A Variable Neighborhood Search Approach for the Interdependent Lock Scheduling Problem[⋆]

Matthias Prandtstetter[⋆⋆], Ulrike Ritzinger, Peter Schmidt, and Mario Ruthmair

AIT Austrian Institute of Technology
Mobility Department – Dynamic Transportation Systems
Giefinggasse 2, 1210 Vienna, Austria
{matthias.prandtstetter|ulrike.ritzinger|mario.ruthmair}@ait.ac.at

Abstract. We investigate a so far not examined problem called the *Interdependent Lock Scheduling Problem*. A *Variable Neighborhood Search* approach is proposed for finding lock schedules along the Austrian part of the Danube River in order to minimize the overall ship travel times. In computational experiments the performance of our approach is assessed and compared to real-world ship trajectories. Notable improvements can be achieved. In addition, the number of (empty) lockages can be significantly reduced when taking them into account during optimization without loosing too much of quality in travel time optimization.

Keywords: Interdependent Lock Scheduling Problem, Variable Neighborhood Search

1 Introduction

Inland navigation can be seen as the most efficient means of transport with respect to ecological objectives [3]. In combination with transportation via trucks and trains it builds a strong overland transportation network. Considering today's transportation volume and emissions related to (overland) transport in Europe [7] a shift of transports from trucks towards trains and especially inland navigation within the next years is highly desired by the European Commission [6].

Among European inland waterways Rhine and Danube are two of the largest (and most important) ones. While the Rhine heads North-South, mainly connecting Switzerland, France, the West of Austria, Germany, and the Netherlands, to the North Sea, the Danube heads West-East, connecting the South of Germany, Austria, Slovakia, Hungary, Croatia, Serbia, Bulgaria, Romania, Moldova, and the Ukraine, to the Black Sea. In contrast to the Rhine, ships traveling on the

⋆ This work is partially funded by the Austrian Federal Ministry for Transport, Innovation and Technology (BMVIT) within the strategic programme I2VSplus under grant 835771 (imFluss).

⋆⋆ corresponding author

(Austrian part of the) Danube (approx. 350 km) have to pass nine watergates at power plants used for electricity production. Naturally, these watergates build a bottleneck in the transportation network as ships have to pass through them to overcome the height difference caused by retaining the river.

It can be seen in the action plan of the Danube Region Strategy [4, 5] that only 10%–20% of the potential volume is transported on the Danube. Therefore, a significant increase in the transport volume will lead to congestion around watergates [14].

In the Austrian funded project *imFluss*, the main goal is to revise the currently applied *first-come, first-serve* strategy for the scheduling process at watergates and investigate the increase of efficiency by applying alternative scheduling strategies. The slot management is based on an optimization approach which schedules ships at watergates such that the overall sum of travel times of the ships is minimized. Furthermore, it allows to reduce congestion at watergates by providing travel speed advises to the captains. Since ships must traverse multiple watergates in short distances at the Danube, the interdependence of the schedules at the watergates has to be considered as well.

Within this paper, we first give a detailed description of the problem domain and then present a Variable Neighborhood Search (VNS) framework which incorporates various Local Search operators. Computational results show the positive impact of the proposed method on the planning approach, and finally conclusions sum the work up.

2 Related Work

The *Interdependent Lock Scheduling Problem* (ILSP), as addressed here, is not yet defined in the (scientific) literature to the best of our knowledge. However, various related problem definitions and corresponding solution approaches can be found in the literature where the most important ones are outlined in this section. The simplest version of the *lock scheduling problem* (LSP) optimizes the traffic flow through a single watergate which consists of a single lock chamber. The objective is to find a schedule involving up- and downstream vessels which maximizes the traffic flow. In [2], a polynomial-time dynamic program is proposed to solve this LSP. An extended version of this basic problem is proposed by Verstichel and Vanden Berghe [13], where multiple parallel lock chambers with different sizes are available. In addition, they integrate a kind of packing problem which focuses on the (optimized) placement of ships inside the lock chamber. They minimize the waiting times of the ships at watergates as the primary objective, but also consider the minimization of the number of lockage operations as a secondary goal. Beside heuristics for creating initial solutions, metaheuristic approaches are applied for improving the so far found solutions. These metaheuristics include a variable neighborhood search (VNS), a multiple neighborhood search and a composite neighborhood search approach. In [12], Verstichel proposes an exact approach based on integer linear programming. However, this method has unpredictable (long) runtimes which led to a

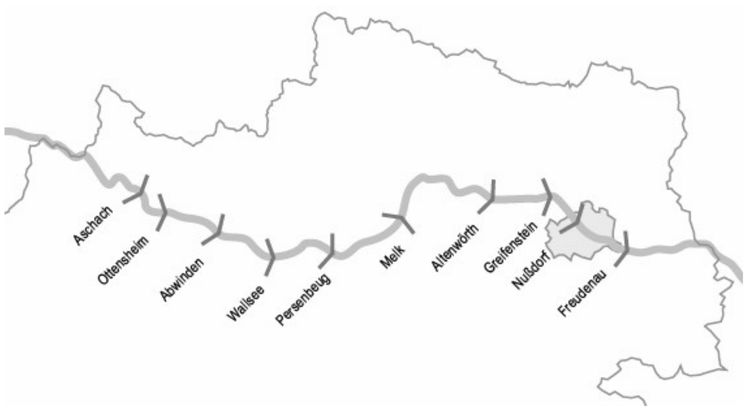

Fig. 1. Sketch of the Austrian part of the Danube with marks where watergates are located. Parts of the river between two watergates (or the state border) are referenced as sections. ©viadonau and DoRIS (`http://www.doris.bmvit.at`)

low acceptance by lock masters. While all of these approaches focus on single watergates only, there are some works which consider the ILSP at the Upper Mississippi River in USA [9]. However, the focus in this work is not combinatorial optimization but an estimation of delays based on traffic volume and interdependence among watergates. In [11] and [10], the authors investigate various strategies which are applied in vessel scheduling and report that a *shortest processing time first* strategy with fairness constraints is more efficient than the classical *first come, first serve* strategy.

3 Problem Description

In the *Interdependent Lock Scheduling Problem* (ILSP) we are given a set of m watergates $\mathcal{G} = \{1, \ldots, m\}$, successively arranged along the river. The river is divided into sections which are bounded by watergates or national borders, as it is depicted in Figure 1 for the Austrian part of the Danube. Let $\mathcal{S}$ be the set of ships which is partitioned into ships $\mathcal{S}^+$ going upstream and ships $\mathcal{S}^-$ going downstream, i.e., $\mathcal{S} = \mathcal{S}^+ \cup \mathcal{S}^-$, $\mathcal{S}^+ \cap \mathcal{S}^- = \emptyset$. Each ship $s \in \mathcal{S}$ has to pass through one or more successive watergates $\mathcal{G}_s \subseteq \mathcal{G}$, either up- or downstream.

Each gate $g \in \mathcal{G}$ consists of two identical and asynchronously operable lock chambers which allow the ships to overcome the water level difference when passing through a watergate. The length of a lock chamber is denoted by κ_g. A lockage operation is defined to either fill or empty the lock chamber with water. The time required for this operation is given by τ_g. Within a lock chamber ships are positioned in a row possibly allowing multiple ships to pass through a gate in a single lockage operation. The ships' positions in a lock chamber define their entering and exiting sequence.

The trip of ship $s \in \mathcal{S}$ starts at a given time and ends when it arrives at its target position. We assume that there are no planned stops for a ship. Additionally, information about the ship length k_s, the ship type r_s, and the average travel speed v_s is available. According to v_s and the positions of the watergates and borders, the travel time for each ship on each river section can be determined. Whenever a ship arrives at a watergate and takes part in a lockage operation it requires some time to enter and to exit the lock chamber, respectively.

In a feasible solution for the ILSP, each ship $s \in \mathcal{S}$ must be assigned to exactly one lock chamber position in exactly one lockage operation at each gate $g \in \mathcal{G}_s$, while respecting the following constraints:

– It must be ensured that only upstream going ships $s \in \mathcal{S}^+$ can be assigned to a lockage operation in which the lock chamber is filled up with water, whereas only downstream going ships $s' \in \mathcal{S}^-$ can be allocated to a lockage operation emptying a chamber.
– The sum of lengths of all ships assigned to the same lockage operation must not exceed the length of the lock chamber κ_g.

Additionally, for each lockage operation a starting time has to be set, based on the following limitations:

– A simultaneous operation of both lock chambers at a watergate is not possible. Therefore, previous lockage operations and the corresponding ships' exiting process have to be finished before the next ships can start their entering process. Only after all ships assigned to the same lockage operation have finished their entering process, the lockage operation is allowed to start.
– Ships are only allowed to enter (or exit) the lock chamber one after another because of safety and space limitations at the gates.
– All ships have to arrive at their target position before planning horizon H.

Because of these constraints, waiting times may arise for some of the ships. The optimization goal of the ILSP is to find a feasible solution such that the sum of total travel times of all ships is minimized, implicitly minimizing the sum of waiting times over all ships. As a second weighted term in the objective function, we optionally consider the minimization of the number of lockage operations since each lockage reduces the amount of water to be used in electricity production.

4 Solution Representation and Decoding

As mentioned in the previous section, a solution in the ILSP is defined by the lockage starting times and a unique assignment of each ship $s \in \mathcal{S}$ to a lock chamber position in a lockage operation for all watergates $\mathcal{G}_s$. Thus, a solution can be stored as a three-dimensional array where the first index refers to a watergate, the second one to a lockage operation at the watergate and the third one to the position of the ship in the lock chamber. The actual values stored in this three-dimensional array correspond to the ship ids. This array can be seen as a *relative schedule* without actual lockage starting times.

Based on this representation and the instance data we propose a decoding procedure to compute the corresponding optimal lockage starting times with respect to minimizing the sum of waiting times. Note that the number of lockage operations is implicitly given in the array. It can be easily seen that choosing the earliest possible starting time for each lockage results in a minimal sum of ship travel times with respect to the current solution. The earliest possible time for lockage l at watergate g depends on

- the previous lockage at gate g (if it exists),
- all lockages at neighboring gates which have at least one ship assigned which is also assigned to l and which has to pass the neighboring gate directly before g, and
- the start times of ships which start their trips in a neighboring section and are assigned to l.

Based on a *relative schedule*, a directed dependency graph can be built with nodes representing lockages and arcs describing a dependency of the lockage at the head on the lockage at the tail of the arc. In case of feasible *relative schedules* this graph has to be acyclic. Then, the starting times for all the lockage operations can be computed by traversing the dependency graph such that a node is only processed (i.e., a lockage starting time is set) if all preceding nodes have already been completed (i.e., starting times of previous lockages at the same or neighboring watergates have been fixed).

5 First Fit Construction Heuristic

In order to find an initial solution which can be further improved by the *Variable Neighborhood Search* framework described in the next section, we present a *First Fit* construction heuristic. It basically iterates over all watergate passages of ships in a chronological order and assigns them to lockages, see Alg. 1 for a pseudo-code of the approach.

In line 5 it is decided whether a vessel can be added to the (currently) last lockage operation at the corresponding watergate. This decision includes structural decisions based on the length of the lock chamber as well as the traveling direction of the ship. In addition, operational decisions are included such as the maximum shift in time caused by inclusion of a vessel to a lockage. In our implementation, additional lockages are added if a lockage will be postponed more than 30 minutes, which reflects the amount of time needed for filling (or emptying) a lock chamber.

6 Variable Neighborhood Search Framework

To improve initial solutions we employ a *Variable Neighborhood Search* (VNS) framework [8]. As a local search method within VNS we incorporate *Variable*

Algorithm 1: FirstFitConstruction

```
1  $Q \leftarrow$ chronologically ordered queue of watergate passing events
2  while $Q \neq \emptyset$ do
3      $e \leftarrow Q.pop()$
4      $g \leftarrow watergate(e)$
5      if $e$ can be assigned to the currently last scheduled lockage at $g$ then
6          assign $e$ to last lockage at $g$
7      else
8          append new lockage(s) at $g$
9          assign $e$ to now last lockage
10     compute $\forall e' \in Q$ dependent on $e$ new earliest possible lockage times
11     chronologically re-order $Q$
```

Neighborhood Descent (VND) [8]. The main idea of VNS and VND is to systematically examine different neighborhood structures such that local optima with respect to single neighborhood structures can be overcome.

6.1 Neighborhood Structures for VND

The neighborhood structures incorporated into our VND implementation are based on the definition of so-called move operators which are used for (slightly) modifying a given solution by small local adjustments. As it is possible to decode a given *relative schedule* such that lockage starting times are optimal with respect to the sum of travel times, no neighborhood structures are defined on varying lockage starting times, cf. Sec. 4. All defined moves work on the *relative schedule* only. In the following we describe the used neighborhood structures.

Shift Vessels The main idea of this neighborhood structure is to move one or more vessels from one lockage to another where the number of ships to be shifted is an input parameter. For implementation issues we decided to allow only multiple ships to be moved together if they are assigned to the same lockage. In general, a shift can only be performed if

– the corresponding watergate stays the same,
– no circular dependencies are introduced in the underlying dependency graph (cf. Sec. 4), and
– the ships to be shifted fit into the target lock chamber together with all currently assigned ships.

However, it is not only necessary to decide to which other lockage vessels are shifted but also at which actual position the ships will be inserted, i.e., the ordering of the ships in the target lock chamber has to be considered. Note, however, that the relative order of the shifted ships and the ships in the target lock chamber do not change.

Based on this definition, the size of this neighborhood can be estimated by a function linear in the number of lockages per watergate times the maximum number of vessels concurrently assigned to a lockage.

Swap Vessels As indicated by the name of this neighborhood structure, the main idea here is to swap the positions and/or lockages of two vessels. Basically, the same precondition applies as for shift moves. However, a swap of vessels scheduled for the same lockage is explicitly included in this move type.

The size of a corresponding neighborhood can therefore be estimated to be quadratic in the number of ships passing a watergate. This implies that on average neighborhoods based on this structure will be significantly larger than neighborhoods based on shifting operations.

Remove Empty Lockages Since a Shift Vessel move may result in empty lockages, it is beneficial to remove them with respect to minimizing the number of lockages. However, removing a single lockage is in most cases not feasible as each lock chamber has alternating upstream and downstream lockages. Thus, we try to remove two lockages at the same time. Therefore, the size of the neighborhood can be estimated by a function which is quadratic in the number of empty lockages.

6.2 Neighborhood Structures for VNS

Analogously to VND, several neighborhood structures are defined to be used within VNS for shaking operations. To keep things simple and fast, we decided to perform a pre-defined number of random shift moves during the shaking phase. Currently, four neighborhood structures are defined with neighborhood structure i performing i random shifts, with $i \in \{1, 2, 3, 4\}$.

7 Experiments and Computational Results

In order to test the performance of the proposed VNS framework and the applied neighborhood structures we conducted a set of computational experiments. For this purpose, we decided to generate a set of test instances which were extracted from real-world data provided by our project partner viadonau who is providing the Austrian River Information Services called DoRIS. Via DoRIS we got anonymized sample data including the trajectories of all vessels for selected days in the period from August 2013 until April 2014. We selected uniformly 30 days in this period. Since the selected period covers days with both low and high traffic demand a broad range of situations and especially number of trips (from 47 up to 151, cf. Tab. 1) has been investigated. Note that there are days with even more trips along the Austrian Danube. We removed, however, all trips which did not pass at least one watergate as those trips have no particular influence on the ILSP. A real-world scenario was simulated by using the start and

Table 1. Results when only minimizing the total travel time (VNS0) and when additionally minimizing the number of lockages with weight 1000 (VNS1000). Column *hist vs. VNS0* depicts the average relative performance in percent of VNS0 with respect to historical data. Columns *VNS0 vs. VNS1000* show the changes in the relevant key performance indicators (total travel time, number of empty lockages, algorithm runtime) for VNS1000 compared to VNS0.

		hist. vs. VNS0	VNS0 vs. VNS1000			
instance	#trips	travel time [%]	travel time [%]	#lockages [%]	#empty [%]	runtime [%]
123_2014-01-07	56	-13.50	-0.01	+0.00	+0.00	+3.09
145_2014-02-03	65	-18.21	+0.05	+0.00	-0.44	+41.42
125_2014-01-12	67	-12.68	+0.00	-1.14	-3.75	+5.78
194_2014-02-04	69	-16.21	-0.00	-0.09	-0.62	+12.11
203_2014-03-06	71	-12.70	-0.08	-1.29	-4.81	+11.80
167_2013-12-08	78	-9.38	+0.01	-0.96	-4.00	-11.08
179_2013-12-02	82	-14.02	+0.19	+0.00	+0.05	+15.44
182_2014-01-10	83	-13.49	-0.03	-2.21	-9.55	+2.28
221_2013-11-07	84	-13.01	-0.13	-5.87	-20.54	+11.42
203_2013-09-05	87	-8.73	+0.04	-2.77	-12.63	+32.44
197_2014-03-07	90	-5.87	+0.07	-2.27	-8.23	+1.50
188_2014-02-07	94	-5.55	+0.18	-1.51	-11.32	-17.47
180_2013-12-07	98	-9.82	+0.07	+0.11	+0.26	-9.80
208_2014-01-09	99	-7.45	+0.06	-5.07	-20.29	+11.59
254_2013-10-07	102	-4.17	+0.02	-1.81	-6.61	+14.94
184_2013-11-09	104	-8.50	-0.02	-3.19	-13.26	-1.03
118_2013-08-12	107	-5.62	+0.09	-2.82	-11.73	-9.56
234_2013-09-02	108	-5.82	-0.09	-4.14	-15.16	+11.74
210_2013-10-13	111	-0.08	+0.03	-3.04	-12.30	+12.01
258_2013-12-06	115	-11.58	+0.01	-2.19	-13.08	-9.59
134_2013-08-05	119	-12.14	-0.18	-5.55	-26.40	+27.01
138_2013-08-06	122	-3.33	-0.09	-2.24	-7.52	+12.00
250_2013-10-12	124	-0.81	+0.12	-3.33	-11.60	+21.73
268_2013-10-09	127	+1.89	-0.05	-2.53	-10.26	+7.07
299_2013-09-04	128	-6.19	+0.21	-1.11	-4.50	-13.93
289_2013-12-05	130	-5.10	+0.18	-0.40	-5.16	+42.45
156_2013-08-08	138	-9.26	+0.32	-7.64	-34.21	-13.32
283_2013-10-11	141	+7.56	-0.31	-4.93	-20.05	+18.94
332_2013-09-07	149	+0.47	-0.16	-1.63	-5.82	-6.84
358_2013-09-06	156	+8.52	+0.36	-3.00	-14.56	-9.31

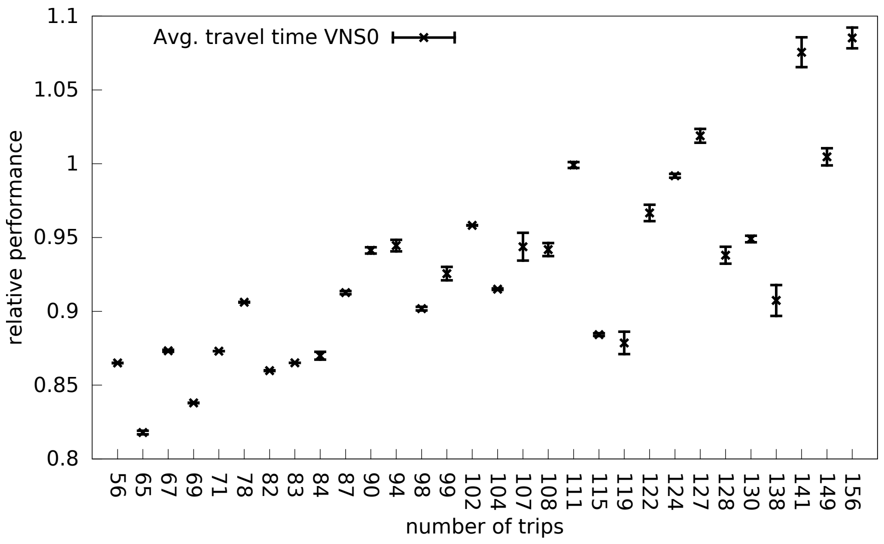

Fig. 2. Average total travel times (with standard deviations) over 30 runs of VNS0 compared to historical data.

target positions as well as departure times of ships as input. Travel times were estimated via the method proposed in [1]—an approach which turned out to be highly accurate. Finally, we compared historic travel times with the output of our algorithm. All of our tests were performed on a single core of a Intel Xeon 2600 processor with 4GB memory per core (although the average RAM consumption was much below that threshold). In addition, 30 runs were performed for each test instance and algorithmic setup.

The actual ordering of neighborhood structures in the VND part of the optimization process was chosen such that first a shift of one vessel is examined, followed by a swap of two ships, a shift of two vessels and a shift of three vessels. Finally, we apply the Remove Empty Lockages neighborhood search. A next improvement step function was employed for all neighborhood structures.

For the historic data provided via DoRIS we currently only have information of the ship trajectories but not on the actual lock operations. Therefore, we decided to compare in a first step the measured total travel time of all ships, with the estimated total travel time when applying a lock schedule as proposed by the VNS framework. As the number of lockages for the historic data is not yet known, we conducted two independent test series where in the first, the number of lockages was not taken into account (in the further context referred to as VNS0) while for the second one additional lockage is worth 1000 extra seconds travel time, i.e., we have chosen a weighting ration of 1:1000 between travel time and number of lockages (referred to as VNS1000 in the following). In Table 1 the obtained results can be seen in detail.

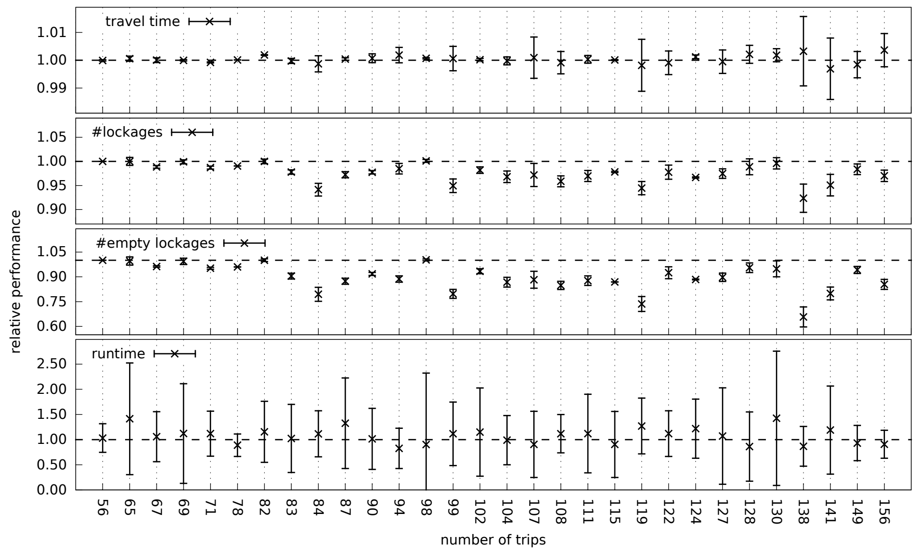

Fig. 3. Relative comparison between VNS0 and VNS1000. The baseline (dashed line) represents the performance of VNS0. The average performance of VNS1000 in relation to VNS0 is plotted together with the standard deviations.

A graphical representation is shown in Figures 2 and 3. While in Figure 2 the relative improvement of the VNS setups with/without respecting the number of lockages is given, Figure 3 shows the relative performance of VNS1000 with respect to VNS0. The areas around the averages represent standard deviations.

An improvement with respect to the historic trajectories can be found in almost all cases. However, for some instances the historic solution could not be beaten. This can be explained by two simple reasons: first, although the method for estimating travel times is highly accurate, it might still happen that travel times are underestimated. Second, the estimation how many ships can be packed into one lock chamber is highly heuristic and there are situations where the lockmaster was highly efficient while our heuristic approach decided that putting two specific ships at the same time in the lock chamber is not feasible. However, more important is the observation that VNS1000 results—although having a more complex objective function—in slightly better results with respect to travel times.

In addition, it can be seen that VNS1000 results in noticeable less empty lockages which is highly important: Although the objective is to maximize the traffic flow (i.e. to minimize the travel times) the additional objective of minimizing the number of lockages is justified by the fact that the embankment dams (in Austria) were erected for energy producing reasons. While lockages with ships can be argued towards an energy supplier, empty lockages are harder to be explained. Therefore, reducing the number of empty lockages while still decreasing the overall travel times is highly welcomed. Although the runtimes of VNS1000 are higher the computation times start at 20 seconds for smaller instances and range up 5000 seconds for the largest instances.

The relative improvement rates for shifts of single ships, swaps, shift of two vessels, shifts of three vessels and removing empty lockages are 82%, 37%, 5%, 0.02% and 36%, respectively.

8 Conclusions and Future Research

Within this work, we introduced the *Interdependent Lock Scheduling Problem* (ILSP) and provided a *Variable Neighborhood Search* (VNS) based approach for solving the ILSP heuristically. While different approaches exist for simpler versions of lock scheduling problems, the ILSP aims at finding an optimal lock schedule for multiple watergates along a river like the Danube. The ILSP is—to our best knowledge – not yet addressed in the academic literature. Based on real-world data provided by the Austrian waterway administration, we were able to show that an optimization approach helps in finding lock schedules resulting in reduced overall ship travel times. Furthermore, the number of (especially empty) lockages could be reduced without increasing the runtimes considerably. Since the ILSP is a highly complex problem, further research will include additional aspects such as considering fairness according to ship waiting times or considering a 2D-bin packing for ship placement in lock chambers.

Acknowledgements. We want to thank our project partners viadonau – Österreichische Wasserstraßen-Gesellschaft mbH and Zentralanstalt für Meteorologie und Geodynamik for providing us valuable insights into meteorological as well as nautical processes and challenges related to them as well as providing us relevant data.

References

1. Asamer, J., Prandtstetter, M.: Estimating ship travel times on inland waterways. In: TRB (ed.) TRB Annual Meeting Compendium of Papers. 14-3020 (2014)
2. Coene, S., Spieksma, F.C.R.: The Lockmaster's problem. In: Caprara, A., Kontogiannis, S. (eds.) 11th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems. OpenAccess Series in Informatics (OASIcs), vol. 20, pp. 27–37. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany (2011)
3. Dolinsek, M., Hartl, S., Hartl, T., Hintergräber, B., Hofbauer, V., Hrusovsky, M., Maierbrugger, G., Matzner, B., Putz, L.M., Sattler, M., Schweighofer, J., Seemann, L., Simoner, M., Slavicek, D.: Handbuch der Donauschifffahrt. bmvit (2013)
4. European Commission: Action plan accompanying the european union strategy for the danube region (2010), `http://www.danube-region.eu/component/edocman/action-plan-eusdr-pdf`
5. European Commission: European union strategy for danube region (2010), `http://www.danube-region.eu/component/edocman/communication-of-the-commission-eusdr-pdf`

6. European Commission: Roadmap to a single european traffic transport area – towards a competitive and resource efficient transport system (2011), `http://ec.europa.eu/transport/strategies/doc/2011_white_paper/white_paper_com(2011)_144_en.pdf`
7. European Commission: EU transport in figures: statistical pocketbook 2013. Publications Office of the European Union (2013)
8. Hansen, P., Mladenović, N.: Variable neighborhood search: Principles and applications. European Journal of Operational Research 130(3), 449–467 (2001)
9. Martinelli, D., Schonfeld, P.: Approximating delays at interdependent locks. Journal of Waterway, Port, Coastal, and Ocean Engineering 121(6), 300–307 (1995)
10. Ting, C.J., Schonfeld, P.: Efficiency versus fairness in priority control: Waterway lock case. Journal of Waterway, Port, Coastal, and Ocean Engineering 127(2), 82–88 (2001)
11. Ting, C.J., Schonfeld, P.: Effects of speed control on tow travel costs. Journal of Waterway, Port, Coastal, and Ocean Engineering 125(4), 203–206 (1999)
12. Verstichel, J.: The lock scheduling problem. Ph.D. thesis, KU Leuven Faculty of Engineering Science (2013)
13. Verstichel, J., Vanden Berghe, G.: A late acceptance algorithm for the lock scheduling problem. In: Voß, S., Pahl, J., Schwarze, S. (eds.) Logistik Management, pp. 457–478. Physica-Verlag HD (2009)
14. viadonau: Locked-through vessel units (2014), `http://www.donauschifffahrt.info/en/facts_figures/statistics/locked_through_vessel_units/`