 51–63. doi:10.1029/WR009i001p00051.

Ceder, A. 2007. *Public Transit Planning and Operation: Theory, Modelling and Practice*. Butterworth-Heinemann (Elsevier), Oxford.

D'Ariano, A., F. Corman, D. Pacciarelli, M. Pranzo. 2008. Reordering and local rerouting strategies to manage train traffic in real time. *Transportation Science* **42**(4) 405–419.

Desrochers, M., F. Soumis. 1988. A generalized permanent labeling algorithm for the shortest path problem with time windows. *INFOR* **26**(3) 191–212.

Desrosiers, J., Y. Dumas, M.M. Solomon, F. Soumis. 1995. Time constrained routing and scheduling. M.O. Ball, T.L. Magnanti, C.L. Monma, G.L. Nemhauser, eds., *Network Routing*, *Handbooks in Operations Research and Management Science*, vol. 8, chap. 2. Elsevier, 35–139.

Dijkstra, E.W. 1959. A note on two problems in connexion with graphs. *Numerische Mathematik* **1** 269–271.

Disser, Y., M. Klimm, E. Lübbecke. 2015. Scheduling bidirectional traffic on a path. M.M. Halldórsson, K. Iwama, N. Kobayashi, B. Speckmann, eds., *International Colloquium on Automata, Languages, and Programming*, *Lecture Notes in Computer Science*, vol. 9135. Springer, Berlin Heidelberg, 406–418.

Duin, C.W., E. Van Der Sluis. 2006. On the complexity of adjacent resource scheduling. *Journal of Scheduling* **9**(1) 49–62. doi:10.1007/s10951-006-5593-6.

Ford, L.R., D.R. Fulkerson. 1958. Constructing maximal dynamic flows from static flows. *Operations Research* **6**(3) 419–433.

Ford, L.R., D.R. Fulkerson. 1962. *Flows in networks*. Princeton University Press, Princeton.

Gawrilow, E., E. Köhler, R.H. Möhring, B. Stenzel. 2008. Dynamic routing of automated guided vehicles in real-time. H.-J. Krebs, W. Jäger, eds., *Mathematics: Key Technology for the Future*. Springer, Berlin, Heidelberg, 165–177. doi:10.1007/978-3-540-77203-3_12.

Griffiths, J.D. 1995. Queueing at the Suez Canal. *Journal of the Operational Research Society* **46**(11) 1299–1309. doi:10.1057/jors.1995.179.

Gucma, Stanislaw. 2016. Parameter optimization of sea waterway system dredged to the specified depth case of the modernized Świnoujście-Szczecin fairway. *Archives of Transport* **40**(4) 29–38. doi:10.5604/08669546.1225461.

Günther, E., F.G. König, N. Megow. 2014. Scheduling and packing malleable and parallel tasks with precedence constraints of bounded width. *Journal of Combinatorial Optimization* **27**(1) 164–181. doi:10.1007/s10878-012-9498-3.

Harbering, J., A. Ranade, M. Schmidt. 2015. Single track train scheduling. *Proceedings of the fifth Multidisciplinary International Scheduling Conference: Theory and Applications (MISTA)*.

Higgins, A., E. Kozan, L. Ferreira. 1997. Heuristic techniques for single line train scheduling. *Journal of Heuristics* **3**(1) 43–62. doi:10.1023/A:1009672832658.

Holm, H., P. Grundevik. 2015. Ship traffic scheduling in the Göta River. URL `https://www.sspa.se/port-and-logistics/ship-traffic-scheduling-gota-river`.

Jaumard, B., T. Hoa Le, H. Tian, A. Akgunduz, P. Finnie. 2013. An enhanced optimization model for scheduling freight trains. *Proceedings of the 2013 ASME/IEEE Joint Rail Conference*. V001T04A003. doi:10.1115/JRC2013-2465.

Koch, Th., T. Achterberg, E. Andersen, O. Bastert, T. Berthold, R.E. Bixby, E. Danna, G. Gamrath, A.M. Gleixner, S. Heinz, A. Lodi, H. Mittelmann, T. Ralphs, D. Salvagnin, D.E. Steffy, K. Wolter. 2011. MIPLIB 2010 – Mixed Integer Programming Library version 5. *Mathematical Programming Computation* **3**(2) 103–163.

Lalla-Ruiz, E., X. Shi, S. Voß. 2016. The waterway ship scheduling problem. *Transportation Research Part D* doi:10.1016/j.trd.2016.09.013. In press, available online.

Lamorgese, L., C. Mannino. 2015. An exact decomposition approach for the real-time train dispatching problem. *Operations Research* **63**(1) 48–64.

Lawler, E.L., J.K. Lenstra, A.H.G. Rinnooy Kan, D.B. Shmoys. 1993. Sequencing and scheduling: Algorithms and complexity. S.C. Graves, A.H.G. Rinnooy Kan, P.H. Zipkin, eds., *Handbooks in Operations Research and Management Science*, vol. 4. Elsevier, Amsterdam, The Netherlands, 445–522.

Lübbecke, E. 2015. On- and offline scheduling of bidirectional traffic. Ph.D. thesis, Institut für Mathematik, Technische Universität Berlin.

Lübbecke, M.E., W. Höhn. 2009. DFG Science TV: Discrete optimisers. `dfg-science-tv.de/en/projects/discrete-optimisers`.

Lusby, R.M., J. Larsen, M. Ehrgott, D. Ryan. 2011. Railway track allocation: models and methods. *OR Spectrum* **33**(4) 843–883. doi:10.1007/s00291-009-0189-0.

Luy, M. 2010. Algorithmen zum Scheduling von Schleusungsvorgängen am Beispiel des Nord-Ostsee-Kanals. Master's thesis, Institut für Mathematik, Technische Universität Berlin. Available at `https://www.diplom.de/document/228321`.

Mitchell, K.N., B.X. Wang, M. Khodakarami. 2013. Selection of dredging projects for maximizing waterway system performance. *Transportation Research Record* **2330** 39–46. doi:10.3141/2330-06.

Passchyn, W., S. Coene, D. Briskorn, J.L. Hurink, F.C.R. Spieksma, G. Vanden Berghe. 2016. The lockmaster's problem. *European Journal of Operational Research* **251**(2) 432–441. doi:10.1016/j.ejor.2015.12.007.

Petersen, E. R., A. J. Taylor. 1988. An optimal scheduling system for the Welland Canal. *Transportation Science* **22**(3) 173–185.

Potts, C.N., M.Y. Kovalyov. 2000. Scheduling with batching: A review. *European Journal on Operational Research* **120**(2) 228–249.

Rambau, J., C. Schwarz. 2010. How to avoid collisions in scheduling industrial robots? Preprint, Universität Bayreuth. URL `opus.ub.uni-bayreuth.de/volltexte/2010/735/`.

Righini, G. 2016. A network flow model of the Northern Italy waterway system. *EURO Journal on Transportation and Logistics* **5** 99–122. doi:10.1007/s13676-014-0068-y.

Schlechte, T. 2012. Railway track allocation: Models and algorithms. Ph.D. thesis, Institut für Mathematik, Technische Universität Berlin.

Schonfeld, P., C.-J. Ting. 1998. Optimization through simulation of waterway transportation investments. *Transportation Research Record* **1620** 11–16.

Schonfeld, P., S.-L. Wang. 2005. Scheduling interdependent waterway projects through simulation and genetic optimization. *Journal of Waterway, Port, Coastal, and Ocean Engineering* **131**(3) 89–97. doi:10.1061/(ASCE)0733-950X(2005)131:3(89).

Shih, M.-C., Y.-C. Lai, T. Dick, M.-H. Wu. 2014. Optimization of siding location for single-track lines. *Transportation Research Record* **2448** 71–79.

Skutella, M. 2009. An introduction to network flows over time. W. Cook, L. Lovász, J. Vygen, eds., *Research Trends in Combinatorial Optimization*. Springer, Berlin, Heidelberg, 451–482.

Skutella, M., W. Welz. 2011. Route planning for robot systems. B. Hu, K. Morasch, St. Pickl, M. Siegle, eds., *Operations Research Proceedings 2010*. Springer Berlin Heidelberg, 307–312. doi:10.1007/978-3-642-20009-0_49.

Solomon, M.M., J. Desrosiers. 1988. Survey paper—time window constrained routing and scheduling problems. *Transportation Science* **22**(1) 1–13.

Szpigel, B. 1973. Optimal train scheduling on a single track railway. M. Ross, ed., *Operational Research '72*. North-Holland, Amsterdam, 343–352.

Ulusçu, Ö.S., B. Özbaş, T. Altıok, İ. Or, T. Yılmaz. 2009. Transit vessel scheduling in the Strait of Istanbul. *Journal of Navigation* **62**(1) 59–77. doi:10.1017/S0373463308005092.

Verstichel, J., G. Vanden Berghe. 2009. A late acceptance algorithm for the lock scheduling problem. S. Voß, J. Pahl, S. Schwarze, eds., *Logistik Management*. Physica-Verlag HD, 457–478. doi:10.1007/978-3-7908-2362-2_23.