ITC 3/47
Journal of Information Technology and Control
Vol. 47 / No. 3 / 2018
pp. 471-488
DOI 10.5755/j01.itc.47.3.20420

Variable Neighborhood Search Methods for the Dynamic Minimum Cost Hybrid Berth Allocation Problem

Received 2018/03/24 | Accepted after revision 2018/08/16

http://dx.doi.org/10.5755/j01.itc.47.3.20420

Variable Neighborhood Search Methods for the Dynamic Minimum Cost Hybrid Berth Allocation Problem

Nataša Kovač
Faculty of Applied Sciences, University of Donja Gorica, Donja Gorica, 81000 Podgorica, Montenegro, knatasa@ac.me

Tatjana Davidović
Mathematical Institute of the Serbian Academy of Science and Arts, Kneza Mihaila 36, 11000 Belgrade, Serbia
tanjad@turing.mi.sanu.ac.rs

Zorica Stanimirović
Faculty of Mathematics, University of Belgrade, Studentski trg. 16/IV, 11 000 Belgrade, Serbia, zoricast@matf.bg.ac.rs

Corresponding author: knatasa@ac.me

This study considers the Dynamic Minimum Cost Hybrid Berth Allocation Problem (DMCHBAP) with fixed handling times of vessels. The objective function to be minimized consists of three components: the costs of positioning, waiting, and tardiness of completion for all vessels. Having in mind that the speed of finding high-quality solutions is of crucial importance for designing an efficient and reliable decision support system in container terminal, metaheuristic methods represent the natural choice to deal with DMCHBAP. Four variants of Variable Neighborhood Search (VNS) metaheuristic are designed for DMCHBAP. All four proposed VNS methods are evaluated on four classes of randomly generated instances with respect to solution quality and running times. The conducted computational analysis indicates that all four VNS-based methods represent promising solution approaches to DMCHBAP and similar problems in maritime transportation.

KEYWORDS: container terminal, scheduling vessels, penalties, metaheuristics, variable neighborhood search.

1. Introduction

Berth Allocation Problem (BAP) is one of the most studied topics in the optimization of maritime transportation. BAP assumes that a set of vessels needs to be allocated to the berths within some planning horizon in such a way that some objective function is optimized. BAP is proved to be NP-hard in [32]. The most detailed classification scheme for BAPs is proposed in [2] and extended in [3]. The classification is based on four attributes: spatial, temporal, handling time and performance measure.

Spatial attribute classifies BAPs as discrete, continuous, hybrid, or draft. In the discrete case (DBAP), each vessel may be allocated only to one berth at a time, while in the continuous case, a vessel can be allocated to any position on quay. Hybrid layout (HBAP) is obtained if vessels can share one berth or one vessel can occupy more than one berth. The fourth BAP layout describes vessel's berthing position based on its draft.

The most common BAP models with respect to the temporal attribute are static and dynamic. In the static model, arrival times impose soft constraints on the berthing times, meaning that a vessel can be speeded up or slowed down. The dynamic model assumes fixed arrival times of the vessels, meaning that they cannot berth before the expected arrival time. According to the handling time attribute, BAP can assume fixed or variable handling times. Handling times may vary depending on the berthing position, on the assignment of Quay Cranes (QCs), or on a QC operation schedule. The performance measure attribute describes the objective function of a considered BAP. Detailed surveys of BAP variants can be found in [3] and [44].

This paper considers a variant of dynamic BAP, denoted by Dynamic Minimum Cost Hybrid Berth Allocation Problem (DMCHBAP) and classified as $hybr \mid dyn \mid fix \mid \sum(w_1 pos + w_2 wait + w_3 tard)$, according to the notation from [2]. Hybrid layout studied in this paper, corresponds to the case shown in Fig. 3d from [2]. The objective function is based on the one proposed in [40] and it is adapted to the dynamic BAP. The objective function is a weighted sum of three components: berthing of a vessel apart from its preferred berthing position, waiting of a vessel with respect to the expected arrival time, and tardiness of a vessel against its due date. This form of objective function reflects the real requirements in majority of ports [16, 39, 40, 48].

Terminal manager needs a fast and efficient decision support system, in order to meet all requirements of the port as a highly dynamic system. The necessity of quickly providing high-quality solution for BAP was a motivation for many authors to apply metaheuristic methods, such as: simulated annealing, tabu search, ant colony optimization, particle swarm optimization, etc. [25]. On the other hand, experimental results from [7] related to static MCHBAP, showed that even small BAP instances are too complex for exact and MIP-based heuristics solvers. These results, as well as the fact that dynamic BAP is harder to solve than its static variant [11], motivated us to use metaheuristic methods as solution approaches to DMCHBAP.

To the best of our knowledge, the only paper dealing with hybrid variant of dynamic BAP is [47]. The authors considered hybrid BAP in bulk ports with an aim to minimize the total service times of vessels and applied Squeaky Wheel Optimization (SWO). Having in mind that the objective function of our DMCHBAP is different from the one considered in [47], SWO from that paper cannot be directly applied to DMCHBAP. We have developed several variants of SWO adapted to the considered BAP. However, even the best SWO variant provided results that are far from satisfactory ones, with respect to both solution quality and running times (see Section 5). Therefore, SWO is excluded from our consideration as a solution method to DMCHBAP.

The choice of Variable Neighborhood Search (VNS) as a metaheuristic method for DMCHBAP is motivated by studies [19] (considering minimum cost static discrete BAP), [6] and [26] (dealing with static MCHBAP). The authors of [19] proposed general VNS exploring three neighborhoods (local insertion, interchange and insertion) in order to minimize the objective function composed of waiting and handling costs, lateness penalties and earliness premiums (considered as benefits and therefore appearing with negative sign). In [6], a deterministic variant of VNS, known as Variable Neighborhood Descent (VND), was successfully applied to static MCHBAP, while [26] proposed General Variable Neighborhood Search (GVNS), which showed to be a promising solution approach for the same problem. In general, if some constraints are excluded, MCHBAPs may be ob-

served as 2-D packing problems, with the goal to pack smaller rectangles into a bigger one of predetermined size. The main characteristic of this kind of problems is that, in order to improve a given (locally minimal) solution, it is required to significantly degrade its quality by some specific transformations. Although VNS, as an improvement forcing method, may not seem as adequate solution approach, the studies [6] and [26] showed that the use of sophisticated data structures and definitions of neighborhoods in VNS-based methods ensure their good performance when solving static MCHBAP. Therefore, starting from VND [6] and GVNS [26], we have designed VND and GVNS approaches for DMCHBAP. In addition, we develop a Multi-Start VND (MS-VND) and Skewed Variable Neighborhood Search (SVNS) as new VNS-based methods for DMCHBAP. The proposed VNS approaches use adequate solution representation, neighborhood structures, and search strategies, which are adapted to the considered DMCHBAP. All four metaheuristic approaches are evaluated on four classes of randomly generated DMCHBAP test instances.

The rest of this paper is organized as follows. A brief review of recent papers addressing metaheuristic approaches to dynamic variants of BAP is given in Section 2. Section 3 introduces the considered DMCHBAP. In Section 4, we provide a detailed description of four VNS-based metaheuristic for DMCHBAP: VND, MS-VND, GVNS, and SVNS. Experimental results and analysis are presented in Section 5. Concluding remarks and some directions for future work are given in Section 6.

2. Related Work

In recent literature dealing with dynamic BAP, dynamic vessel arrivals are considered and addressed by several variants of metaheuristic methods, such as randomized Local Search (LS), Tabu Search (TS), Genetic Algorithm (GA), Simulated Annealing (SA), Particle Swarm Optimization (PSO), etc. Nishimura et al. [36] presented GA for the discrete space and dynamic vessels' arrival times for berth scheduling problem. Imai et al. [21] presented a formulation of the dynamic discrete BAP at a terminal with indented berths, and proposed GA as a solution method. Han et al. [17] combined GA with SA in the case of dynamic discrete BAP in order to minimize the total service time of all the vessels. Theofanis et al. [45] presented an optimization-based GA for the dynamic discrete BAP with an aim to minimize the total weighted service time of vessels that may have various service priorities. Dynamic berth allocation and Quay Crane Assignments Problem (QCAP) was considered in [4]. A hybrid method, obtained by combining parallel GA and a constructive heuristic algorithm for generating promising solutions, is applied to solve QCAP. Discrete space and dynamic vessel arrival BAP with stochastic vessel handling times and known probability distributions was studied in [22]. Two objectives were considered, risk and total service time, which were minimized by Evolutionary Algorithm (EA).

Golias and Haralambides in [14] concurrently minimized vessels' tardiness and waiting time and maximized the premium from vessels' early departure in the case of dynamic discrete BAP. As a solution method, the authors used GA previously proposed in [15]. Hierarchical optimization approach for dynamic discrete BAP was studied in [42], involving two conflicting objective functions that correspond to two levels of hierarchy. To solve this problem, the authors designed GA based on the *k-th best algorithm*. The studies [49] and [50] considered dynamic discrete BAP that minimizes total waiting time of calling vessels, where arrival times and handling times of vessels were considered as stochastic parameters following the normal distribution. In both papers, a reduced search space GA, based on the characteristics of the optimal solution, was used. In [41], dynamic continuous BAP and QCAP were studied and addressed by GA approach. Two conflicted objectives were considered: minimizing the total service time and maximizing the robustness or buffer times.

TS was used to solve dynamic discrete BAP and it was further extended for continuous BAP in [5]. The authors considered the BAP model with the objective function that minimizes the sum of the service times for vessels. Minimization of the total weighted flow time in dynamic continuous BAP was studied in [31], where two variants of Greedy Randomized Adaptive Search Procedure (GRASP) were developed to find near optimal solutions. The dynamic discrete BAP was solved in [37] by combining clustering search method and SA, where the objective function minimizes the

weighted sum of service times. In [23], the cost of the non-optimal berthing location and costs of tardiness were minimized in the case of continuous BAP.

In [46], PSO was applied for the first time as solution approach to dynamic discrete BAP. Golias et al. [13] used lambda-optimal based heuristic for dynamic discrete BAP to guarantee local optimality at a predefined neighborhood. In [28] and [29], Partial Optimization Metaheuristic Under Special Intensification Conditions (POPMUSIC) was used for dynamic discrete BAP with the aim to minimize the total (weighted) service time of the incoming container vessels. The authors developed two variants of the algorithm by hybridization of the metaheuristic approach with mathematical programming. More precisely, CPLEX solver is used to exactly solve defined sub-problems. Based on promising experimental results, the authors concluded that POPMUSIC has clear advantage over the best approximate approaches known to date, and that it can be successfully applied to this kind of problems as stand-alone solution technique. Hybridization of POPMUSIC and the branch-and-cut algorithm incorporated in CPLEX solver is also used in [30] in the case of BAP under time-dependent limitations where berthing depends on tidal and water depth constraint. A new model for the dynamic BAP was proposed by Simrin and Diabat in [43]. The authors used GA to minimize the total time that vessels spend at a terminal. Alsoufi et al. [1] proposed a mathematical model for robust berth allocation and implemented hybrid meta-heuristic based on GA and Branch-and-Cut algorithm in order to minimize the total tardiness of vessel departure time and to reduce the cost of berthing.

Gargari and Niasar in [10] applied VNS method to minimize vessels waiting time and berth idle time in the case of the discrete dynamic BAP. The authors used two types of neighborhoods (insert and swap), which were explored only in the case when a new allocation produced a feasible solution. Discrete dynamic BAP under tidal and water depth constraints is solved in [27] by applying Adaptive VNS. This variant of VNS involves multi-start strategy that exploits an adaptive mechanism with the goal to minimize the service time of each vessel. In [8], VNS approach was used to solve tactical BAP that incorporates uncertainty in vessel arrival times. VNS for tactical BAP is based on two neighborhood structures: reinsertion movement, which is used in shaking phase and in VND part of local search, and interchange movement, which is explored only in VND part. An overview of solution approaches to different variants of BAP can be found in [25].

3. Dynamic Minimum Cost Hybrid Berth Allocation Problem

DMCHBAP, considered in this paper, deals with assigning a berthing position and a berthing time to each incoming vessel to be served within a given planning horizon, with an aim to minimize the total berthing cost. This cost consists of three components: the costs of positioning, waiting, and tardiness of completion for all vessels. The vessels are defined by the following set of data: expected arrival time, processing time, length, due date, preferred berth position, and penalties.

As illustrated in Fig. 3, a solution to DMCHBAP can be presented in a space-time-diagram. It is assumed that both coordinates are discrete, i.e., the space is modelled by the berth indices, whereas the time horizon is divided into segments, such that berthing time of each vessel is represented by an integer. Each vessel is represented by a rectangle with the height equal to the length of a vessel (expressed by the number of berths), while the width corresponds to the required handling time. The berthing position and berthing time of a vessel are given in the lower-left vertex of a rectangle, denoted by *the reference point* of a vessel (marked by the index of vessel in Fig. 3). A berthing plan is *feasible* if $a)$ the rectangles do not overlap and $b)$ all rectangles can fit in the given space-time-diagram (see Fig. 3).

In order to define DMCHBAP, we start from the definition of static MCHBAP given in [24] and adapt it to the dynamic version of BAP. DMCHBAP is characterized by the input data, objective function, and a set of constraints defining feasible solutions. The input data of DMCHBAP are listed below:

l: Total number of vessels;

m: Total number of berthing positions;

T: Total number of time units in the planning horizon (corresponding to $T-1$ time segments);

vessels: Sequence of data describing all l vessels, where $vessels = \{vessel_k : k = 1,\ldots,l\}$;

$vessels_k$: 8-tuple with the following structure

$$vessel_k = (ETA_k, a_k, b_k, d_k, s_k, c_{1k}, c_{2k}, c_{3k}).$$

The elements of a 8-tuple $vessel_k$ represent the following data for each vessel:

ETA_k: The expected time of arrival of $vessel_k$;

a_k: Processing time of $vessel_k$, if single crane is used;

b_k: Length of $vessel_k$ expressed by the number of berths;

d_k: Required departure time of $vessel_k$;

s_k: Least-cost berthing location of the reference point of $vessel_k$;

c_{1k}: Penalty cost, if $vessel_k$ cannot dock at its preferred berth;

c_{2k}: Penalty cost per unit time, if $vessel_k$ cannot berth at ETA_k;

c_{3k}: Penalty cost per unit time, if $vessel_k$ is delayed beyond the required departure time d_k.

A feasible solution of DMCHBAP consists of pairs (B_k, At_k), $k = 1,2,\ldots,l$ where $B_k \in \{1,2,\ldots,m\}$ denotes the lowest berth index allocated to the $vessel_k$ and $At_k \in \{1,2,\ldots,T-1\}$ represents the minimum time index of a $vessel_k$. Pair (B_k, At_k) actually corresponds to the reference point of $vessel_k$, $k = 1,2,\ldots,l$. A feasible solution of DMCHBAP is subject to the following sets of constraints:

Constraints 1. Each berth can be assigned to only one vessel at time segment t, $t = 1,\ldots,T-1$;

Constraints 2. A berth can be allocated to a vessel only between its arrival and departure times.

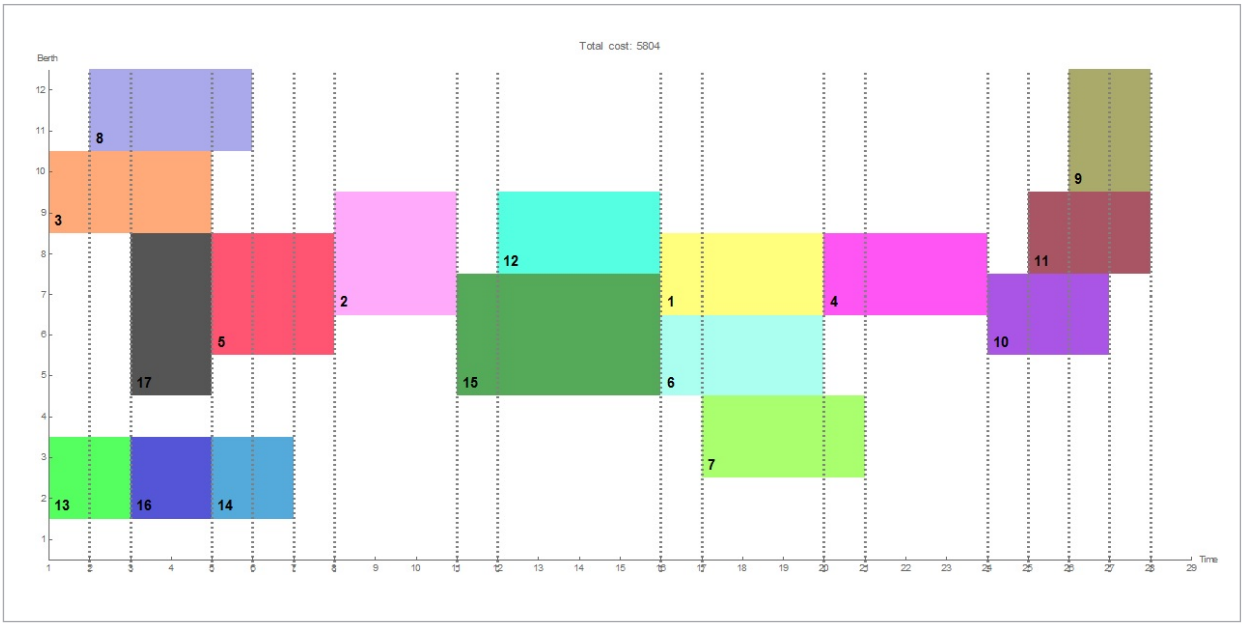
The goal of DMCHBAP is to minimize the total penalty cost including: the penalty incurred as a result of missing the preferred berthing location of the reference point, the penalty resulted by the actual berthing later than the expected arrival time, and the penalty cost arising from the delay of the departure after the required due time. The last two terms influence the objective function only in case they are positive. Similarly to [40], the objective function can be expressed as:

$$\sum_{k=1}^{l}\left(c_{1k}\sigma_k + c_{2k}(At_k - ETA_k)^+ + c_{3k}(Dt_k - d_k)^+\right), \quad (1)$$

where

$$\sigma_k = \sum_{t=1}^{T}\sum_{i=1}^{m}\left\{|i - s_k| : vessel\ k\ occupies\ position\ (t,i)\right\}, \quad (2)$$

Figure 1
Illustration of BAP solution



$$(a-b)^{+}=\begin{cases}a-b, & if\ a>b,\\ 0, & otherwise,\end{cases} \quad (3)$$

and $Dt_k = At_k + \lceil a_k / b_k \rceil$, $Dt_k \in \{1,2,\ldots,T\}$, representing the departure time of the $vessel_k$. Namely, if only one crane is used to serve the $vessel_k$, the required processing time is a_k. However, if the $vessel_k$ occupies more than one berth (each equipped by a crane), the processing time will be reduced.

According to the definition of the objective function given in [40], σ_k is expressed by the given double sum (2). It can be explained as follows: as the $vessel_k$ can occupy several berths and only one is preferred (which usually means that it contains the required equipment for serving the vessel), all other allocated berths have to be penalized. The lack of a proper equipment on these berths requires the engagement of additional equipment and/or labor. For these reasons, the cost of handling a vessel may increase. Finally, $(a-b)^{+}$ denotes that this term has impact on the objective function value only if its value is positive.

Note that DMCHBAP is strongly NP-hard, because it can be observed as a machine scheduling problem [9, 38]. Actually, we prove that even the simpler variant of DMHCBAP is strongly NP-hard.

Theorem 1. *DMCHBAP is strongly NP-hard even in the restricted case, in which the preferred berth restrictions are ignored and the objective function represents only total weighted tardiness.*

Proof. In order to prove this theorem, we show that the well-known identical machine scheduling problem with release dates and minimization of total weighted tardiness can be polynomially reduced to DMCHBAP. In the standard three field scheduling notation [9], the problem can be classified as $P_m \mid r_j \mid \sum w_j T_j$. In order to follow previously given notations, we rename index j to k. We further assume that vessels represent jobs and berths correspond to the machines. Consequently, vessel's processing time defined by $\lceil a_k / b_k \rceil$ can be observed as job processing time p_k, while expected arrival time ETA_k corresponds to the job release time r_k. In addition, job's due date is actually vessel's due date d_k and the tardiness is calculated in the standard way, assuming that $w_k = c_{3k}$. To ignore preferred berthing location and waiting time of vessel, the corresponding penalties c_{1k} and c_{2k} are set to zero. The obtained machine scheduling problem is known to be strongly NP-hard, even for $m = 1$ (see [38]).

4. VNS-Based Metaheuristics for DMCHBAP

VNS is a simple and effective metaheuristic method based on local search procedure [20, 34]. The basic idea of VNS is the systematic change of neighborhoods within a descent phase, to find a local optimum, and within a perturbation phase to escape from the corresponding valley. The main components of VNS are *shaking* and *local search*. The role of shaking is to help the algorithm to escape from a local optimum and to explore the search space in an efficient manner. Local search is used to intensify exploration of neighborhoods around promising solutions. Different variants of VNS have been proposed in the literature up to now: Basic VNS (BVNS), Reduced VNS (RVNS), Variable Neighborhood Descent (VND), Variable Neighborhood Decomposition Search (VNDS), General VNS (GVNS), Skewed VNS (SVNS), Primal-dual VNS, Parallel VNS, etc. An overview of VNS-based methods and applications to combinatorial and global continuous optimization problems can be found in [20].

The study [6] considers static MCHBAP and proposes a new optimization method based on the deterministic variant of VNS method, known as Variable Neighborhood Descent (VND). This variant does not involve shaking component and the local search is performed through multiple neighborhoods. General Variable Neighborhood Search (GVNS) for the static MCHBAP is proposed in [26] to allow better diversification of solutions. GVNS involves shaking phase that helps the algorithm to escape from the local minimum and to efficiently explore the search space. In addition, instead of simple local search, GVNS uses VND. In this paper, we modify VND and GVNS approaches from [6] and [26], and develop VND and GVNS implementations for DMCHBAP. In order to provide better diversification of initial solutions, we propose Multi-Start variant of VND. In addition, we develop Skewed VNS (SVNS) as a solution approach to DMCHBAP. The basic idea behind SVNS is to address the problem of exploring valleys far from the incumbent solution. This is performed by accepting local optimum with objective function values close to the objective value of incumbent solution, but not necessarily better (see [20]).

The detailed description of other aspects of the proposed VND, MS-VND, GVNS, and SVNS implementations for DMCHBAP is provided in the following subsections.

4.1. Basic Definitions and Solution Representation

All four VNS methods proposed in this study are based on the combinatorial formulation for DMCHBAP and use the same data structures and initialization (preprocessing) phase as in [6] and [26]. As a part of the initialization, for all vessels and for all possible vessel positions in two dimensional plane, the list of 3-tuples elements (berth, time, penalty cost) is created. This list is denoted by Ψ and it consists of l individual ξ lists, each corresponding to one vessel. The ξ lists are sorted in non-decreasing order according to the penalty cost values for each vessel individually. The role of ξ lists is to ensure efficient search of the solution space by its significant reduction in each step. More precisely, any change in the allocation of vessels produces changes in the corresponding ξ lists in such a way that, for each vessel, only feasible positions remain as the elements in its ξ list. The fact that the ξ lists remain sorted at each step of the search, makes it easy to detect positions with smaller penalty costs for each allocated vessel (if any exists). In addition, Ψ list structure enables easy identification of unfeasible solution: if at any moment there exists a vessel with empty ξ list, the corresponding solution could be discarded as unfeasible.

During the execution of VNS-based methods for DMCHBAP, two main decisions are to be made: the selection of a vessel to be allocated and the selection of its position in berth-time plane. In the initialization phase of algorithms, these decisions are made stochastically, based on the priorities of vessels.

All four VNS approaches proposed for DMCHBAP use the same solution representation based on *sequence pair*, which was introduced in [35]. It involves two types of permutations, denoted by H and V that describe the positions of vessels in the port. These permutations are formed based on the following rules:

- **a** if vessel j precedes vessel i in the permutation H, then vessel j "cannot see" vessel i on "left-up" view,
- **b** if vessel j precedes vessel i in the permutation V, then j "cannot see" i on "left-down" view.

For each of the implemented VNS-based methods for DMCHBAP, global variables are: current solution (*Solution*), local improvement of the shaken current solution (*LocalBest*), the best found solution (*GlobalBest*) and the CPU time of the first occurrence of the best found solution (*minT*).

4.2. VND for DMCHBAP

Each vessel allocation may be uniquely represented as a pair of permutations (H,V). On the other hand, each pair (H,V) corresponds to a class of allocations. Therefore, the study [6] introduces a procedure that efficiently finds a feasible allocation that minimizes total cost while preserving (H,V) ordering in the case of static MCHBAP.

Algorithm 1 Variable Neighborhood Descent algorithm

```
procedure VND(vessels, k_max)
    Solution ← InitialSolution(vessels)
    (H, V) ← Permutations(Solution)
    ωS ← NotPreferredPosition(Solution)
    k ← 1
    while k ≤ k_max do
        noImpr ← True
        ChangePositionH(ωS, k)
        ChangePositionV(ωS, k)
        ChangePositionHV(ωS, k)
        if noImpr then
            k ← k + 1
        else
            k ← 1
        end if
    end while
end procedure
```

The pseudo-code of the VND for DMCHBAP is presented in Alg. 4.2. The procedure used for generating initial solution for VND and GVNS from [6] and [26], respectively, cannot be applied to DMCHBAP, as this procedure often creates infeasible solution. For this reason, we propose two new methods for generating initial solution for DMCHBAP, which are incorporated in the procedure InitialSolution. Both methods start with creating the subsets of conflicting vessels, based on their most preferred berths and *ETA* parameter values. In the first method, groups are sorted in non-increasing order of their cardinality. One by one, groups are allocated such that total cost of each group is minimized. If no feasible solution is obtained, the second method is applied. The groups of conflicting vessels are now sorted in non-decreasing order based on average number of feasible positions for vessels belonging to the same group. The allocation of each group is followed by updating the feasible positions in ξ lists of the remaining vessels and resorting the non-allocated groups.

Based on *Solution* obtained by procedure InitialSolution, the corresponding pair (H,V) is formed by pro-

cedure PERMUTATIONS. The group of vessels ωS that are not placed on their most preferred positions is identified. The vessels belonging to ωS are sorted in non-increasing order of their costs with respect to the current best solution. During the algorithm's run, the content of set ωS may change, however, its elements are always sorted according to the corresponding costs.

As in [6], VND uses three types of neighborhoods, which are applied only to the vessels from ωS. The algorithm always starts from the vessel in ωS having the largest cost and continues with vessels having smaller costs. For a given size k, $k = 1, 2, 3, ...k_{max}$, the neighborhoods are explored in the following order:

1. CHANGEPOSITIONH: selected vessel is first moved k positions to the left in permutation H, and if there is no improvement, the same vessel is moved k positions to the right in H, while permutation V remains unchanged;
2. CHANGEPOSITIONV: selected vessel is first moved k positions to the left in permutation V, and if there is no improvement, the same vessel is moved k positions to the right in V, while permutation H remains unchanged;
3. CHANGEPOSITIONHV: represents a combination of CHANGEPOSITIONH and CHANGEPOSITIONV, where all possible changes of H and V are considered.

4.3. MS-VND for DMCHBAP

At the beginning of each iteration of MS-VND, the initial solution is constructed by procedure INITIALIZE. This procedure starts with an empty solution and constructs a complete initial solution. In order to generate it, the procedure must make two decisions: to choose a vessel and to choose allowable position from ξ list for the selected vessel.

Algorithm 2 Multi-Start Variable Neighborhood Descent algorithm

```
procedure MS-VND(vessels, k_max, RunTime, λ1, λ2, λ3)
    while SESSIONTIME() ≤ RunTime do
        Solution ← INITIALIZE(vessels, λ1, λ2, λ3)
        (H, V) ← PERMUTATIONS(Solution)
        ωS ← NOTPREFERREDPOSITION(Solution)
        VND1((H, V), ωS, k_max)
    end while
end procedure
```

The criterion for vessel selection in MS-VND is a linear combination of ETA parameter, the size of the corresponding rectangle associated to the vessel in two dimensional plane, and the calculated average cost of all possible ξ list elements for the observed vessel. Coefficients of these three parameters are denoted by λ_1, λ_2, and λ_3, respectively. The linear combination of parameters with the corresponding coefficients represents the priority of a vessel used for its selection by the roulette wheel. The values of coefficients λ_i are determined experimentally in such a way that $\lambda_1 + \lambda_2 + \lambda_3 = 1$ holds.

For a selected vessel, all potential positions are considered with respect to the penalty cost value. The selection of vessels' positions in the initialization phase of MS-VND is performed stochastically, based on the position costs. The positions with smaller costs have higher chances to be selected by the roulette wheel. Procedure INITIALIZE calculates probabilities for all feasible positions from ξ list and selects a position by using randomly generated number and the roulette wheel. The considered vessel is fixed on that position and ξ lists are reduced for all unused vessels. The above described process is repeated l times.

From the constructed initial solution, algorithm forms permutation pair (H, V) and the group of vessels ωS. These parameters are further passed as input data to algorithm VND1, which is similar to VND described in Subsection 4.2. The only difference is that procedures PERMUTATIONS and NOTPREFERREDPOSITION are called outside it and additional input parameters $((H, V), \omega S)$ are passed to VND1. These steps are repeated until stopping criterion is satisfied (predefined amount of running time $RunTime$). The best found solution through multiple VND runs is returned as the output of MS-VND. The pseudo-code of MS-VND for DMCHBAP is presented in Alg. 4.3.

4.4. GVNS for DMCHBAP

The initial solution and pair (H, V) are generated in the same way as in VND. GVNS employs SHAKE procedure based on stochastic transformations of the current best solution. During the shaking step, two transformations are applied in order to form a new solution:

1. First, k random groups of vessels are chosen in accordance with the priority proportional to the total cost of the group. The chosen groups are placed at the beginning of the list containing all the groups.

2 Next, k random pairs of vessels are selected based on the calculated priority, where the priority is proportional to the vessel's cost in the current best solution. The locations of the selected pairs of vessels are swapped. For this transformation, vessels do not need to be in the same group.

Let (H_1,V_1) be a neighbor solution obtained from Shake phase. Instead of simple local search, GVNS employs VND to find improvements of the solution (H_1,V_1), by systematic exploration of its six neighborhoods. These neighborhoods are changing H or V permutation, or both permutations simultaneously, with an aim to find the optimal sequence pair. The six procedures for exploring neighborhoods are applied in the following order:

Algorithm 3 General Variable Neighborhood Search algorithm

```
procedure GVNS(vessels, k_max, RunTime)
    {Solution, groups} ← InitialSolution(vessels)
    GlobalBest ← Solution
    (H, V) ← Permutations(Solution)
    k ← 1
    while SessionTime() ≤ RunTime do
        {H_1, V_1, groups_1} ← Shake(vessels, groups, k)
        noImpr ← True
        noImpr ← SingleSwapH(noImpr)
        noImpr ← SingleSwapV(noImpr)
        noImpr ← SingleMoveH(noImpr)
        noImpr ← SingleMoveV(noImpr)
        noImpr ← DoubleSwapHV(noImpr)
        noImpr ← DoubleMoveHV(noImpr)
        if noImpr then
            k ← k + 1
            if k > k_max then
                k ← 1
            end if
        else
            k ← 1
        end if
    end while
end procedure
```

1 SingleSwapH selects two vessels and exchanges their positions in permutation H, leaving permutation V unmodified;

2 SingleSwapV selects two vessels and swaps their positions in permutation V, while permutation H is unchanged;

3 SingleMoveH selects two vessels v_i and v_j and moves vessel v_j immediately after vessel v_i in permutation H, no matter if the vessel v_i is in front of or behind vessel v_j in the current permutation;

4 SingleMoveV performs moving of selected vessel v_j immediately after vessel v_i in permutation V, regardless of mutual order of vessels v_i and v_j;

5 DoubleSwapHV consists of all neighbors obtained by one SingleSwapH and one SingleSwapV that are not necessarily performed on the same pair of vessels;

6 DoubleMoveHV is performed by one SingleMoveH and one SingleMoveV that are not necessarily applied on the same pair of vessels.

GVNS incorporates the *first improvement* principle. More precisely, when an improvement of the current solution is found in one of the six neighborhoods, the remaining neighborhoods are skipped and this is controlled by *noImpr* variable. As soon as *noImpr* gets value *false*, global variables *Solution* and *GlobalBest* are updated inside the corresponding local search procedure. Each procedure of local search starts by examining the value of *noImpr* variable. The *false* value will prevent the execution of the corresponding local search procedure. If *GlobalBest* is improved, the neighborhood counter k is reset to 1. The value of parameter k_{max} represents the maximal neighborhood size for shaking. GVNS algorithm finishes when the stopping criterion (predefined amount of running time) is satisfied. The described steps of GVNS for DMCHBAP are presented in Alg. 4.4.

4.5. SVNS for DMCHBAP

In cases when a high-quality local optimum has been found, VNS will most likely lead the search at random to far-away neighborhoods. As the size of neighborhoods to be explored increases, VNS has a tendency to degenerate into a multi-start heuristic. In order to address the problem of getting out of very large neighborhoods, Hansen and Mladenović [18] proposed a variant of VNS, denoted by Skewed Variable Neighborhood Search (SVNS). The main idea behind SVNS is to accept a local optimum even if it is slightly worse than the incumbent solution. The strategy applied in SVNS helps in solving problem instances having several separated and possibly distant neighborhoods containing near-optimal solutions [18, 20].

This was our motivation to develop SVNS as an additional VNS approach to DMCHBAP. The pseudocode of our SVNS algorithm is presented in Alg. 4.5. The structure of SVNS for DMCHBAP is similar to the

Algorithm 4 Skewed Variable Neighborhood Search algorithm

```
procedure SVNS(vessels, k_max, α, RunTime)
    {Solution, groups} ← INITIALSOLUTION(vessels)
    GlobalBest ← Solution
    while SESSIONTIME() ≤ RunTime do
        (H, V) ← PERMUTATIONS(Solution)
        k ← 1
        while k ≤ k_max do
            {H_1, V_1, groups_1} ← SHAKE(vessels, groups, k)
            noImpr ← True
            noImpr ← SINGLESWAPH(noImpr)
            noImpr ← SINGLESWAPV(noImpr)
            noImpr ← SINGLEMOVEH(noImpr)
            noImpr ← SINGLEMOVEV(noImpr)
            noImpr ← DOUBLESWAPHV(noImpr)
            noImpr ← DOUBLEMOVEHV(noImpr)
            if COST(LocalBest) − α|COST(LocalBest)
               −COST(Solution)|<COST(Solution) then
               Solution ← LocalBest
               k ← 1
            else
               if noImpr then
                  k ← k + 1
               else
                  k ← 1
               end if
            end if
        end while
        Solution ← GlobalBest
    end while
end procedure
```

structure of GVNS described in Section 4.4. As SVNS allows the acceptance of worse solutions compared to the current best one, it was necessary to activate the global variable *LocalBest*. The initially generated solution is placed in the variable *Solution*, which is further subject to SHAKE procedure. Local Search phase tries to improve the newly produced solution by exploring each of the six neighborhoods in the order indicated in Alg. 4.5. At the end of Local Search phase, the obtained solution is saved in the *LocalBest* variable. The difference between GVNS and SVNS is in the acceptance criterion of the local optimum *LocalBest* produced by VND phase. SVNS is more tolerant in accepting local optimum *LocalBest* that does not improve the current *Solution* regarding the objective function value. The level of tolerance is controlled by parameter $\alpha > 0$. Whenever $Cost(LocalBest) - \alpha \,|\, Cost(LocalBest) - Cost(Solution) \,|$ is less than $Cost(Solution)$, the search resumes from *LocalBest* by setting $k = 1$. Obviously, it is necessary to keep the information about the global best solution *GlobalBest* and to update it whenever an improvement is achieved. As we already explained, this is done within the corresponding local search procedure.

5 Experimental Results

In general, there is a lack of benchmark instances for BAPs in literature. Therefore, we generated test examples randomly, but systematically, following the idea from [12]. Metaheuristics VND, MS-VND, GVNS, and SVNS developed for DMCHBAP are evaluated on four generated data sets and the obtained results are analyzed. Data sets used in our experimental study are available online at http://www.mi.sanu.ac.rs/tanjad/DMCHBAP.htm. Each of them contains randomly generated instances characterized by the following parameters:

- the first data set: $l = 10,15$ vessels, $m = 8$ berths, the time horizon of $T = 15$ units, $l = 20$ vessels, $m = 8$ berths, the time horizon of $T = 20$ units, and $l = 25$ vessels, $m = 8$ berths, the time horizon of $T = 25$ units;
- the second data set: $l = 35,40,45$ vessels, $m = 8$ berths, and the time horizon of $T = 112$ units;
- the third data set: contains randomly generated instances involving $l = 50,55,60$ vessels in the case of $m = 13$ berths and $T = 112$ time units;
- he fourth data set: $l = 70,80,90,100$ vessels, $m = 13$ berths, and the time horizon of $T = 112$ units.

The data used to specify various types of vessels are presented in Table 1 taken from [33]. The set of test instances involves three types of vessels: *feeder*, *medium*, and *mega*. For each type, the corresponding percentage of test instances, handling time range, penalty amounts (in units of US$ 1000) and number of berths occupied by specific type of vessels are listed in Table 1. The distribution of the least-cost berthing location for vessels is homogeneous.

Table 1
Vessel specifications for generated test instances

Vessel type	Popul.%	Time range	C1	C2	C3	C4	NBerths
Feeder	60%	1 to 3	2	3	3	9	1
Medium	30%	4 to 5	3	6	6	18	2
Mega	10%	6 to 8	4	9	9	27	3

In order to obtain optimal solutions for small size problem instances we adapted MILP model, proposed in [7] for MCHBAP (the static variant of Minimum

Cost Hybrid BAP), to the considered DMCHBAP. We have executed the obtained MILP model within the framework of commercial CPLEX solver, version 12.3, which was run on the same configuration as the one used for metaheuristic methods.

All four VNS metaheuristic approaches are coded in the *Wolfram Mathematica v8.0* programming language. It is important to note that, unlike classical programming languages, *Mathematica* interprets instructions and therefore, the running times of algorithms may increase. However, our comparison is fair, having in mind that all VNS-based algorithms are executed under the same conditions. All computational experiments with CPLEX, VND, MS-VND, GVNS, and SVNS were conducted on the same platform, i.e., on a computer with an *Intel Pentium 4* 3.00 GHz CPU and 512 MB of RAM, running the *Microsoft Windows XP Professional Version 2002 Service Pack 2* operating system. Note that executable version of CPLEX 12.3 is optimized for this platform, meaning that it is favored with respect to other algorithms.

Having in mind that metaheuristics are stochastic methods, their stability is examined by performing repeated runs on each instance. In our computational experiments, MS-VND, GVNS, and SVNS methods where executed 10 times with time limit of 10 minutes for all test examples, i.e., variable $RunTime$ is set to 10 minutes. VND is deterministic in nature, and, therefore, it was run only once on each tested instance.

Preliminary computational experiments are performed on the subset of test instances in order to determine appropriate parameter values for each of the considered VNS-based approaches. Table 2 shows the list of parameter values for each metaheuristic that led to its best performance.

We have also developed several variants of SWO adapted to the DMCHBAP, following the ideas presented in [47]. Each of the implemented SWO variants starts from a feasible initial solution and dynamically changes it based on vessels' priorities. Vessels with larger allocation cost in current solution have higher priority to be chosen for the next allocation. Once feasible solution is produced, all vessels are sorted according to their cost in decreasing order and new allocation starts. One by one, vessels are allocated in port on randomly chosen feasible position by roulette wheel. If allocation leads to an unfeasible solution, a new random solution is generated. In the case that all vessels are allocated in port, i.e., solution is complete and feasible, new vessels' priorities are calculated and SWO algorithm performs the next iteration. The imposed stopping criterion for each SWO variant is 10 minutes of running time. Implemented SWO approaches for DMCHBAP are also coded in the *Wolfram Mathematica v8.0* and executed on the same platform as VNS-based methods. On each instance, SWO was run 10 times. In the rest of this section, we present only the results of the best performing SWO variant.

Table 2
Parameter specifications

Metaheur.	Parameter settings					
	k_{max}	$RunTime$	α	λ_1	λ_2	λ_3
VND	l	-	-	-	-	-
MS-VND	l	10 minutes	-	0.8	0	0.2
GVNS	l	10 minutes	-	-	-	-
SVNS	l	10 minutes	0.25	-	-	-

Tables 3 and 4 contain the comparison of results obtained by CPLEX solver and considered metaheuristic methods on the first data set that includes small size problem instances. The first column of Table 3, denoted by *Class* contains instance's specification, given in the form $m \times T - l$, where m represents the number of berths, T indicates the number of time units in the planning horizon, and l stands for the number of vessels. The second column contains the identification number (index) of each instance in the corresponding class. The next two columns are related to the results of CPLEX solver, containing objective function value of the optimal solution OPT and the corresponding running time, denoted by *Time* and given in seconds. In the next four columns, results related to the best performing SWO variant are presented. In the column named *Best*, the best found total cost (obtained after 10 SWO executions) is given, while the average total cost $AvgC$ and average minimum CPU time $AvgT$ (out of 10 runs) are presented in the next two columns. In order to measure the quality of the obtained SWO results, in column *G%*, we present the average gap calculated as $100 \cdot \frac{AvgC - OPT}{OPT}$. The next three columns contain results related to the VND metaheuristic. The column named *Best* contains the best found total cost. Column *Time* indicates the

running time (in seconds) required by VND to obtain its best solution. The gap *G%* for VND is calculated as $100 \cdot \frac{Best - OPT}{OPT}$. The results of MS-VND, GVNS, and SVNS in Table 3 are given in the same way as in the case of SWO. In order to highlight the best performing method with respect to the solution quality, the best-known (optimal) solutions for each instance are bolded in Table 3. Similarly, for the best performing method with respect to CPU time, the shortest (average) CPU times for each instance are bolded in Table 3.

From the results presented in Table 3, it can be seen that all four VNS-based methods reach optimal solutions provided by CPLEX solver on each of small size problem instances. SVNS shows the best stability, as its average percentage gap is 0%, meaning that SVNS reached optimal solution in all 10 runs for each instance. The values presented in column *G%* indicate that MS-VND, and GVNS methods also showed to be stable, as the corresponding average percentage gaps are 0.03%, and 0.09%, respectively. SWO method evinced poor performance on the set of small size instances, as it provided solutions that are quite far from the optimal ones for each considered instance. The best performing SWO variant produces solutions with average percentage gap of 79.68% from the optimal ones.

Regarding average running times, GVNS was the fastest in returning its best solutions, followed by VND, SVNS, and MS-VND, while SWO was the slowest method. However, all five methods were significantly faster compared to CPLEX solver, which needed 2436.81 seconds (on average) to produce optimal solutions for all instances in the set. The average running times of the proposed VNS-based methods were: 15.74 *s* for GVNS, 19.58 *s* for VND, 25.44 *s* for SVNS, and 77.34 *s* for MS-VND, while SWO required 241.40 *s* (on average) to return its best solutions. This implies that the proposed GVNS was more than 154 times faster compared to CPLEX, and 1.24, 1.62, 4.91, and 15.34 times faster than VND, SVNS, MS-VND, and SWO, respectively.

As it can be seen from Table 3, even the best variant of SWO did not produce satisfactory results regarding solution quality and running times. Therefore, SWO is excluded from detailed computational experiments on the other data sets.

Table 4 contains the comparison of results obtained by CPLEX solver and the proposed VNS-based metaheuristic methods on the largest instances from the

Table 3

Computational results of CPLEX, SWO, and VNS-based metaheuristics on instances from the first data set (m = 8, T= 15, 20, l = 10, 15, 20)

		CPLEX		SWO				VND			MS-VND				GVNS				SVNS			
Class	*i*	*OPT*	*Time*	*Best*	*AvgC*	*AvgT*	*G%*	*Best*	*Time*	*G%*	*Best*	*AvgC*	*AvgT*	*G%*	*Best*	*AvgC*	*AvgT*	*G%*	*Best*	*AvgC*	*AvgT*	*G%*
8x15-10	1	369	40.03	465	483.9	143.52	31.14	**369**	97.31	0.00	**369**	**369**	97.25	0.00	**369**	**369**	9.86	0.00	**369**	**369**	**9.66**	0.00
	2	208	10.90	291	332.1	270.82	59.66	**208**	0.10	0.00	**208**	**208**	0.10	0.00	**208**	**208**	0.02	0.00	**208**	**208**	**0.01**	0.00
	3	188	18.19	303	332.9	171.51	77.07	**188**	13.69	0.00	**188**	**188**	13.82	0.00	**188**	**188**	**1.39**	0.00	**188**	**188**	1.41	0.00
	4	180	18.36	340	398.5	293.66	121.39	**180**	0.28	0.00	**180**	**180**	0.26	0.00	**180**	**180**	**0.03**	0.00	**180**	**180**	**0.03**	0.00
	5	225	12.48	360	396.7	227.04	76.31	**225**	0.12	0.00	**225**	**225**	0.12	0.00	**225**	**225**	**0.02**	0.00	**225**	**225**	**0.02**	0.00
8x15-15	1	360	1402.45	614	641.5	169.84	78.19	**360**	**2.95**	0.00	**360**	**360**	104.26	0.00	**360**	**360**	36.67	0.00	**360**	**360**	54.13	0.00
	2	269	486.57	511	576.5	233.45	114.31	**269**	**0.83**	0.00	**269**	**269**	28.20	0.00	**269**	**269**	12.68	0.00	**269**	**269**	12.06	0.00
	3	362	203.32	527	551	222.75	52.21	**362**	32.00	0.00	**362**	**362**	55.21	0.00	**362**	**362**	18.40	0.00	**362**	**362**	**13.85**	0.00
	4	404	547.02	624	680.1	183.04	68.34	**404**	**1.22**	0.00	**404**	**404**	26.58	0.00	**404**	**404**	21.95	0.00	**404**	**404**	37.94	0.00
	5	553	1500.74	719	790.8	227.08	43.00	**553**	5.14	0.00	**553**	**553**	3.62	0.00	**553**	**553**	11.08	0.00	**553**	**553**	**2.69**	0.00
8x20-20	1	565	4323.16	856	986.2	275.29	74.55	**565**	5.22	0.00	**565**	**565**	5.13	0.00	**565**	**565**	0.54	0.00	**565**	**565**	**0.53**	0.00
	2	416	5391.39	731	850.9	362.84	104.54	**416**	**6.81**	0.00	**416**	**416**	93.05	0.00	**416**	**416**	25.31	0.00	**416**	**416**	37.55	0.00
	3	392	4732.67	799	862.1	292.24	119.92	**392**	**23.81**	0.00	**392**	392.6	266.54	0.15	**392**	**392**	43.38	0.00	**392**	**392**	53.53	0.00
	4	509	3214.67	923	958.8	251.23	88.37	**509**	84.38	0.00	**509**	509.1	177.48	0.02	**509**	**509**	**13.38**	0.00	**509**	**509**	18.97	0.00
	5	600	14650.23	1034	1117.5	296.70	86.25	**600**	**19.84**	0.00	**600**	602	288.54	0.33	**600**	608.4	41.34	1.40	**600**	**600**	139.15	0.00
average:		373.33	2436.81	606.47	663.97	241.40	79.68	**373.33**	19.58	0.00	**373.33**	373.51	77.34	0.03	**373.33**	373.89	**15.74**	0.09	**373.33**	**373.33**	25.44	0.00

Table 4

Computational results of CPLEX and VNS-based metaheuristics on instances from the first data set (m = 8, T= 25, l = 25)

			CPLEX		VND			MS-VND				GVNS				SVNS			
Class	*i*	*BK*	*LB*	*UB*	*Best*	*Time*	*G%*	*Best*	*AvgC*	*AvgT*	*G%*	*Best*	*AvgC*	*AvgT*	*G%*	*Best*	*AvgC*	*AvgT*	*G%*
8x25-25	1	505	395	534	**505**	49.93	0.00	**505**	**505.00**	115.64	0.00	**505**	**505.00**	**7.16**	0.00	**505**	**505.00**	45.28	0.00
	2	645	349	887	646	**43.59**	0.16	**645**	660.80	285.09	2.45	646	657.00	181.34	1.86	**645**	647.20	137.45	0.34
	3	395	346	395	**395**	12.26	0.00	**395**	**395.00**	**6.23**	0.00	**395**	**395.00**	88.41	0.00	**395**	**395.00**	27.76	0.00
	4	474	376	485	**474**	**41.25**	0.00	**474**	481.70	114.96	1.62	**474**	494.40	92.58	4.30	**474**	479.70	213.03	1.20
	5	508	318	667	510	**6.71**	0.39	**508**	513.8	371.24	1.14	**508**	538.1	147.87	5.93	**508**	**512.1**	223.28	0.81
average:		505.40	356.80	593.60	506.00	**30.75**	0.11	**505.40**	511.26	178.63	1.04	505.60	517.90	103.47	2.42	**505.40**	**507.80**	129.36	0.47

first data set, with $m = 8$, $T = 25$, $l = 25$. For these instances we impose the time limit on CPLEX execution of 1 hour (having in mind that the decisions in the port are to be made very quick, on a minute basis). The first column of Table 4 contains instance's specification, while the second column, denoted as *BK*, presents the best-known objective function values, provided either by CPLEX or by the proposed VNS-based metaheuristics. The next two columns are related to the results of CPLEX solver, containing lower and upper bounds on the objective function value of the optimal solution. The rest of Table 4 is related to the results of the four proposed VNS methods, which are given in the same way as in Table 3.

The results presented in Table 4 show that all four VNS-based methods improved upper bounds provided by CPLEX, with the exception of one instance for which the best solutions of all four VNS methods coincide with the upper bound that CPLEX returned. Again, VND and SVNS showed the best stability, as their average percentage gap is 0.11% and 0.47%, respectively. In the case of MS-VND, and GVNS, the values of average percentage gaps were 1.04% and 2.42%, respectively, indicating that these two VNS-based methods also have good stability. On average, VND was the fastest method, followed by GVNS, SVNS, and MS-VND. The average running times of the proposed VNS-based methods were: 30.75 *s* for VND, 103.47 *s* for GVNS, 129.36 *s* for SVNS, and 178.63 *s* for MS-VND, implying that the proposed VND was 3.36, 4.21, and 5.81 times faster than GVNS, SVNS, and MS-VND, respectively.

Tables 5 and 6 show the results obtained by the proposed VNS-based approaches to DMCHBAP on the

Table 5

Computational results of VNS-based metaheuristics on instances from the second data set (m = 8, T= 112, l = 35, 40, 45)

			VND			MS-VND				GVNS				SVNS			
l	*i*	*BK*	*Best*	*Time*	*G%*	*Best*	*AvgC*	*AvgT*	*G%*	*Best*	*AvgC*	*AvgT*	*G%*	*Best*	*AvgC*	*AvgT*	*G%*
35	1	527	**527**	8.25	0.00	**527**	**527**	**3.53**	0.00	**527**	**527**	11.23	0.00	**527**	**527**	5.41	0.00
	2	711	**711**	0.66	0.00	**711**	**711**	0.27	0.00	**711**	**711**	0.15	0.00	**711**	**711**	**0.08**	0.00
	3	973	**973**	0.63	0.00	**973**	**973**	0.25	0.00	**973**	**973**	0.15	0.00	**973**	**973**	**0.08**	0.00
	4	566	**566**	0.58	0.00	**566**	**566**	0.24	0.00	**566**	**566**	0.15	0.00	**566**	**566**	**0.08**	0.00
	5	536	**536**	6.97	0.00	**536**	**536**	2.99	0.00	**536**	**536**	0.79	0.00	**536**	**536**	**0.38**	0.00
	6	528	**528**	0.58	0.00	**528**	**528**	0.26	0.00	**528**	**528**	0.15	0.00	**528**	**528**	**0.08**	0.00
	7	757	787	3.83	3.96	787	787	1.62	3.96	784	786.7	**0.48**	3.92	**757**	**773.4**	169.04	2.17
	8	567	583	0.63	2.82	598	598	0.26	5.47	583	583	**0.16**	2.82	**567**	**572.6**	81.77	0.99
	9	957	**957**	0.69	0.00	**957**	**957**	0.28	0.00	**957**	**957**	0.16	0.00	**957**	**957**	**0.08**	0.00
	10	818	831	1.34	1.59	831	831	0.54	1.59	**818**	829.5	**0.20**	1.41	**818**	**821.3**	200.55	0.40
40	1	540	**540**	213.19	0.00	**540**	**540**	93.62	0.00	**540**	**540.3**	105.09	0.06	**540**	**540**	**48.33**	0.00
	2	577	**577**	0.80	0.00	**577**	**577**	0.34	0.00	**577**	**577**	0.18	0.00	**577**	**577**	**0.09**	0.00
	3	587	762	0.73	29.81	762	762	**0.31**	29.81	**587**	681.5	37.10	16.10	**587**	**594.1**	69.11	1.21
	4	874	920	0.53	5.26	920	920	**0.25**	5.26	888	910.9	0.26	4.22	**874**	**875.4**	159.17	0.16
	5	649	**649**	0.69	0.00	**649**	**649**	0.30	0.00	**649**	**649**	0.18	0.00	**649**	**649**	**0.10**	0.00
	6	852	915	0.84	7.39	915	915	0.40	7.39	873	903.2	**0.25**	6.01	**852**	**862.6**	88.32	1.24
	7	570	594	1.47	4.21	594	594	0.67	4.21	576	587.8	**0.50**	3.12	**570**	**574.2**	67.91	0.74
	8	655	**655**	0.84	0.00	**655**	**655**	0.38	0.00	**655**	**655**	0.21	0.00	**655**	**655**	**0.11**	0.00
	9	660	723	0.73	9.55	723	723	0.33	9.55	699	717.4	**0.22**	8.70	**660**	**680.4**	232.42	3.09
	10	967	**967**	1.02	0.00	**967**	**967**	0.46	0.00	**967**	**967**	0.22	0.00	**967**	**967**	**0.12**	0.00
45	1	693	**693**	1.17	0.00	**693**	**693**	0.52	0.00	**693**	**693**	0.22	0.00	**693**	**693**	**0.11**	0.00
	2	844	**844**	1.03	0.00	**844**	**844**	0.45	0.00	**844**	**844**	0.26	0.00	**844**	**844**	**0.12**	0.00
	3	848	857	1.20	1.06	857	857	0.54	1.06	**848**	854.3	**0.35**	0.74	**848**	**848.9**	176.12	0.11
	4	879	885	1.16	0.68	922	922	0.49	4.89	885	885	**0.29**	0.68	**879**	**882**	139.15	0.34
	5	602	656	0.94	8.97	656	656	**0.42**	8.97	623	636.9	0.63	5.80	**602**	**611.3**	236.51	1.54
	6	1123	1212	0.80	7.93	1212	1212	0.36	7.93	1212	1212	**0.24**	7.93	**1123**	**1203.1**	99.02	7.13
	7	761	786	1.06	3.29	786	786	0.45	3.29	770	782.6	**0.35**	2.84	**761**	**768.8**	248.18	1.02
	8	1045	**1045**	1.63	0.00	**1045**	**1045**	0.69	0.00	**1045**	**1045**	0.33	0.00	**1045**	**1045**	**0.16**	0.00
	9	824	834	1.34	1.21	834	834	0.56	1.21	834	834	**0.30**	1.21	**824**	**832.8**	75.42	1.07
	10	956	981	1.17	2.62	992	992	0.55	3.77	981	981	**0.30**	2.62	**956**	**981.9**	132.63	2.71
average:		748.2	769.8	8.55	3.01	771.9	771.9	**3.74**	3.28	757.6	765.1	5.37	2.27	**748.2**	**755.0**	74.35	0.80

Table 6

Computational results of VNS-based metaheuristics on instances from the third data set (m = 13, T= 112, l = 50, 55, 60)

			VND			MS-VND				GVNS				SVNS			
l	*i*	*BK*	*Best*	*Time*	*G%*	*Best*	*AvgC*	*AvgT*	*G%*	*Best*	*AvgC*	*AvgT*	*G%*	*Best*	*AvgC*	*AvgT*	*G%*
50	1	575	**575**	44.63	0.00	**575**	**575**	48.02	0.00	**575**	**575**	76.30	0.00	**575**	**575**	**26.63**	0.00
	2	677	**677**	**14.70**	0.00	**677**	**677**	16.85	0.00	**677**	**677**	33.54	0.00	**677**	**677**	17.03	0.00
	3	798	**798**	1.31	0.00	**798**	**798**	1.26	0.00	**798**	**798**	0.28	0.00	**798**	**798**	**0.15**	0.00
	4	532	**532**	1.42	0.00	**532**	**532**	1.35	0.00	**532**	**532**	0.28	0.00	**532**	**532**	**0.15**	0.00
	5	893	**893**	1.45	0.00	**893**	**893**	1.50	0.00	**893**	**893**	0.33	0.00	**893**	**893**	**0.17**	0.00
	6	504	530	**1.50**	5.16	530	530	1.59	5.16	**504**	511	25.15	1.39	**504**	**504**	198.51	0.00
	7	823	**823**	1.38	0.00	**823**	**823**	1.54	0.00	**823**	**823**	0.31	0.00	**823**	**823**	**0.16**	0.00
	8	688	689	1.36	0.15	689	689	1.47	0.15	**688**	688.9	**0.34**	0.13	**688**	**688**	79.20	0.00
	9	1005	1095	5.58	8.96	1095	1095	5.64	8.96	1009	1073	**1.09**	6.77	**1005**	**1024.6**	98.76	1.95
	10	926	**926**	1.34	0.00	**926**	**926**	1.35	0.00	**926**	**926**	**0.32**	0.00	**926**	926.9	28.64	0.10
55	1	634	**634**	1.56	0.00	**634**	**634**	1.58	0.00	**634**	**634**	0.30	0.00	**634**	**634**	**0.16**	0.00
	2	911	917	1.61	0.66	917	917	**1.60**	0.66	911	915.3	53.21	0.47	**911**	**911**	69.64	0.00
	3	1129	**1129**	8.13	0.00	**1129**	**1129**	8.19	0.00	**1129**	**1129**	1.10	0.00	**1129**	**1129**	**0.46**	0.00
	4	1033	**1033**	1.72	0.00	**1033**	**1033**	1.72	0.00	**1033**	**1033**	0.43	0.00	**1033**	**1033**	**0.18**	0.00
	5	1025	**1025**	2.17	0.00	**1025**	**1025**	2.09	0.00	**1025**	**1025**	0.50	0.00	**1025**	**1025**	**0.21**	0.00
	6	741	**741**	3.17	0.00	**741**	**741**	3.12	0.00	**741**	**741**	0.61	0.00	**741**	**741**	**0.26**	0.00
	7	881	**881**	1.75	0.00	**881**	**881**	1.69	0.00	**881**	**881**	0.45	0.00	**881**	**881**	**0.19**	0.00
	8	1047	1073	1.70	2.48	1073	1073	1.64	2.48	1073	1073	**0.45**	2.48	**1047**	**1070.4**	32.01	2.23
	9	935	944	3.06	0.96	944	944	2.97	0.96	**935**	941.9	**0.69**	0.74	**935**	**939.9**	73.58	0.52
	10	1050	1106	1.84	5.33	1106	1106	1.83	5.33	1106	1106	**0.44**	5.33	**1050**	**1085.1**	103.70	3.34
60	1	991	1003	253.69	1.21	1003	1003	119.42	1.21	**991**	**991**	115.15	0.00	**991**	**991**	**57.62**	0.00
	2	1167	1168	33.08	0.09	1168	1168	**13.25**	0.09	**1167**	1187.9	105.33	1.79	**1167**	**1167**	67.09	0.00
	3	783	**783**	2.42	0.00	**783**	**783**	0.91	0.00	**783**	**783**	0.42	0.00	**783**	**783**	**0.24**	0.00
	4	822	825	4.58	0.36	825	825	1.98	0.36	825	825	**0.67**	0.36	**822**	**824.7**	38.63	0.33
	5	1166	**1166**	4.70	0.00	**1166**	**1166**	2.02	0.00	**1166**	**1166**	0.69	0.00	**1166**	**1166**	**0.35**	0.00
	6	1143	**1143**	2.06	0.00	1201	1201	0.90	5.07	**1143**	**1143**	**0.47**	0.00	**1143**	1189.4	37.98	4.06
	7	732	**732**	2.09	0.00	**732**	**732**	0.95	0.00	**732**	**732**	0.50	0.00	**732**	**732**	**0.24**	0.00
	8	875	881	2.27	0.69	881	881	0.97	0.69	876	880	**0.71**	0.57	**875**	**878.7**	259.06	0.42
	9	978	1126	2.91	15.13	1126	1126	1.28	15.13	978	1081.5	**0.84**	10.58	**978**	**999.2**	148.42	2.17
	10	1311	1314	6.11	0.23	1314	1314	2.64	0.23	1314	1314	**0.89**	0.23	**1311**	**1313.4**	25.34	0.18
average:		892.5	905.4	13.84	1.38	907.3	907.3	**8.38**	1.55	895.6	902.7	14.06	1.03	**892.5**	**897.8**	45.49	0.51

second and third data set, respectively. These data sets contain randomly generated test instances of larger dimensions, unsolved to optimality by CPLEX solver. Therefore, Tables 5 and 6 present the comparison of results obtained by VND, MS-VND, GVNS, and SVNS. The first column of Table 5 contains the number of vessels l. The next column (with heading i) indicates the index of the considered instance, while the third column (named BK) refers to the best-known cost value. The results of VND, MS-VND, GVNS, and SVNS are presented in the same way as in Table 3. As optimal solution is not known, the gap $G\%$ for VND is calculated as $100 \cdot \frac{Best - BK}{BK}$. In the case of MS-VND, GVNS, and SVNS, the average gap $G\%$ is calculated as $100 \cdot \frac{AvgC - BK}{BK}$. In Table 5, the best-known solutions and the shortest (average) CPU times for the best performing method on each instance are bolded. The results of comparison on the third data set are presented in Table 6, which has the same structure as Table 5.

As it can be seen from Table 5, SVNS was able to obtain the best-known solutions for all test instances, with average gap of 0.80%. VND, MS-VND and GVNS found best-known solution on 15, 14, and 18 (out of 30) test instances, respectively. However, the resulting average gaps remain very small, 3.01% for VND, 3.28% for MS-VND, and 2.27% for GVNS. Regarding the (average) minimum CPU time, the superior method is MS-VND, followed by GVNS, VND, and SVNS. The corresponding (average) minimum CPU times are 3.74, 5.37, 8.55, and 74.35 seconds, respectively. This means that MS-VND is 1.44 times faster than GVNS, 2.29 times faster than VND, and 19.88 times faster than SVNS.

The results presented in Table 6 on larger size test instances, show that SVNS remains superior to other three methods regarding solution quality. For each test instance from the third data set, SVNS method produced best-known solution at least once within 10 runs. On the same data set, SVNS reached the best-known solution in each of 10 runs in the case of 28 out of 30 examples. In the case of instance $l = 50$ and $i = 10$ from the third data set, remaining three algorithms performed better on average with $AvgC = 926$, while for SVNS $AvgC = 926.9$. For instance $l = 60$ and $i = 6$ from the same data set, VND and GVNS showed slightly better performance with $AvgC = 1143$ compared to SVNS with $AvgC = 1189.4$ and MS-VND with $AvgC = 1201$. VND, MS-VND and GVNS have similar performance: GVNS reached best-known solution on 18 out of 30 instances, while VND and MS-VND generated 17 and 16 best-known solutions, respectively. All four methods have small average gaps from the best-known solution: 0.51% for SVNS, 1.03% for GVNS, 1.38% for VND, and 1.55% for MS-VND. MS-VND showed the best performance in respect to CPU time (8.38 *s*) followed by VND (13.84 *s*), GVNS (14.06 *s*) and SVNS (45.49 *s*). Therefore, MS-VND is 1.65 times faster than VND, 1.68 times faster than GVNS and 5.43 times faster than SVNS.

From the presented computational results, it can be seen that the average gap values and required CPU times are quite small for all four VNS based methods, and therefore, all of them can be considered suitable for DMCHBAP. However, SVNS outperforms other methods on both data sets regarding the solution quality, while MS-VND is able to provide high quality solutions in short execution times.

Table 7 shows summarized computational results of the proposed VNS methods on the fourth data set with $m = 13$, $T = 112$ and $l = 70,80,90,100$. These test instances are the hardest ones, because of the large number of vessels and high density of their allocation. Table 7 has the same structure as Tables 5 and 6, the only difference is that each column of Table 7 contains average values obtained on the subset of 10 generated instances ($i \in \{1,..10\}$) from the fourth set with fixed value of l. As in the case of the first three data sets, each VNS method is run 10 times on each instance from the fourth data set. On each subclass with fixed value of l, the best result regarding average best cost, average cost, average CPU time and average gap is bolded. The summarized results presented in Table 7 show that all four VNS-based methods have stable performance. On average, the fastest method is VND (77.75 *s*), however, its average gap is the highest one (3.39%). Other three VNS methods have small computational times with no significant difference among them (between 116.45 *s* and 138.63 *s*). On average, VND is 1.50 times faster than GVNS, 1.59 times faster than MS-VND, and 1.78 times faster than SVNS on the fourth data set. GVNS shows the best performance regarding stability, as its average gap is 1.94%. However, the average gaps of MS-VND and SVNS are also quite small (under 3%). Detailed computational results on these instances can be found at http://www.mi.sanu.ac.rs/tanjad/DMCHBAP.htm.

Computational results presented in Table 7 show that all four presented methods stay stable and efficient even in the case of very hard test instances with large number of allocated vessels. These results verify that VNS based methods can be considered suitable for DMCHBAP and that expected running times on large test instances remain desirable small.

6 Conclusion

In order to meet all requirements of a port as a highly dynamic system, terminal manager needs an ef-

Table 7

Computational results of VNS-based metaheuristics on instances from the fourth data set (m = 13, T = 112, l = 70, 80, 90, 100)

		VND			MS-VND				GVNS				SVNS			
l	*BK*	*Best*	*Time*	*G%*	*Best*	*AvgC*	*AvgT*	*G%*	*Best*	*AvgC*	*AvgT*	*G%*	*Best*	*AvgC*	*AvgT*	*G%*
l=70	991.70	1014.10	119.93	2.29	996.70	1005.07	188.73	1.34	**994.80**	1008.62	**108.30**	1.77	996.20	**1003.57**	134.24	**1.26**
l=80	1016.40	1025.50	**28.45**	0.93	1019.60	1024.01	86.10	**0.75**	1023.90	1025.22	39.68	0.90	**1021.50**	**1023.97**	64.47	0.77
l=90	1393.30	1486.60	**43.46**	6.69	1460.00	1479.52	115.45	6.19	**1406.00**	1453.36	180.56	4.15	1409.40	**1452.01**	186.39	**4.05**
l=100	1436.40	1488.10	119.17	3.65	2820.90	1484.15	164.22	3.26	**1442.30**	**1449.49**	137.27	**0.92**	1443.90	1464.21	**108.64**	2.08
average:	1209.45	1253.58	**77.75**	3.39	1574.30	1248.19	138.63	2.88	**1216.75**	**1234.17**	**116.45**	**1.94**	1217.75	1235.94	123.44	2.04

ficient and reliable decision support system. The performance of decision support system heavily depends on the speed of finding high-quality solutions for underlying berth allocation problem. We studied the Dynamic Minimum Cost Hybrid Berth Allocation Problem (DMCHBAP), which has a great importance in maritime transportation. We prove that even simple variant of this problem is NP-hard, and therefore, it should be addressed by metaheuristic methods. We have developed four VNS-based approaches to DMCHBAP: Variable Neighborhood Descent (VND), Multi-Start Variable Neighborhood Descent (MS-VND), General Variable Neighborhood Search (GVNS), and Skewed Variable Neighborhood Search (SVNS). In order to compare the efficiency of the four VNS-based metaheuristics against each other, four sets of randomly generated test instances were considered. As metaheuristics are generally stochastic methods, we have examined their stability by performing multiple runs.

The obtained experimental results show that all four VNS-based metaheuristics reach known optimal solution on each small size instance (with exception of one instance in the case of VND) in very short CPU times. For larger problem dimensions, all four VNS-based methods were able to find high quality solutions in short running times. On average, SVNS shows the best performance regarding the solution quality, while VND and MS-VND methods are superior in respect to the required CPU time.

Our computational results indicate that all four VNS-based metaheuristics have obvious potential as solution methods for DMCHBAP and related problems in maritime transportation. For further improving of metaheuristics' performance, their parallelization, hybridization and combination with exact methods might be considered.

Acknowledgments

This research was partially supported by Serbian Ministry of Education, Science, and Technological Development under the grants nos. 174010 and 174033.

References

1. Alsoufi, G., Yang, X., Salhi, A. H. Robust Berth Allocation Using a Hybrid Approach Combining Branch-and-Cut and the Genetic Algorithm. In: Blesa M. et al. (Eds.), Hybrid Metaheuristics. Lecture Notes in Computer Science, Springer, Cham, 2016, 9668, 187-201. https://doi.org/10.1007/978-3-319-39636-1_14
2. Bierwirth, C., Meisel, F. A Survey of Berth Allocation and Quay Crane Scheduling Problems in Container Terminals. European Journal of Operational Research, 2010, 202(3), 615-627. https://doi.org/10.1016/j.ejor.2009.05.031
3. Bierwirth, C., Meisel, F. A Follow-Up Survey of Berth Allocation and Quay Crane Scheduling Problems in Container Terminals. European Journal of Operational Research, 2015, 244(3), 675-689. https://doi.org/10.1016/j.ejor.2014.12.030
4. Chang, D., Jiang, Z., Yan, W., He, J. Integrating Berth Allocation and Quay Crane Assignments. Transportation Research Part E, 2010, 46(6), 975-990. https://doi.org/10.1016/j.tre.2010.05.008
5. Cordeau, J. F., Laporte, G., Legato, P., Moccia, L. Models and Tabu Search Heuristics for the Berth-Allocation Problem. Transportation Science, 2005, 39(4), 526-538. https://doi.org/10.1287/trsc.1050.0120
6. Davidović, T., Kovač, N., Stanimirović, Z. VNS-Based Approach to Minimum Cost Hybrid Berth Allocation Problem. Proceedings of XLII International Symposium on Operations Research, (SYMOPIS 2015), Silver Lake, Serbia, September 15-18, 2015, 237-240.
7. Davidović, T., Lazić, J., Mladenović, N., Kordić, S., Kovač, N., Dragović, B. MIP-Heuristics for Minimum Cost Berth Allocation Problem. Proceedings of International Conference on Traffic and Transport Engineering, (ICTTE 2012), Belgrade, Serbia, November 29-30, 2012, 21-28.
8. Expósito-Izquiero, C., Lalla-Ruiz, E., Lamata, T., Melián-Batista, B., Moreno-Vega, J. M. Fuzzy Optimization Models for Seaside Port Logistics: Berthing and Quay Crane Scheduling. In: Madani K., Dourado A., Rosa A., Filipe J., Kacprzyk J. (Eds.), Computational Intelligence. Studies in Computational Intelligence, Springer, 2016, 613, 323-343.
9. Garey, M. R., Johnson, D. S. Computers and Intractability: A Guide to the Theory of NP-Completness. W. H. Freeman and Company, 1979.

10. Gargari, M. M., Niasar, M. S. F. A Dynamic Discrete Berth Allocation Problem for Container Terminals. Proceedings of International Maritime and Port Technology and Development Conference, Trondheim, Norway, October 27-29, 2014, 11-18. https://doi.org/10.1201/b17517-3

11. Gharehgozli, A. H., Roy, D., de Koster, R. Sea Container Terminals: New Technologies and OR Models. Maritime Economics & Logistics, 2016, 18(2), 103-140. https://doi.org/10.1057/mel.2015.3

12. Giallombardo, G., Moccia, L., Salani, M., Vacca, I. Modeling and Solving the Tactical Berth Allocation Problem. Transportation Research Part B: Methodological, 2010, 44(2), 232-245. https://doi.org/10.1016/j.trb.2009.07.003

13. Golias, M. M., Boile, M., Theofanis, S. A Lamda-Optimal Based Heuristic for The Berth Scheduling Problem. Transportation Research Part C: Emerging Technologies, 2010, 18(5), 794-806. https://doi.org/10.1016/j.trc.2009.07.001

14. Golias, M. M., Haralambides, H. E. Berth Scheduling with Variable Cost Functions. Maritime Economics and Logistics, 2011, 13(2), 174-189. https://doi.org/10.1057/mel.2011.4

15. Golias, M. M., Saharidis, G. K., Boile, M., Theofanis, S., Ierapetritou, M. G. The Berth Allocation Problem: Optimizing Vessel Arrival Time. Maritime Economics and Logistics, 2009, 11(4), 358-377. https://doi.org/10.1057/mel.2009.12

16. Government of India. Reducing Dwell Time of Cargo at Ports, Report of the Inter-ministerial Group. Planning Commission, Government of India, 2007. Accessed on August 17, 2017.

17. Han, M., Li, P., Sun, J. The Algorithm for Berth Scheduling Problem by the Hybrid Optimization Strategy GASA. Proceedings of 9th International Conference on Control, Automation, Robotics and Vision, (ICARCV'06), Singapore, December 5-8, 2006, 1-4. https://doi.org/10.1109/ICARCV.2006.345455

18. Hansen, P., Mladenović, N. Developments of The Variable Neighborhood Search. In: Ribeiro, C., Hansen, P. (Eds.), Essays and Surveys in Metaheuristics, Kluwer Academic Publishers, 2002, 415-439. https://doi.org/10.1007/978-1-4615-1507-4_19

19. Hansen, P., Oğuz, C., Mladenović, N. Variable Neighborhood Search for Minimum Cost Berth Allocation. European Journal of Operational Research, 2008, 191(3), 636-649. https://doi.org/10.1016/j.ejor.2006.12.057

20. Hansen, P., Mladenović, N., Brimberg, J., Moreno Pérez, J. A. Variable Neighborhood Search. In: Gendreau, M., Potvin, J.-Y. (Eds.), Handbook of Metaheuristics, Springer, New York Dordrecht Heidelberg London, 2010, 61-86. https://doi.org/10.1007/978-1-4419-1665-5_3

21. Imai, A., Nishimura, E., Hattori, M., Papadimitriou, S. Berth Allocation at Indented Berths for Mega-Containerships. European Journal of Operational Research, 2007, 179(2), 579-593. https://doi.org/10.1016/j.ejor.2006.03.034

22. Karafa, J., Golias, M. M., Ivey, S., Saharidis, G. K. D., Leonardos, N. The Berth Allocation Problem with Stochastic Vessel Handling Times. The International Journal of Advanced Manufacturing Technology, 2013, 65(1-4), 473-484. https://doi.org/10.1007/s00170-012-4186-0

23. Kim, K. H., Moon, K. C. Berth Scheduling by Simulated Annealing. Transportation Research Part B, 2003, 37(6), 541-560. https://doi.org/10.1016/S0191-2615(02)00027-9

24. Kordić, S., Davidović, T., Kovač, N., Dragović, B. Combinatorial Approach to Exactly Solving Discrete and Hybrid Berth Allocation Problem. Applied Mathematical Modeling, 2016, 40(21-22), 8952-8973. https://doi.org/10.1016/j.apm.2016.05.004

25. Kovač, N. Metaheuristic Approaches for the Berth Allocation Problem. Yugoslav Journal of Operations Research, 2017, 27(3), 265-289. https://doi.org/10.2298/YJOR160518001K

26. Kovač, N., Stanimirović, Z., Davidović, T. Metaheuristic Approaches for the Minimum Cost Hybrid Berth Allocation Problem. In: Konstantopoulos, C., Pantziou, G. (Eds.), Modelling, Computing and Data Handling Methodologies for Maritime Transport, Springer, New York Dordrecht, Heidelberg London, 2017, 1-47.

27. Lalla-Ruiz, E. A., Batista, B. M., Vega, J. M. M. Adaptive Variable Neighborhood Search for Berth Planning in Maritime Container Terminals. Proceedings of GREEN-COPLAS 2013: IJCAI 2013 Workshop on Constraint Reasoning, Planning and Scheduling Problems for a Sustainable Future, Beijing, China, August 3-9, 2013, 35-42.

28. Lalla-Ruiz, E. A., Voß, S. Towards a Matheuristic Approach for the Berth Allocation Problem. In: Pardalos P., Resende M., Vogiatzis C., Walteros J. (Eds.), Learning and Intelligent Optimization. LION 2014. Lecture Notes in Computer Science, Springer, Cham, February 16-21, 2014, 8426, 218-222. https://doi.org/10.1007/978-3-319-09584-4_20

29. Lalla-Ruiz, E. A., Voß, S. POPMUSIC as a Matheuristic for the Berth Allocation Problem. Annals of Mathematics and Artificial Intelligence, 2016, 76(1-2), 173-189. https://doi.org/10.1007/s10472-014-9444-4

30. Lalla-Ruiz, E., Voß, S., Expósito-Izquierdo, C., Melián-Batista, B., Moreno-Vega, J. M. A POPMUSIC-Based Approach for the Berth Allocation Problem Under Time-Dependent Limitations. Annals of Operations Research, 2017, 253(2), 871-897. https://doi.org/10.1007/s10479-015-2055-6

31. Lee, D. H., Chen, J. H., Cao, J. X. The Continuous Berth Allocation Problem: A Greedy Randomized Adaptive Search Solution. Transportation Research Part E: Logistics and Transportation Review, 2010, 46(6), 1017-1029. https://doi.org/10.1016/j.tre.2010.01.009

32. Lim, A. The Berth Planning Problem. Operations Research Letters, 1998, 22(2), 105-110. https://doi.org/10.1016/S0167-6377(98)00010-8

33. Meisel, F. Seaside Operations Planning in Container Terminals. Springer, Berlin Heidelberg, 2009. https://doi.org/10.1007/978-3-7908-2191-8

34. Mladenović, N., Hansen, P. Variable Neighborhood Search. Compututers and Operations Research, 1997, 24(11), 1097-1100. https://doi.org/10.1016/S0305-0548(97)00031-2

35. Murata, H., Fujiyoshi, K., Nakatake, S., Kajitani, Y. VLSI Module Placement Based on Rectangle-Packing by the Sequence-Pair. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 1996, 15(12), 1518-1524. https://doi.org/10.1109/43.552084

36. Nishimura, E., Imai, A., Papadimitriou, S. Berth Allocation Planning in the Public Berth System by Genetic Algorithms. European Journal of Operational Research, 2001, 131(2), 282-292. https://doi.org/10.1016/S0377-2217(00)00128-4

37. de Oliveira, R. M., Mauri, G. R., Lorena, L. A. N. Clustering Search for the Berth Allocation Problem. Expert Systems with Applications, 2012, 39(5), 5499-5505. https://doi.org/10.1016/j.eswa.2011.11.072

38. Pinedo, M. Scheduling Theory, Algorithms and Systems. Prentice Hall, 2008.

39. Port of Rotterdam. 2017 Annual Report: Results Create New Scope for Ambitious Investment Programme. http://www.portofrotterdam.com/en/news-and-press-releases/2017-annual-report-results-create-new-scope-for-ambitious-investment. Accessed on March 23, 2018.

40. Rashidi, H., Tsang, E. P. K. Novel Constraints Satisfaction Models for Optimization Problems in Container Terminals. Applied Mathematical Modelling, 2013, 37(6), 3601-3634. https://doi.org/10.1016/j.apm.2012.07.042

41. Rodriguez-Molins, M., Ingolotti, L., Barber, F., Salido, M. A., Sierra, M. R., Puente, J. A Genetic Algorithm for Robust Berth Allocation and Quay Crane Assignment. Progress in Artificial Intelligence, 2014, 2(4), 177-192. https://doi.org/10.1007/s13748-014-0056-3

42. Saharidis, G. K. D., Golias, M. M., Boile, M., Theofanis, S., Ierapetritou, M. G. The Berth Scheduling Problem with Customer Differentiation: A New Methodological Approach Based on Hierarchical Optimization. The International Journal of Advanced Manufacturing Technology, 2010, 46(1-4), 377-393. https://doi.org/10.1007/s00170-009-2068-x

43. Simrin, A., Diabat, A. The Dynamic Berth Allocation Problem: A Linearized Formulation. RAIRO-Operations Research, 2015, 49(3), 473-494. https://doi.org/10.1051/ro/2014039

44. Stahlbock, R., Voß, S. Operations Research at Container Terminals: A Literature Update. OR Spectrum, 2008, 30(1), 1-52. https://doi.org/10.1007/s00291-007-0100-9

45. Theofanis, S., Boile, M., Golias, M. An Optimization Based Genetic Algorithm Heuristic for the Berth Allocation Problem. IEEE Congress on Evolutionary Computation, CEC 2007, September 25-28, 2007, 4439-4445. https://doi.org/10.1109/CEC.2007.4425052

46. Ting, C. J., Wu, K. C., Chou, H. Particle Swarm Optimization Algorithm for the Berth Allocation Problem. Expert Systems with Applications, 2014, 41(4), 1543-1550. https://doi.org/10.1016/j.eswa.2013.08.051

47. Umang, N., Bierlaire, M., Vacca, I. Exact and Heuristic Methods to Solve the Berth Allocation Problem in Bulk Ports. Transportation Research Part E: Logistics and Transportation Review, 2013, 54, 14-31. https://doi.org/10.1016/j.tre.2013.03.003

48. UNCTAD. Review of Maritime Transport. http://unctad.org/en/PublicationsLibrary/rmt2017_en.pdf, 2017. Accessed on March 23, 2018.

49. Zhou, P., Kang, H. Study on Berth and Quay-Crane Allocation Under Stochastic Environments in Container Terminal. Systems Engineering-Theory and Practice, 2008, 28(1), 161-169. https://doi.org/10.1016/S1874-8651(09)60001-6

50. Zhou, P., Kang, H., Lin, L. A Dynamic Berth Allocation Model Based on Stochastic Consideration. Proceedings of the Sixth World Congress on Intelligent Control and Automation, 2006, (WCICA 2006.), Dalian, China, June 21-23, 2006, 2, 7297-7301.