Routing of barge container ships by mixed-integer programming heuristics

Vladislav Maraša, Jasmina Lazićd, Tatjana Davidovićb,∗, Nenad Mladenovićb,c

a University of Belgrade, Faculty of Transport and Traffic Engineering, Vojvode Stepe 305, Belgrade, Serbia
b Mathematical Institute of the Serbian Academy of Sciences and Arts, P.O. Box 367, Belgrade, Serbia
c Department of Mathematics, Brunel University, London, UK
d MathWorks Inc, Matrix House, Cambridge Business Park, CB4 0HH, Cambridge, UK

A R T I C L E   I N F O

Article history:
Received 1 December 2011
Received in revised form 26 October 2012
Accepted 12 March 2013
Available online 21 April 2013

MSC:
90B06
90B35
90C11
90C90

Keywords:
Routing
Barge container ships
Empty containers
0–1 mixed integer programming
Meta-heuristics

A B S T R A C T

We investigate the optimization of transport routes of barge container ships with the objective to maximize the profit of a shipping company. This problem consists of determining the upstream and downstream calling sequence and the number of loaded and empty containers transported between any two ports. We present a mixed integer linear programming (MILP) formulation for this problem. The problem is tackled by the commercial CPLEX MIP solver and improved variants of the existing MIP heuristics: Local Branching, Variable Neighborhood Branching and Variable Neighborhood Decomposition Search. It appears that our implementation of Variable Neighborhood Branching outperforms CPLEX MIP solver both regarding the solution quality and the computational time. All other studied heuristics provide results competitive with CPLEX MIP solver within a significantly shorter amount of time. Moreover, we present a detailed case study transportation analysis which illustrates how the proposed approach can be used by managers of barge shipping companies to make appropriate decisions and solve real life problems.

© 2013 Elsevier B.V. All rights reserved.

1. Introduction

This paper addresses the issue of liner barge container transport routes, connected to maritime container services. More precisely, we investigate the hinterland barge transport of containers arrived to or departing from a transshipment port (the sea port located at river mouth) on a sea or mainline container ships. One of the most important parts of this problem is adjusting the barge and sea container ships arrival times in the transshipment port. This clearly indicates that considered container barge transport acts as a typical feeder service.

Barge transport has recently proved to be a well-developed mode for container transport, particularly in Northwest Europe. Characteristics of barge container transport that have increased the importance of this transport mode are reliability of barge container services and low cost of barge transport operations. Therefore, container-on-barge transport is becoming even more competitive than its alternative mode, i.e. road transport for specific kind of transport activities. Following these trends, significant efforts have also been done in promoting barge transport of containers on the Danube and region of Southeast Europe. Over the last decade, container-on-barge transport has shown annual growth figures of 10–15% [1].

On the other hand, service network design in barge container transport has not been investigated enough. To the best of our knowledge, there are only a few researches dealing with these issues [1–4], even though there are many service network design problems in barge container transport that need further research attention. Therefore, this paper is intended to contribute to the successful design and development of service networks in container-on-barge transport, as this transport mode is expected to gain much more interest in years to come.

Determining transport routes of barge container ships has recently received a lot of attention (see, for example [4–7]). For these problems optimality may be defined in accordance with various factors: maximization of shipping company profit [8–11], minimization of system costs [12,13], minimization of operating costs [14], etc. Obtaining an optimal solution by any factor is very important for doing successful transport business. Unfortunately, like in many other practical cases, the complexity of real life prob-
lems exceeds the capacity of the present computation systems. Therefore, the efficient algorithms for finding good suboptimal solutions are required. Meta-heuristic methods are the most natural choice, as they were already widely used in recent literature [11,15,16]. In this paper we consider the barge container ship routing problem with the aim of maximizing the shipping company profit while picking up and delivering containers (both loaded and empty) along the inland waterway. The problem has been studied for the first time in [4,16]. Here, the mixed integer programming formulation for this problem is presented and used within CPLEX MIP solver to obtain optimal solutions of small size instances. For larger instances, MIP heuristic methods are used to obtain good suboptimal solutions. We apply three state-of-the-art heuristics for 0–1 MIP problem: Local Branching (LB) [17], Variable Neighborhood Branching (VNB) [18] and Variable Neighborhood Decomposition Search for 0–1 MIP problems (VNDS-MIP) [19]. In addition, we propose an extension of the original VNDS-MIP method, called LS-VND. In which considers not only binary but also general integer variables. Detailed experimental results confirm the superiority of heuristic methods over the commercial exact solver. At the end, for a small representative test instance we present detailed transportation analysis describing obtained optimal route and container flows along it.

The rest of this paper is organized as follows. In the next section we present a brief literature review. The description of the considered problem: optimization of transport routes of barge container ships is given in Section 3. Intuitive description as well as mathematical formulation is given and the problem complexity is discussed. Section 4 contains an overview of the MIP-based heuristic methods that we used for this problem (to obtain good suboptimal solutions in reasonable computation time). Experimental evaluations are described in Section 5, while Section 6 contains concluding remarks.

2. Literature review

Planning, routing and scheduling of container ships in both maritime and hinterland transport was significantly studied in relevant literature. We briefly describe here some of the most recent advances in this area. The relation between barge network design, transport market and the performance of intermodal barge transport was studied in [20]. A conceptual model for barge network design which describes the design variables for barge networks and their relation to the performance indicators of intermodal barge transport from a shippers’ and operators’ perspective was presented. It was explained that the vessel size and the circulation time of vessels are major factors for this issue and directly influence the cost and quality performance of barge transport. Similarly, Konings [6] investigated whether a hub-and-spoke service could be a fruitful tool to improve the performance of the container-on-barge transport and hence to gain market share. He indicated that a hub-and-spoke network can produce efficient barge services, by increasing both the productivity of a vessel through optimized sailing schedules and the efficiency of capacity utilization (depending on waterway dimensions). In addition, future requirements and opportunities of barge terminals to further improve the competitiveness of container barge transport were explored in [21]. Notteboom [22] assessed the trade-offs linked to the time factor in liner service schedules from the perspective of a shipping line. The author also discussed the range of measures and planning tools container carriers can deploy to maximize schedule reliability. The main conclusions of his paper were: (1) carriers’ strategies in dealing with potential disruptions in service schedules differ substantially, which identifies a service variable that adds to market segmentation in liner shipping; (2) port congestion is the main source of schedule unreliability; (3) future research scopes have to include different trade routes (not just East Asia – Northern Europe lines), inter-round trip effects, and network effects.

In [23] similarities and dissimilarities between the spatial and the functional development of the container river service networks of the Yangtze and the Rhine River were discussed. It was shown that the Yangtze service network has the tendency to converge, in more than one aspect, with the historical development pattern of inland container services in the Rhine basin.

In [11] the designing service networks for container liner shipping in the sea container industry, while explicitly taking into account empty container repositioning was addressed. The problem was modeled as a Knapsack problem that was reduced to a location routing problem. A Genetic Algorithms (GA) based heuristic was proposed to find a set of calling ports, an associated port calling sequence, the number of ships (by ship size category) and the resulting cruising speed to be deployed in the service networks, with the objective of profit maximization for a liner shipping company. An application to the problem of container transportation in Southeast Asia was presented.

Network flow techniques were employed to construct a model for short-term ship scheduling and container shipment from the carrier’s perspective in [12]. A network flow technique was applied to construct the model, which included multiple ship-flow and container-flow networks. The authors developed a solution algorithm based on Lagrangian relaxation, a subgradient method, and a heuristic for the upper-bound solution. On the other hand, the authors of [24] proposed a novel mixed integer linear programming mathematical model for the liner shipping network design problem in a competitive environment. This model addressed the competition between a newcomer liner service provider and an existing dominating operator, both operating on hub-and-spoke networks.

Although there is a substantial amount of articles related to ship routing and scheduling, the work on the allocation of empty containers in the barge container transport has been scarce. On the other hand, for the sea transport, empty container repositioning problems have been studied in various ways: as determinis‐

3. Formulation of the problem

The problem considered in this paper consists of finding the route for a given barge container ship as to maximize the profit of the shipping company. An example of inland waterway is presented in Fig. 1.

Port 1 is the sea or hub port, while ports 2 to n represent river ports. Index associated to each port depends on its distance from port 1. The arrows indicate streams for the sequence of calling ports. The demanded container traffic between each pair of ports \((i, j), i, j = 1, 2, \ldots, n, i \neq j\) is specified. The solution of this problem defines upstream and downstream calling sequence and number of loaded and empty containers transported between any two ports
while achieving maximum profit of the shipping company. The first port (a sea port, located at a river mouth) and the last port (the furthest port upstream) are always included in a solution, while the remaining \( n - 2 \) ports in either direction (upstream or downstream) may or may not appear in the optimal (and even any feasible) solution.

As it is not realistic to suppose that capacity of barge container ship ensures the satisfaction of all customer demands, container traffic between ports has a highly significant role. More precisely, the objective is to determine the number of containers (both loaded and empty) to be transferred between any two calling ports while achieving maximum profit of the shipping company.

### 3.1. Problem overview

Routing is a fairly common problem in transport, but the barge container transport problem addressed here has certain intrinsic features that make the design of transport routes and corresponding models particularly difficult. The following assumptions impose the restrictions on the routing:

- the barge shipping company (or charterer) wants to hire a ship or a tow for a period of one or several years (‘period time charter’) in order to establish container service on a certain inland waterway;
- the ship follow the same route during a pre-specified planning time horizon; as common assumption, this horizon may assumed to be one year long;
- the trade route is characterized by one sea or hub port located at a river mouth and several intermediate calling river ports;
- the model assumes a weekly known cargo demand for all port pairs; this assumption is valid as the data regarding throughput from previous periods and future prediction allow obtaining reliable values of these demands;
- a barge container ship route corresponds to a feeder container service; the starting and ending point on the route should be the same, i.e. in this case it is the sea port where transshipment of containers from barge to sea container ships and vice versa takes place;
- a barge container ship travels upstream from the starting sea port to the final port located on inland waterway, where the ship sails from in the downstream direction to the same sea port ending the route;
- maximum allowed route time, including sailing time and service time in ports, has to be set in accordance with the schedule of the mainline sea container ship calling at the transshipment port;
- it is not necessary for the barge container ship to visit all ports on the inland waterway; in some cases, calling at a particular port or loading all containers available at that port may not be profitable;
- the ship does not have to visit the same ports in upstream and downstream directions;
- all the container traffic emanating from a port may not be selected for transport even if that port is included in the route;
- container service is organized as liner and accordingly liner terms are valid; this imposes that a barge shipping company has to deal with transshipment costs, port dues and empty container repositioning costs, in addition to the cost of container transport;
- the demand for empty containers at a port is the difference between the total traffic originating from the port and the total loaded container traffic arriving at the port for the specified time period; the assumption is valid since this study addresses the problem of determining the optimal route of a barge container ship for only one ship operator (similar to the case studied by [11]);
- empty container transport [27,33,37] does not incur additional costs as it is performed using the excess capacity of barge company ships (this transport actually incurs some costs, but its value is negligible in comparison with empty container handling, storage and leasing costs);
- if a sufficient container quantity is not available at a port, the shortage is made up by leasing containers with the assumption that there are enough containers to be leased (for details see [11]).

The objective when designing the transport route of a barge container ship is to maximize shipping company profit, i.e. the difference between the revenue arising from the service of loaded containers (\( R \)) and the transport costs. These costs are related to shipping (TC) as well as empty container handling (EC). Therefore, the objective function has the form (see [11]):

\[
Y = R - TC - EC. \tag{1}
\]

The model is generated with the assumption that one ship or tow is performing all weekly realized transport. An extension to a multi-ship problem is straightforward. It enables a better satisfaction of customer demands but requires modification of the one-ship variant. On the other hand, if a shipping company has an option to charter one among several various ships at disposal, then each ship can be evaluated separately using this model to determine the best option.

The total number of ships employed on a route is determined as the ratio of barge container ship round-trip time and service frequency of mainline sea container ships (one week in our case). For example, if the round-trip time is 20 days, the company should charter 3 ships to be able to satisfy weekly demands of the customers. To describe all components of the relation (1) and precise the calculation procedure for the shipping company profit we present a mixed-integer linear programming formulation of the considered problem.

### 3.2. Mathematical formulation

In this section the barge container ship route is formulated as the mixed-integer linear program (MILP) with an aim to maximize the shipping company profit. The problem is characterized by the following input data (measurement units are given in square brackets if applicable):

- \( n \): number of ports on the inland waterway, including the sea port;  
- \( v_s \) and \( v_u \): upstream and downstream barge container ship speed, respectively [km/h];  
- \( scf \) and \( scel \): specific fuel and lubricant consumption, respectively [t/kWh];  
- \( fs \) and \( flp \): fuel and lubricant price, respectively [US$/t];  
- \( P_m \): engine output (propulsion) [kW];  
- \( dcc \): daily time charter cost of barge container ship [US$/day];
The optimal route of the barge container ship may be identified by solving the following mathematical model (linear program). Decision variables of the model are:

- binary variables \( x_{ij} \) defined as follows:
  \[
  x_{ij} = \begin{cases} 
  1 & \text{if ports } i \text{ and } j \text{ are directly connected in the route}, \\
  0 & \text{otherwise}; 
  \end{cases}
  \]

- integer variables \( z_{ij} \) and \( w_{ij} \), representing the number of loaded and empty containers, respectively, transported from port \( i \) to port \( j \) [TEU].

The mathematical programming formulation is as follows

\[
\begin{align*}
\text{max} & \quad Y \\
\text{s.t.} & \quad \sum_{q=1}^{n} \sum_{s=1}^{j} (z_{qs} + w_{qs}) \leq C + M(1 - x_{ij}), \\
& \quad i = 1, 2, \ldots, n - 1; \quad j = i + 1, \ldots, n \\
& \quad \sum_{q=1}^{n} \sum_{s=1}^{j} (z_{qs} + w_{qs}) \leq C + M(1 - x_{ij}), \\
& \quad i = 1, 2, \ldots, n - 1; \quad j = i + 1, \ldots, n \\
\end{align*}
\]

\[
\begin{align*}
\sum_{q=1}^{n} \sum_{s=1}^{j} (z_{qs} + w_{qs}) & \leq C + M(1 - x_{ij}), \\
& \quad i = 2, \ldots, n; \quad j = 1, \ldots, i - 1 \\
& \quad \sum_{q=1}^{n} \sum_{s=1}^{j} (z_{qs} + w_{qs}) \leq C + M(1 - x_{ij}), \\
& \quad i = 2, \ldots, n; \quad j = 1, \ldots, i - 1 \\
\end{align*}
\]

where \( M \) represents large enough constant.

Constraints (3)–(6) model the departure (3) and (4)) and arrival (5) and (6)) of ship and containers to and from each port on the route, respectively, in both upstream and downstream direction. More precisely, these constraints specify that loaded containers could be transported between ports \( i \) and \( j \) only if both ports are included in the route. Moreover, it is not possible to transport more containers than the weekly expected number (given as the input data). Capacity constraints (7) and (8), guarantee that the total number of loaded and empty containers on-board will not exceed the ship carrying capacity at any voyage segment. The constant \( N \) is introduced to make the constraints valid for the ports that are not included in the route. Constraints (9) and (10) are the set of network constraints ensuring that the ship visits the ending ports of the route. The second set of network constraints (11) and (12) guarantee that the route represents a connected liner trip. The barge container ship is left with a choice of calling or not calling at any inner port.

Constraint (13) is related to the round-trip time of the barge container ship (denoted by \( t_{tot} \) [h]). The role of this constraint is to prevent the ship ending and calling at port 1 long before or after the arrival of the sea ship in this port. The round-trip time can be calculated as the sum of total voyage time, handling time of loaded and empty containers in ports and time required for entering and leaving ports (14).

\[
t_{tot} = \left( \frac{L_{1}}{v_{1}} + \frac{L_{2}}{v_{2}} + t_{1} + t_{k} \right) + \sum_{i=1}^{n} \sum_{j=1}^{n} (z_{ij}(f_{ij} + u_{ij})) \\
+ w_{ij}(l_{ij} + u_{ij}) + x_{ij}(p_{ij} + p_{ai}) \\
\]

According to Eq. (1) and given input data, the profit value \( Y \) is calculated as follows:

\[
Y = \sum_{i=1}^{n} \sum_{j=1}^{n} \left( \frac{d_{ij} \cdot \max}{n} + P_{out} \left( \frac{l_{ij}}{v_{1}} + \frac{l_{ij}}{v_{2}} \right) \right) (f_{ij} \cdot scf_{i} + lp \cdot slc_{i}) \\
+ \sum_{i=1}^{n} \sum_{j=1}^{n} x_{ij} \cdot pec_{ij} + \sum_{i=1}^{n} \sum_{j=1}^{n} z_{ij}(u_{ij} \cdot f_{ij} + l_{ij}) \\
- \sum_{i=1}^{n} \left( \frac{d_{ij} \cdot s_{wi} + x_{ij} \cdot l_{wi} + \sum_{j=1}^{n} w_{ij}(uec_{ij} + lec_{ij}) \right) \]

\[
\sum_{j=2}^{n} x_{ij} = 1 \\
\sum_{i=2}^{n} x_{i1} = 1 \\
\sum_{i=1}^{q-1} x_{ij} - \sum_{j=q+1}^{n} x_{ij} = 0, \quad q = 2, \ldots, n - 1 \\
\sum_{i=q+1}^{n} x_{ij} - \sum_{j=1}^{q-1} x_{ij} = 0, \quad q = 2, \ldots, n - 1 \\
\min \frac{t_{tot}}{24} \leq \max \frac{t_{tot}}{tt}
\]
The number of containers to be stored at each port $i$, $s_{Wi}$, and the number of containers to be leased at each port $i$, $l_{Wi}$, can be defined as in [11]. After the linearization we obtain constraints (16)–(23):

$$
S_i - M g_i \leq 0 \quad (16)
$$

$$
D_i - P_i + S_i \geq 0 \quad (17)
$$

$$
D_i - P_i + S_i - M(1 - g_i) \leq 0 \quad (18)
$$

$$
Q_i - M h_i \leq 0 \quad (19)
$$

$$
P_i - D_i + Q_i \geq 0 \quad (20)
$$

$$
P_i - D_i + Q_i - M(1 - h_i) \leq 0 \quad (21)
$$

$$
l_{Wi} = Q_i - \sum_{j=1}^{n} w_{ji} \quad (22)
$$

$$
s_{Wi} = S_i - \sum_{j=1}^{n} w_{ij} \quad (23)
$$

where:

- $Q_i$: the number of demanded containers at each port $i$ (TEU);
- $S_i$: the number of excess containers at each port $i$ (TEU);
- $P_i$: the number of containers destined for port $i$ (TEU);
- $D_i$: the number of containers departing from port $i$ (TEU);
- $g_i, h_i$: auxiliary variables.

The rate of empty container repositioning, storing and leasing depends on container inflow and outflow in each port. As these flows are becoming more balanced, empty container problem loses its significance. On the other hand, the model given in this paper allows that any number of containers $z_{ij}$ smaller than demand $x_{ij}$ may be transported between ports $i$ and $j$. In such a way, the option to balance container in/out flows to/from any port is introduced. This means that the mathematical model itself has huge impact on the reduction of the empty container handling costs. Therefore, this approach is intended to lowering the needs for empty container repositioning, storing and leasing.

Another important consequence of the situation $z_{ij} < x_{ij}$, i.e. that not all transport demands at any calling port are satisfied, is the appearance of unsatisfied clients. It is especially critical if a certain number of containers of one client is transported, while the rest is rejected. In this case, the applied approach would lead to the unsatisfied customers, loss of profit and bad reputation of the shipping company. However, transport demand in each particular port is usually composed of demands coming from several clients. Therefore, any value of $z_{ij}$, which is less than $x_{ij}$, should be carefully analyzed by the decision-maker in order to select the containers to be accepted for transport. As we show in Section 5, in our examples the number of unsatisfied demands is small compared to the capacity utilization of the selected barge container ship. In addition, at the same time the needs for empty containers are reduced.

The described problem is similar to the one considered in [11]. The main difference between these two models lies in the service network design: while in [11] the sea container service is considered, here we deal with the barge container transport. Moreover, the definition of objective function is adapted to barge vessels. Finally, the structure of our model is more complex since it includes round trip time limits, downstream–upstream navigation and selection of the container quantity to be loaded at any port.

3.3. Problem complexity and optimal solution

Our ship routing problem represents a highly unconstrained version of the selective traveling salesman problem with pick up and deliveries, or equivalently, the single vehicle routing problem with selective pickup and deliveries. Therefore, the fact that it is strongly NP-hard is straightforward.

The solution of this problem defines upstream and downstream calling sequence and the number of loaded and empty containers transported between any two calling ports, while achieving maximum profit of the shipping company. The calling sequence (in one direction) is defined by the upper right (and down left) triangle of the binary matrix $X$ containing decision variables $x_{ij}$. Thus, to determine the calling sequence we have to assign values to $n^2 - n$ binary variables. At most $n - 1$ elements (for each direction) are equal to one.

The container traffic between calling ports is defined by the elements of matrices $Z = [z_{ij}]_{n \times n}$ and $W = [w_{ij}]_{n \times n}$, again for $i \neq j$. In addition, we have to determine the total round trip time $t_{tot}$, shipping company profit $Y$, and number of leased $l_{Wi}$ and stored $s_{Wi}$ empty containers at each port. These are the variables reported to the user. Moreover, there are some auxiliary variables that are determined during the solution process in connection to handling empty containers. These are binary variables $h_i, g_i$ and integers $S_i, P_i, D_i, Q_i$. As it is modeled by the constraints (16)–(23).

To summarize, we have to determine $(n^2 - n) + 2n + n^2 + 4n + 4n^2 + 4n$ integer variables and two real (floating point) values.

The CPLEX MIP solver [38] was used to optimally solve small size problem instances of the above described MILP. We were able to optimally solve instances with 15 ports within 10 min to 1 h of CPU time, and also some of the instances with 20 ports, for which the required CPU time exceeded 29 h. We could allow long execution time required to find optimal solution since the barge container ship follows the same route for at least one year period. However, the main problem for the CPLEX MIP solver to find an optimal solution of larger instances appears to be the lack of memory. Let us state that our experiments were performed on a computer with 8 GB of RAM which is fairly large. Although 16 or 32 GB of RAM are common by now, we would not expect to be able to solve much larger instances. Obviously, like in many other combinatorial optimization problems, real examples are too complex to be solved to optimality. Therefore, we propose the use of meta-heuristic approaches which are the common way to tackle these kind of problems [11,15]. In particular, we apply 0–1 MIP heuristics described in the next section. Our experiments show that the objective values obtained by MIP heuristics within only 1 h are of a very similar, or even better quality than those obtained by the CPLEX MIP solver in much (usually 10–20 times) longer time.

4. Description of the MIP heuristics

According to the mathematical formulation provided in Section 3, the ships routing problem presented in this paper is the special case of the 0–1 mixed integer programming problem (0–1 MIP):

$$
\max \{\chi(X) = c^T \xi | \xi \in X\},
$$

where

$$
X = \{\xi \in \mathbb{R}^{N(0)}|A \xi \geq b, \xi_j \in \{0, 1\} \forall j = B; \xi_j \in Z_+^{B} \forall j \in C\}
$$

is the feasible set, $N = C \cup B \cup B$ is the set of indices of all variables, $C$ is the set of indices of continuous variables, $B$ is the set of indices of binary variables, $\mathbb{R}^{N(0)}$ is the set of indices of integer variables, $\mathbb{R}^B$ is the set of indices of integer variables, $\mathbb{R}^C$ is the set of indices of integer variables, $\mathbb{R}^B \cap \mathbb{R}^C = \emptyset$, $\mathbb{R}^B \cup \mathbb{R}^C = \mathbb{R}, B \neq \emptyset$. Indeed, all problem constraints and the profit function $Y$ are linear, and the set of binary variables $\{x_{ij}, h_i, g_i|0 \leq i, j \leq N\}$ is non-empty. Therefore, it is possible to tackle this problem by using 0–1 MIP solution methods, the so-called matheuristics.

Although the term matheuristic (short from math-heuristic, also known as model-based heuristic) is still recent, a number of methods for solving optimization problems which can be considered as matheuristics have emerged over the last decades [19]. The main advantage of matheuristics is their convergence, which means that
they provide optimal solution if the running time is not limited. Neighborhood search type meta-heuristics, such as Variable Neighborhood Search (VNS) [39], are proved to be very effective when combined with optimization techniques based on the mathematical programming problem formulations. The common idea behind heuristics is the following: set some variables to some particular values by using meta-heuristics rules in order to create sub-problems to be solved within the exact MIP solver framework.

In this paper, we apply three state-of-the-art heuristics: Local Branching (LB) [17], Variable Neighborhood Branching (VNB) [18] and Variable Neighborhood Decomposition Search for 0–1 MIP problems (VNDs-MIP) [19]. In order to describe these algorithms, we first introduce some notations.

**Notations and definitions.** Let \( P \) be a given 0–1 MIP problem as defined in (24). The linear relaxation \( LP(P) \) of problem \( P \) is obtained from \( P \) by releasing the integer requirements on \( x \). For a given \( \xi \in \{0, 1\}^{|I|} \times \mathbb{Z}^{|S_p|} \times \mathbb{Z}_{\geq 0}^{|S_c|} \), and an arbitrary \( \eta \in \{0, 1\}^{|I|} \times \mathbb{Z}^{|S_p|} \times \mathbb{Z}_{\geq 0}^{|S_c|} \), the distance between \( \xi \) and \( \eta \) is defined as

\[
\delta(\xi, \eta) = \sum_{k \in I} |\xi_k - \eta_k|
\]

and can be linearized as (see [17]):

\[
\delta(\xi, \eta) = \sum_{k \in I} (1 - \eta_k) + |\eta_k - \xi_k|
\]

Further on we will denote with \( (P|C) \) the subproblem obtained by adding the following constraint to problem \( P \): we can now also introduce the following subproblem notation: \( P(k, \xi, \eta) = (P(\xi, \eta), |\delta(\xi, \eta)| \leq k) \), for \( k \in \mathbb{N} \cup \{0\} \).

The neighborhood structures \( N_k(\xi) \), \( k = k_{\text{min}} \leq k \leq k_{\text{max}} \leq |I| \), can be defined knowing the distance \( \delta(\xi, \eta) \) between any two solutions \( \xi, \eta \in X \). The set of all solutions in the \( k \)th neighborhood of \( \xi \in X \) is denoted as \( N_k(\xi) \), where

\[
N_k(\xi) = \{ \eta \in X | \delta(\xi, \eta) \leq k \}
\]

From the definition of \( N_k(\xi) \), it follows that \( N_k(\xi) \subseteq N_{k+1}(\xi) \), for any \( k \in \{k_{\text{min}}, k_{\text{min}} + 1, \ldots, k_{\text{max}} - 1\} \), since \( \delta(\xi, \eta) \leq k \) implies \( \delta(\xi, \eta) \leq k + 1 \). It is trivial that, if we completely explore neighborhood \( N_{k_{\text{max}}}(\xi) \), it is not necessary to explore neighborhood \( N_k(\xi) \).

**Local search in the 0–1 MIP solution space.** Introducing the neighborhood structures into the 0–1 MIP solution space \( X \) makes possible the employment of a classical local search in order to explore \( X \). The pseudo-code of the corresponding local search procedure, named LocalSearch-MIP, is presented in Fig. 2.

The LocalSearch-MIP explores the solution space of the input 0–1 MIP problem \( P \), starting from the given initial solution \( \xi \). Input parameter \( k^* \) controls the change of the neighborhood size during the search process. The algorithm returns the best solution found. The neighborhood \( N_k(\xi) \) of \( \xi \) is defined as a subproblem \( P(k, \xi) \). Statement \( \eta = \text{MIPSolve}(P, \xi) \) denotes a call to a generic MIP solver

\[
\text{UpdateNeighborhood}(k, k^*, \eta) =
\begin{cases}
1 & \text{if } (\text{status} = \text{‘optimalSolution’}) \land \eta \neq \eta^* \\
2 & \text{if } (\text{status} = \text{‘feasibleSolution’}) \land \eta \neq \eta^* \land \eta \neq \text{‘optimalSolution’} \\
3 & \text{if } (\text{status} = \text{‘provenInfeasible’}) \land \eta \neq \eta^* \\
4 & \text{else if } (\text{status} = \text{‘provenInfeasible’}) \land \eta \neq \eta^* \\
5 & \text{return } k
\end{cases}
\]

for a given input problem \( P \), where \( \xi \) is a given starting solution and \( \eta \) is the best solution found, returned as the result. The constant \( \text{status} \) denotes the solution status as obtained from the MIP solver.

In each iteration, the current neighborhood \( N_k(\xi) \) of the incumbent solution \( \xi^* \) is explored (line 3 in Fig. 2). If a better solution is found, the incumbent solution is accordingly updated (lines 4–5 in Fig. 2). If the current subproblem is solved exactly (the incumbent solution is proven optimal for the current subproblem) or the current subproblem is proven infeasible, the distance constraint \( \delta(\xi, \xi) \leq k \) is added to the original problem, in order to discard the neighborhood \( N_k(\xi) \) from the search space (see lines 6–8 in Fig. 2). The neighborhood size is then updated (line 9 in Fig. 2) and the whole process is iterated until the fulfillment of stopping criteria.

### 4.1. Local Branching

Local Branching (LB), introduced by Fischetti and Lodi [17], is in fact the first local search method for 0–1 MIPs, as described above (see pseudo-code in Fig. 2), which employs the linearization (26) of the Hamming distance in the 0–1 MIP solution space. For the current incumbent solution \( \xi \), the search process begins by exploring a neighborhood of \( \xi \) of a predefined size \( k = k^* \) (defined as a subproblem \( P(k, \xi) \)). A generic MIP solver is used as a black-box for solving problems \( P(k, \xi) \), for different values of \( k \). After a new incumbent is found, the whole process is iterated. The LB pseudo-code can be represented in a general form given in Fig. 2, with the special form of procedure \( \text{UpdateNeighborhood}(k, k^*, \eta) \) as given in Fig. 3.

The input parameters for Local Branching, as a special case of local search for 0–1 MIPs (see pseudo-code in Fig. 2), include an input 0–1 MIP problem \( P \), an initial solution \( \xi \) and the neighborhood size control parameter \( k^* \). The LB executes iteration by iteration until some predefined stopping criterion is satisfied. The stopping criterion normally includes the total running time allowed, but may also the maximum number of diversifications (see line 3 in Fig. 3) and/or the total number of iterations.

In this paper we propose an improved implementation of the original LB algorithm from [17]. Whereas in the original LB algorithm the initial neighborhood size is fixed to a certain predefined value (equal to 20) for all tested instances, we observe that the same value of this parameter incurs different behavior of the search process for the different problem instances. It is obvious that the neighborhood size 20 does not provide as good level of diversification in the case of instances with 650 binary variables, as in the case of instances with 110 binary variables. The same holds for any other fixed value which does not take into account specific characteristics of different instances. This is why we decide to set the value of the initial neighborhood size to 20% of the total number of binary variables of a particular instance, therefore adjusting it to the specific features of each particular instance.

### 4.2. Variable Neighborhood Branching

Variable Neighborhood Branching (VNB) is another heuristic for solving 0–1 MIP problems, using the general-purpose MIP solver as a black-box [18]. VNB adds constraints to the original problem, as in the LB method. However, in VNB, neighborhoods are changed in
a systematic manner, according to the rules of the general Variable Neighborhood Search (VNS) algorithm [39,40]. The local search method used within VNB is a deterministic VNS variant, Variable Neighborhood descent (VND) (see [41]). VND starts from a given initial feasible solution as the incumbent solution x and examines its current neighborhood entirely. In case of improvement, the same search procedure is performed starting from the improved solution x as the new incumbent. If there is no improvement, the next neighborhood of x is explored. This process is repeated until the fulfillment of the stopping criterion (which usually includes the maximum number of neighborhoods, and/or the predefined running time limit). The pseudo-code of VND for MIPs, called VND-MIP, can also be represented in a general form given in Fig. 2, but with the special form of procedure UpdateNeighborhood(k, k′, status) as given in Fig. 4.

The input parameters for VND-MIP, as a special case of local search for 0–1 MIPs (see pseudo-code in Fig. 2), include an input 0–1 MIP problem P, an initial solution x and the neighborhood size control parameter k. Note that parameter k in VND-MIP represents the maximum allowed neighborhood size, whereas in LB it represents the initial neighborhood size for a given incumbent solution vector.

The diversification in Variable Neighborhood Branching (shaking step in general VNS) is performed by choosing the first feasible solution from the disk of radii k and k * kstep, where k is the current neighborhood size, and kstep is a given input parameter. The disk size is increased as long as the feasible solution is not found. In addition to the input 0–1 MIP problem P and the initial solution x, the input parameters of VNB are the minimum neighborhood size kmin, the neighborhood size increment step kstep, the maximum neighborhood size kmax and the maximum neighborhood size kend within VND-MIP. Similarly as in the case of LB, we propose an improved implementation of the original VNB algorithm from [18]. In the original VNB algorithm, the values of kmin, kstep and kmax are fixed to 5, 5 and 150, respectively, for all tested instances. Here we observe that these values affect the search process differently for different problem instances. For example, setting the maximum neighborhood size in VNB to 150 for instances with 110 binary variables is not meaningful, since the maximum distance between any two solution vectors cannot exceed 110. On the other hand, for instances with 650 binary variables setting kmax = 150 may not provide enough diversification. A similar reasoning can be applied for any other fixed values which do not take into account specific characteristics of different instances. This is why we decide to set the values of kmax, kmin and kstep to 50, 5% and 5% of the total number of binary variables of a particular instance, respectively. In this way we exploit the specific features of each particular instance.

4.3. Variable Neighborhood Decomposition Search for 0–1 MIPs

Variable Neighborhood Decomposition Search (VNDs) is a two-level VNS scheme for solving optimization problems, based upon the decomposition of the original problem. It was proposed for the first time in [41]. Recently, VNDs has been implemented for solving 0–1 MIPs [19] and the resulting procedure is abbreviated as VNDs-MIP. The approach proposed in [19] is a VNDs based diving strategy, which combines linear programming (LP) solver, MIP solver and VND based MIP solving method (VNDs-MIP) in order to efficiently solve a given 0–1 MIP problem.

Input parameters for VNDs-MIP are an input 0–1 MIP problem P, an initial integer feasible solution x and an integer k which controls the size of the subproblems generated within VNDs-MIP. Starting from incumbent integer feasible solution x of P and an optimal solution x of LP(P), binary variables are ranked in a non-decreasing order of the modules of the differences between the values of x and x. Subproblems within VNDs are obtained by successively fixing a certain number of ordered binary variables to their values in the incumbent integer solution. In this way, the subproblem involves free binary variables which are furthest from their linear relaxation values. Then these subproblems are solved exactly or within the CPU time limit. The subproblems are changed by the hard fixing of the variables (or diving), according to the VNS rules. The pseudo-code of the VNDs-MIP procedure is provided in Fig. 5.

4.4. Large Scale Variable Neighborhood Search for MIPs

In this paper we propose an extension of the basic VNDs-MIP algorithm from [19] which also considers general integer variables (see pseudo-code in Fig. 6).

If the distance function (25) is replaced with Δ : X2 → ℝ, which takes into account general integer variables:

\[ \Delta(x, y) = \sum_{j \in B} |x_j - y_j|. \]  

(28)

it is clear that very large neighborhoods are obtained as a result.

In the large scale variant of the basic VNDs-MIP, here denoted as LS-VNDs-MIP (LS-VNDs for short), distance function (28) is employed in the VNDs stage (lines 5–6 in LS-VNDs pseudo-code from Fig. 6) and distance function (25) in the local search (i.e. VNDs-MIP) stage.

Input parameters for LS-VNDs are an input 0–1 MIP problem P, an initial integer feasible solution x and a bound integer k which controls the size of the subproblems generated within LS-VNDs. Starting from incumbent integer feasible solution x of P and an optimal solution x of LP(P), all integer variables (both binary and general integer) are ranked according to their distances from the corresponding LP solution values (line 5 in pseudo-code in Fig. 6). This is a generalization of basic VNDs-MIP, which only considers binaries in this stage. Selected integer variables are then fixed to their values in the incumbent integer feasible solution (line 9 in pseudo-code in Fig. 6). The set of variables to be fixed is updated in each iteration in a VNDs manner, as in the basic VNDs-MIP. If an improvement occurs, VND-MIP is further applied as a local search procedure (line 12 in the pseudo-code in Fig. 6), but only with neighborhoods defined in the 0–1 MIP solution space. Although it is possible to define local search in the solution space of general MIP problems (see, e.g. [42] or [43]) using the definition of distance function (28), this requires introduction of additional variables and constraints to the initial MIP problem, therefore resulting in a larger problem, which takes longer to solve. This is why we have decided to apply local search only in the 0–1 MIP solution space in this stage.

One should note that the main difference between VNDs-MIP and LS-VNDs is in the variable domain. Namely, LS-VNDs uses both binary and general integer variables, while VNDs-MIP is focused only on binary variables.
VNDS-MIP\((P, \xi', k^*)\)
1 Set proceed\(_1\) ← true, proceed\(_2\) ← true; Set \(p = |B|\);
2 Find an optimal solution \(\bar{\xi}\) of the LP relaxation of \(P\);
3 if (\(\bar{\xi}\) is integer feasible) then return \(\bar{\xi}\);
4 while (proceed\(_1\)) do
5 \(\delta_j = |\xi_j' - \bar{\xi}_j|, j \in B\); Index \(\xi_j\) so that \(\delta_j \leq \delta_{j+1}, j = 1..p - 1\);
6 Set \(q = \{j \in B \mid \delta_j \neq 0\}\);
7 Set \(k_{\text{min}} = p - q, k_{\text{step}} = [q/k^*], k_{\text{max}} = p - k_{\text{step}}, k = k_{\text{max}}\);
8 while (proceed\(_2\) and \(k \geq 0\)) do
9 Fix values of \(\xi_1, \ldots, \xi_k\) to \(\xi_1', \ldots, \xi_k'\), to obtain subproblem \(Q\);
10 \(\xi'' = \text{MIPsolve}(Q, \xi'')\);
11 if \((c^T\xi'' < c^T\xi')\) then
12 \(\xi' = \text{LocalSearch-MIP}(P, \xi'', k^*)\); break;
13 else
14 if \((k - k_{\text{step}} < k_{\text{min}})\) then \(k_{\text{step}} = \max\{\lfloor k/2 \rfloor, 1\}\);
15 Set \(k = k - k_{\text{step}}\);
16 endwhile
17 Update proceed\(_2\);
18 endwhile
19 return \(\xi'\).

Fig. 5. VNDS for 0–1 MIPs.

LS-VNDS-MIP\((P, \xi', k^*)\)
1 Set proceed\(_1\) ← true, proceed\(_2\) ← true; Set \(p = |B \cup G|\);
2 Find an optimal solution \(\bar{\xi}\) of the LP relaxation of \(P\);
3 if (\(\bar{\xi}\) is integer feasible) then return \(\bar{\xi}\);
4 while (proceed\(_1\)) do
5 \(\Delta_j = |\xi_j' - \bar{\xi}_j|, j \in B \cup G\); Index \(\xi_j\) so that \(\Delta_j \leq \Delta_{j+1}, j = 1..p - 1\);
6 Set \(q = \{j \in B \cup G \mid \Delta_j \neq 0\}\);
7 Set \(k_{\text{min}} = p - q, k_{\text{step}} = [q/k^*], k_{\text{max}} = p - k_{\text{step}}, k = k_{\text{max}}\);
8 while (proceed\(_2\) and \(k \geq 0\)) do
9 Fix values of \(\xi_1, \ldots, \xi_k\) to \(\xi_1', \ldots, \xi_k'\), to obtain subproblem \(Q\);
10 \(\xi'' = \text{MIPsolve}(Q, \xi'')\);
11 if \((c^T\xi'' < c^T\xi')\) then
12 \(\xi' = \text{VND-MIP}(P, \xi'', k^*)\); break;
13 else
14 if \((k - k_{\text{step}} < k_{\text{min}})\) then \(k_{\text{step}} = \max\{\lfloor k/2 \rfloor, 1\}\);
15 Set \(k = k - k_{\text{step}}\);
16 endwhile
17 Update proceed\(_2\);
18 endwhile
19 Update proceed\(_1\);
20 return \(\xi'\).

Fig. 6. LS-VNDS for 0–1 MIPs.
5. Computational results

In this section we present the comparison results for the application of the described MIP-based heuristic search methods. In fact, we compared 5 solution methods: CPLEX, LB, VNB, VNDS-MIP and LS-VNDS within the same CPU time limit. The computational results are presented and discussed in an effort to assess and analyze the efficiency of the presented model. We study the performance of different solution methods for our model from two points of view: first we compared solution qualities and running times and then we analyze mathematical programming aspects and transportation usefulness of the obtained results.

5.1. Experimental environment

**Hardware and software.** Our tests are performed on Intel Core 2 Duo CPU E6750 on 2.66 GHz with RAM = 8 GB under Linux Slackware 12, Kernel: 2.6.21.5. The applied MIP-based heuristics are all coded in C++programming language for Linux operating system and compiled with gcc (version 4.1.2) and the option -o2. For exact solving we used CPLEX 11.2 [38] MIP solver and AMPL [44,45] running on the same machine. Moreover, CPLEX 11.2 is used as a generic MIP solver in all tested MIP-based heuristics.

**Test bed.** The lack of publicly available test instances for the considered problem prevented us to compare our results to the similar results from the literature. Therefore, our test examples were generated randomly, in such a way that the number of ports $n$ varied from 10 to 25 with increment 5. Moreover, for each value of $n$, 5 instances were produced with different ship characteristics (carying capacities, daily charter costs, downstream and upstream speeds, engine outputs, fuel and lubricant consumptions) which are summarized in Table 1. Input data are set in accordance with the common values found at different sources like internet sites (www.portofantwerp.com, www.rotterdamportinfo.com), project [5] and papers [6,21,23,46]. In this way we produced hard and easy examples for each problem size. The set of instances with their basic MIP properties is listed in Table 2. All the instances in the form of AMPL *.dat files can be downloaded from www.mi.sanu.ac.rs/~tanjad/ships.htm.

**Parameters.** According to our preliminary experiments, we have decided to use the different parameter settings for different instance sizes. Thus, different parameter settings were used for the four groups of instances generated for 10, 15, 20 and 25 ports, respectively. According to our preliminary experiments, the total running time limit ($t_{lim}$) for all methods (including CPLEX MIP solver alone) was set to 60, 900, 1800 and 3600 s for 10, 15, 20 and 25 ports, respectively. In all heuristic methods, the time limit for subproblems ($t_{sub}$) within the main method was set to 10% of the total running time limit. In two VNDS heuristics, the time limit for the VND-MIP procedure within VNDS is also set to 10% of the total running time limit. In VNB, parameters regarding initialization and change of neighborhood size are set in the following way: the maximum neighborhood size $k_{max}$ is approximately 50% of the number of binary variables and minimum neighborhood size $k_{min}$ and neighborhood size increase step $k_{step}$ are set to 5% of the maximum neighborhood size. Namely, $k_{min} = k_{step} = \lfloor b/20 \rfloor$ and $k_{max} = 10 k_{min}$, where $b = \lvert b \rvert$ is the number of binary variables for the particular instance. For example, for 10 ports this yields $k_{min} = k_{step} = 6$ and $k_{max} = 60$. In LB, the initial neighborhood size $k$ is set to approximately 20% of the number of binary variables for the particular instance, i.e. $k = \lfloor b/5 \rfloor$. Actual values for all parameters are summarized in Table 3. All CPLEX MIP solver parameters are set to their default values.

5.2. CPLEX results

In Table 4, results for the tested instances are presented, as obtained by the CPLEX MIP solver invoked by AMPL, without any running time limitations.

The first column of this table lists the names of our test examples. The best objective values are given in the second column while the corresponding running times are provided in the column three. The column four contains the CPLEX MIP solver status at the end of computation. Total running times the CPLEX MIP solver needed to reach the corresponding statuses are indicated in the last column. The objective values from column 2 which are proven optimal are bolded. There is a number of instances for which the CPLEX MIP solver has managed to obtain a near-optimal solution, but failed to find an optimal one, even without any running time limitations (as indicated in column 4 of Table 4). It can be observed that, for largest instances, CPLEX MIP solver failed to prove the (near)optimality of the solution, due to the insufficient memory resources.

### Table 1

<table>
<thead>
<tr>
<th>Container barge ships</th>
<th>No. units</th>
<th>TEU</th>
<th>$P_{max}$ [kW]</th>
<th>Total TEU</th>
<th>$v_1$ [km/h]</th>
<th>$v_2$ [km/h]</th>
<th>dcc [$]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ship 1</td>
<td>Motorized cargo push vessel</td>
<td>1</td>
<td>90</td>
<td>2 × 607</td>
<td>215</td>
<td>19</td>
<td>11</td>
</tr>
<tr>
<td></td>
<td>Pushed barges</td>
<td>1</td>
<td>165</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>Ship 2</td>
<td>Motorized cargo push vessel</td>
<td>1</td>
<td>145</td>
<td>2 × 1204</td>
<td>409</td>
<td>20</td>
<td>12</td>
</tr>
<tr>
<td></td>
<td>Pushed barges</td>
<td>2</td>
<td>132</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>Ship 3</td>
<td>Motorized cargo push vessel</td>
<td>1</td>
<td>77</td>
<td>2 × 565</td>
<td>242</td>
<td>19</td>
<td>11</td>
</tr>
<tr>
<td></td>
<td>Pushed barges</td>
<td>1</td>
<td>165</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>Ship 4</td>
<td>Motorized cargo push vessel</td>
<td>1</td>
<td>60</td>
<td>667</td>
<td>180</td>
<td>16</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>Pushed barges</td>
<td>2</td>
<td>60</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>Ship 5</td>
<td>Motorized cargo push vessel</td>
<td>1</td>
<td>98</td>
<td>2 × 927</td>
<td>338</td>
<td>20</td>
<td>12</td>
</tr>
<tr>
<td></td>
<td>Pushed barges</td>
<td>4</td>
<td>60</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
</tbody>
</table>

### Table 2

MIP characteristics of various size test instances.

<table>
<thead>
<tr>
<th>Instance</th>
<th>Number of variables</th>
<th>Number of constraints</th>
</tr>
</thead>
<tbody>
<tr>
<td>port10,1 – port10,5</td>
<td>352</td>
<td>110</td>
</tr>
<tr>
<td>port15,1 – port15,5</td>
<td>752</td>
<td>240</td>
</tr>
<tr>
<td>port20,1 – port20,5</td>
<td>1302</td>
<td>420</td>
</tr>
<tr>
<td>port25,1 – port25,5</td>
<td>2002</td>
<td>650</td>
</tr>
</tbody>
</table>

### Table 3

Parameter values for different MIP heuristics.

<table>
<thead>
<tr>
<th>Instance</th>
<th>Time</th>
<th>LB</th>
<th>VNB</th>
</tr>
</thead>
<tbody>
<tr>
<td>port10,1 – port10,5</td>
<td>t_{lim} 60</td>
<td>6</td>
<td>22</td>
</tr>
<tr>
<td>port15,1 – port15,5</td>
<td>t_{lim} 900</td>
<td>90</td>
<td>48</td>
</tr>
<tr>
<td>port20,1 – port20,5</td>
<td>t_{lim} 1800</td>
<td>180</td>
<td>84</td>
</tr>
<tr>
<td>port25,1 – port25,5</td>
<td>t_{lim} 3600</td>
<td>360</td>
<td>130</td>
</tr>
</tbody>
</table>
5.3. Comparison results of MIP-heuristic methods

Here we present the comparison results of five methods tested: CPLEX MIP solver, LB, VNB, VNDS and LS-VNDS, within the same amount of CPU time. The imposed running time limitations (see parameter settings above) are listed in Table 3.

The best objective function (profit) values obtained by all MIP methods are provided in Table 5. The highest profit values for each example are bolded, indicating the method(s) that performed the best in this case. At the same time, running times required to obtain the best profit value by each method are presented in Table 6 and the shortest among them is bolded, regardless the quality of the obtained objective value.

According to the average profit values from Table 5, we can see that VNB has the best performance quality-wise, whereas other methods are worse than CPLEX MIP solver. However, regarding the running times presented in Table 6, the CPLEX MIP solver is the slowest, with 1518 s average running time, followed by VNB with 1369 s average running time. LB and VNDS show similar performance with average running times of 517 s and 509 s, respectively.

When the speed is more important then the solution quality, the LS-VNDS heuristic is clearly the best choice, with only 188 s average execution time.

In summary, we may conclude that using the heuristic methods for tackling the presented ship routing problem is beneficial, both regarding the solution quality and (especially) the execution time. VNB heuristic proves to be better than the CPLEX MIP solver regarding both criteria (solution quality/execution time), LB and VNDS-based heuristics do not achieve as good solution quality as CPLEX, but have significantly better execution time, especially the LS-VNDS method which is approximately 8 times faster than CPLEX. Soft variable fixing (VN and LB) appears to be more effective (quality-wise) for this model than the hard variable fixing (VNDS-based methods). The solution quality performance of the basic VNDS-MIP may be explained by the fact that the number of general integer variables in all instances is more than twice as large as the number of binary variables, and therefore the subproblems generated during the VNDS-MIP process by fixing only binary variables are still large and not so easy for the CPLEX MIP solver. Therefore, the improvement in VNDS-MIP usually does not occur.

<table>
<thead>
<tr>
<th>Instance</th>
<th>Best CPLEX obj. value</th>
<th>Obtained after (s)</th>
<th>CPLEX status</th>
<th>Status time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>port10_1</td>
<td>22,339.01</td>
<td>5.69</td>
<td>Proven optimal</td>
<td>6.72</td>
</tr>
<tr>
<td>port10_2</td>
<td>24,738.23</td>
<td>6.23</td>
<td>Proven optimal</td>
<td>0.39</td>
</tr>
<tr>
<td>port10_3</td>
<td>23,294.74</td>
<td>6.23</td>
<td>Near-optimal</td>
<td>6.27</td>
</tr>
<tr>
<td>port10_4</td>
<td>20,686.27</td>
<td>0.99</td>
<td>Proven optimal</td>
<td>1.04</td>
</tr>
<tr>
<td>port10_5</td>
<td>25,315.00</td>
<td>2.69</td>
<td>Near-optimal</td>
<td>2.78</td>
</tr>
<tr>
<td>port15_1</td>
<td>12,268.96</td>
<td>279.33</td>
<td>Proven optimal</td>
<td>287.75</td>
</tr>
<tr>
<td>port15_2</td>
<td>25,340.00</td>
<td>238.66</td>
<td>Near-optimal</td>
<td>65.43</td>
</tr>
<tr>
<td>port15_3</td>
<td>13,798.22</td>
<td>238.66</td>
<td>Near-optimal</td>
<td>267.59</td>
</tr>
<tr>
<td>port15_4</td>
<td>22,372.58</td>
<td>1088.45</td>
<td>Proven optimal</td>
<td>1134.38</td>
</tr>
<tr>
<td>port15_5</td>
<td>15,799.96</td>
<td>74.47</td>
<td>Near-optimal</td>
<td>131.51</td>
</tr>
<tr>
<td>port20_1</td>
<td>19,856.25</td>
<td>11016.10</td>
<td>Out of memory</td>
<td>29622.45</td>
</tr>
<tr>
<td>port20_2</td>
<td>33,204.57</td>
<td>34708.11</td>
<td>Proven optimal</td>
<td>36826.32</td>
</tr>
<tr>
<td>port20_3</td>
<td>20,969.39</td>
<td>15508.74</td>
<td>Out of memory</td>
<td>29835.21</td>
</tr>
<tr>
<td>port20_4</td>
<td>27,750.46</td>
<td>14426.23</td>
<td>Out of memory</td>
<td>18753.01</td>
</tr>
<tr>
<td>port20_5</td>
<td>24,257.89</td>
<td>51176.12</td>
<td>Proven optimal</td>
<td>7364.10</td>
</tr>
<tr>
<td>port25_1</td>
<td>21,389.57</td>
<td>11515.30</td>
<td>Out of memory</td>
<td>28301.91</td>
</tr>
<tr>
<td>port25_2</td>
<td>33,789.32</td>
<td>28793.07</td>
<td>Out of memory</td>
<td>29226.30</td>
</tr>
<tr>
<td>port25_3</td>
<td>23,201.33</td>
<td>12069.33</td>
<td>Out of memory</td>
<td>26702.03</td>
</tr>
<tr>
<td>port25_4</td>
<td>28,096.20</td>
<td>25409.96</td>
<td>Out of memory</td>
<td>24725.92</td>
</tr>
<tr>
<td>port25_5</td>
<td>25,899.97</td>
<td>10722.78</td>
<td></td>
<td>30587.42</td>
</tr>
</tbody>
</table>

Table 4

Objective values–profits (bolded optimal) and corresponding execution times as obtained by CPLEX/AMPL without any running time limitations.
Table 6
Comparison results for five methods – Computational times.

<table>
<thead>
<tr>
<th>Instance</th>
<th>CPU time required to obtain best objective value for</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CFLEX</td>
</tr>
<tr>
<td>port10_1</td>
<td>21.30</td>
</tr>
<tr>
<td>port10_2</td>
<td>0.99</td>
</tr>
<tr>
<td>port10_3</td>
<td>19.79</td>
</tr>
<tr>
<td>port10_4</td>
<td>3.03</td>
</tr>
<tr>
<td>port10_5</td>
<td>8.83</td>
</tr>
<tr>
<td>port15_1</td>
<td>900.00</td>
</tr>
<tr>
<td>port15_2</td>
<td>212.76</td>
</tr>
<tr>
<td>port15_3</td>
<td>873.43</td>
</tr>
<tr>
<td>port15_4</td>
<td>900.00</td>
</tr>
<tr>
<td>port15_5</td>
<td>426.72</td>
</tr>
<tr>
<td>port20_1</td>
<td>1800.00</td>
</tr>
<tr>
<td>port20_2</td>
<td>1800.00</td>
</tr>
<tr>
<td>port20_3</td>
<td>1800.00</td>
</tr>
<tr>
<td>port20_4</td>
<td>1800.00</td>
</tr>
<tr>
<td>port20_5</td>
<td>1800.00</td>
</tr>
<tr>
<td>port25_1</td>
<td>3600.00</td>
</tr>
<tr>
<td>port25_2</td>
<td>3600.00</td>
</tr>
<tr>
<td>port25_3</td>
<td>3600.00</td>
</tr>
<tr>
<td>port25_4</td>
<td>3600.00</td>
</tr>
<tr>
<td>port25_5</td>
<td>3600.00</td>
</tr>
<tr>
<td>Average</td>
<td>1518.34</td>
</tr>
</tbody>
</table>

Table 7 clearly indicates the significance of economic scaling in the shipping industry. Since operating, voyage and capital costs do not increase in proportion to the TEU capacity of the vessel, using a bigger ship reduces the unit cost per TEU. As unit costs are getting lower, barge container ships with higher carrying capacity are becoming more profitable. However, profit increase can be achieved with the assumption that the cargo or container flows between ports are sufficient to achieve high level of ship TEU capacity utilization. From Table 1 we can see that ships 2 and 5 have significantly larger capacities (409 TEU and 338 TEU, respectively) than the others. Therefore, in Table 7 the highest objective values are always obtained for one of these two ships.

Maximum turnaround time max is set to 21 days for instances with 10 ports, and to 28 days for all other instances (with 15, 20 and 25 ports). Minimum turnaround time min is set to 𝑚𝑖𝑛 = max − 1. These settings enable us to examine the importance of capital costs (time charter costs in our case) in barge container shipping. Our conclusion coincides with previously reported results in the barge container transport [1] where it was indicated that capital costs represent the highest portion of a shipping company total costs. It is also the main reason why profits acquired for 10 ports are, in many instances, higher than profits reached for larger number of possibly calling ports for the same type of ship: the total time charter costs are much higher in routes obtained for instances with 15, 20 and 25 ports in comparison with 10 ports instances. Therefore, total revenue is charged more with this cost for examples where max is set to 28 days.

On the other hand, the contribution and the importance of total time charter costs is also evident from the values of the objective function of ship 4 in the examples with 15, 20 and 25 ports: Daily time charter costs of ship 4 are much lower than for the other 4 ships. This impacts its second best position for the latest three types of instances.

From Tables 4 and 5 we can draw the conclusions about the best barge container ship, and consequently the container lines, to be used for any number of calling ports. For example, by comparing results for 10 ports in Tables 4 and 5, it is easy to see that container barge ship 5 achieves the highest profit. Relevant objective value, in this case, has been obtained by LS-VDNS method. Therefore, an overview of all the other results will be based on the outcomes provided by this heuristic method. Having this in mind, optimal route of barge container ship 5 for 10 possible ports of call includes the callings at the following ports, in both directions:

- **Upstream:** 1 → 3 → 5 → 6 → 7 → 8 → 9 → 10.
- **Downstream:** 10 → 9 → 8 → 7 → 6 → 5 → 4 → 3 → 1.

Fig. 10 gives a schematic overview of the container flows at the considered inland waterway. These flows are based on the number of both loaded (𝑤𝑖) and empty (𝑤𝑗) containers to be transported by barge container ship 5 between any two ports in the obtained route. Values of these variables, as well as results relating to the structure of transshipped containers in each port (number of loaded and unloaded containers), directions of container flows and density of traffic are given in the so called “downward” (“chess”) Table 8.

Total costs of the considered barge container ship on its route, composed of transport related and port related costs (daily time charter costs, fuel and lubricant costs, port taxes and cargo handling costs), as well as empty containers repositioning costs (empty containers storage, short-term leasing and handling costs) are 196,999.68 $0. As total revenue, acquired due to transport of loaded containers along the route is 222,315.00 $0, the shipping company profit becomes 25,315.32 $0.

Average capacity utilization of the barge container ship 5 is 88.14% (89.05% during the upstream and 87.08% in the downstream navigation). The total number of transported containers is 1062 TEU. It represents 51.2% of all transport demands, and 82.7 % of the demands in the calling ports. These figures clearly indicate the level of demands satisfaction: only 17.3% of them were not satisfied although the capacity of the used barge container ship is almost four times smaller than the transport demands in the calling ports. Similarly, 51.2% of total transport demands (taking into account also
Table 7
Optimal ships for various numbers of ports.

<table>
<thead>
<tr>
<th>10 ports instances</th>
<th>Highest obj. value</th>
<th>15 ports instances</th>
<th>Highest obj. value</th>
<th>20 ports instances</th>
<th>Highest obj. value</th>
<th>25 ports instances</th>
<th>Highest obj. value</th>
</tr>
</thead>
<tbody>
<tr>
<td>port10_1</td>
<td>22,339.01</td>
<td>port15_1</td>
<td>12,268.96</td>
<td>port20_1</td>
<td>19,586.02</td>
<td>port25_1</td>
<td>21,619.18</td>
</tr>
<tr>
<td>port10_2</td>
<td>24,738.23</td>
<td>port15_2</td>
<td>25,340.00</td>
<td>port20_2</td>
<td>33,204.26</td>
<td>port25_2</td>
<td>33,528.22</td>
</tr>
<tr>
<td>port10_3</td>
<td>23,294.74</td>
<td>port15_3</td>
<td>13,798.64</td>
<td>port20_3</td>
<td>21,043.05</td>
<td>port25_3</td>
<td>23,019.65</td>
</tr>
<tr>
<td>port10_4</td>
<td>20,686.27</td>
<td>port15_4</td>
<td>22,372.58</td>
<td>port20_4</td>
<td>27,962.31</td>
<td>port25_4</td>
<td>28,388.23</td>
</tr>
<tr>
<td>port10_5</td>
<td>25,315.00</td>
<td>port15_5</td>
<td>15,800.00</td>
<td>port20_5</td>
<td>24,235.86</td>
<td>port25_5</td>
<td>24,621.21</td>
</tr>
</tbody>
</table>

Fig. 7. Optimal route of Barge container ship 5 for 10 possibly calling ports and schematic overview of obtained container flows.

Table 8
Container flows on the route of Barge container ship 5.

<table>
<thead>
<tr>
<th>Loading ports</th>
<th>Unloading ports</th>
<th>P1</th>
<th>P3</th>
<th>P4</th>
<th>P5</th>
<th>P6</th>
<th>P7</th>
<th>P8</th>
<th>P9</th>
<th>P10</th>
<th>Total loaded [TEU]</th>
<th>Traffic density [TEUkm/km]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P1</td>
<td>-</td>
<td>37</td>
<td>37</td>
<td>47</td>
<td>47</td>
<td>49</td>
<td>63</td>
<td>53</td>
<td>331</td>
<td>-</td>
<td>331</td>
<td>328</td>
</tr>
<tr>
<td>P3</td>
<td>37</td>
<td>-</td>
<td>0</td>
<td>2</td>
<td>6</td>
<td>3</td>
<td>12</td>
<td>42</td>
<td>37</td>
<td>-</td>
<td>338</td>
<td></td>
</tr>
<tr>
<td>P4</td>
<td>45</td>
<td>0</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>45</td>
<td>338</td>
<td></td>
</tr>
<tr>
<td>P5</td>
<td>47</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>19</td>
<td>13</td>
<td>24</td>
<td>56</td>
<td>52</td>
<td>-</td>
<td>338</td>
<td></td>
</tr>
<tr>
<td>P6</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>4</td>
<td>18</td>
<td>25</td>
<td>47</td>
<td>-</td>
<td>338</td>
<td></td>
</tr>
<tr>
<td>P7</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>11</td>
<td>27</td>
<td>11</td>
<td>49</td>
<td>-</td>
<td>338</td>
<td></td>
</tr>
<tr>
<td>P8</td>
<td>71</td>
<td>0</td>
<td>11</td>
<td>8</td>
<td>-</td>
<td>-</td>
<td>5</td>
<td>11</td>
<td>16</td>
<td>90</td>
<td>265</td>
<td></td>
</tr>
<tr>
<td>P9</td>
<td>52</td>
<td>18</td>
<td>12</td>
<td>22</td>
<td>-</td>
<td>11</td>
<td>-</td>
<td>23</td>
<td>-</td>
<td>115</td>
<td>159</td>
<td></td>
</tr>
</tbody>
</table>

Total unloaded [TEU]
| Up            | -    | 35   | -    | 37   | 47   | 49   | 89   | 129  | 159  | 543  | -                 | -                           |
| Down          | 328  | 47   | 45   | 52   | -    | 17   | 9    | -    | 480  | -    | -                 | -                           |

The ports that are not included in the route) were satisfied by the barge container ship of more than six time smaller capacity. The satisfaction of total transport demands can be easily improved by iterative application of the same model to the remaining demands.

The total round trip time of the barge container ship 5 on its route is 503.97 h or 20.99 days. This means that arrival time of barge ship will be overlapping with the relevant maritime container ship on the main route. Therefore, transshipment of containers between these two ships, with no significant storage time, will be possible to organize on such routes.

Increased power output enables ship to sail with higher speed, but, at the same time, contribute to the higher fuel consumption and fuel costs (similar technological level). On the other hand, a bigger ship means increased carrying capacities and positive effects on units costs, but leads to the higher time charter costs. Determining optimal barge container ship to be employed on any route should be a result of compromise between all the above mentioned parameters. Therefore, the model proposed in this paper may have significant role in assisting the barge shipping companies to make the most suitable decisions regarding engagement of barge container ships on the available routes.

6. Conclusion

This paper deals with the optimization of transport routes of barge container ships from a shipping company point of view. We addressed the barge container ship routing problem as a problem of maximizing the shipping company profit while picking up and
delivering containers along the inland waterway with empty container repositioning. To the best of our knowledge, the published research on the joint optimization of barge container shipping and empty container repositioning in liner barge transport systems is scarce.

We identified the considered barge container ship routing problem as a special case of the 0–1 MIP problem. Therefore, we treated this problem by applying state-of-the-art heuristics for 0–1 MIP problems: Local Branching (LB), Variable Neighborhood Branching (VNB) and two variants of Variable Neighborhood Decomposition Search for 0–1 MIP problems (VND-MIP and LS-VNDS). Using the heuristic methods for tackling the presented ship routing problem is beneficial, both regarding the solution quality and (especially) the execution time. The VNB heuristic proves to be better than the CPLEX MIP solver regarding both criteria (solution quality/execution time), LB and VND-based heuristics do not achieve as good solution quality as CPLEX, but have significantly better execution time, especially the LS-VNDS method which is approximately 8 times faster than CPLEX.

From the transportation usefulness point of view, provided the container volume and port facilities are available, we can conclude that an owner of a large barge container ship has more chances to generate a positive cash-flow than the companies possessing only smaller barge container ships. This explains why, over the last century, barge container ships have become bigger. The penalty of increasing the ship size is the loss of flexibility, which impacts on the revenue side by limiting the number of ports that could be visited.

The model and solution method given in this research could be very useful practical tool for the barge container carriers to make long term strategic decisions about establishing barge container hinterland transport routes. They can solve their real life problems, test different solutions of the problems and choose those which are more suitable for their own needs. Therefore, the planning process in the barge shipping company could be improved by applying the proposed decision support system based on our optimization model, which would significantly impact the shipping company business results.

For the future research, this study can be extended in several directions. The extension to a multi-ship problem is straightforward. Many important factors like allowing the change of a ship route during planning horizon, detailed modeling of ship service in ports, stochastic formulation of some parameters like container demand may be further included. However, it will significantly increase the complexity of the problem.

Acknowledgements

This research has been partially supported by NSF Serbia, Grants Nos. 01174010, 01174033, TR36024 and TR36027. The authors would like to thank Professor Theodor Gabriel Crainic from Département de management et technologie Université du Québec à Montréal and CIRRELT, Université de Montréal and Ms Berit Dangaard Brøuer from the Department of Management Engineering, Technical University of Denmark for their valuable comments and suggestion on the presented material. We would also like to thank the anonymous referees for the valuable suggestions that led to the improved presentation of the results described in this paper.

References


