Applied Mathematical Modelling 000 (2016) 1–22

Contents lists available at ScienceDirect

Applied Mathematical Modelling

journal homepage: www.elsevier.com/locate/apm

Combinatorial approach to exactly solving discrete and hybrid berth allocation problem

Stevan Kordić[a,*], Tatjana Davidović[b], Nataša Kovač[a], Branislav Dragović[a]

[a] Maritime Faculty, University of Montenegro, Kotor, Montenegro
[b] Mathematical Institute, Serbian Academy of Sciences and Arts, Belgrade, Serbia

ARTICLE INFO

Article history:
Received 14 March 2015
Revised 28 February 2016
Accepted 4 May 2016
Available online xxx

Keywords:
Combinatorial optimization
Branch and bound
Optimal solution
Berth allocation problem
Minimization of total cost

ABSTRACT

This paper presents an exact combinatorial algorithm for solving the *Discrete Berth Allocation Problem (DBAP)* and the *Hybrid Berth Allocation Problem (HBAP)* with fixed handling times of vessels based on the original algorithm for solving combinatorial problems called *Sedimentation Algorithm*. We address the issues of DBAP and HBAP according to the Rashidi and Tsang model. To the best of our knowledge, the proposed algorithm is the first exact combinatorial algorithm for solving the general DBAP and HBAP based on Rashidi and Tsang model. Computational results prove the superiority of the proposed algorithms compared with the exact solvers based on the Mixed Integer Programming (MIP) models. Efficient C implementation enabled us to solve instances with up to 65 vessels. This resolves most of the real life problems, even in large ports.

1. Introduction

The *Berth Allocation Problem* (BAP) consists of allocating berths to a set of vessels that need to be served within a given time horizon in a container port. Vessels are, among other information, represented by a set of data that includes the expected time of arrival, size, projected handling time, preferred berth in the port, and penalties. BAP can be defined as follows: for each vessel in the set, the berth index and the time interval are allocated in the manner that the given objective function is minimized. In Lim [1], BAP was proven to be a NP-hard problem.

BAPs can be classified as discrete, continuous or hybrid, see [2]. In DBAP quay is partitioned into a number of units, called berths, and each berth can serve one vessel at a time. Each vessel, on the other hand, occupies exactly one berth. Time is also partitioned into discrete units, which allows the usage of integer arithmetic for the calculation of the objective function value. HBAP is similar to DBAP with the exception that a larger vessel can occupy several successive berths while smaller vessels can share one berth. Conversely, in the continuous BAP there is no partitioning of the quay, i.e. vessels are allowed to take arbitrary position within the boundaries of the quay. Another possible classification distinguishes static and dynamic BAPs. In the static BAP, it is assumed that vessels arrival times impose soft constraint on the berthing times. The vessels already wait at the port and can berth immediately or the vessel can be speeded up in order to meet berthing time earlier than the expected arrival time. In the dynamic BAP fixed arrival times are given for the vessels, hence, vessels cannot berth before expected arrival time. A detailed BAP classification can be found in [2,3].

* Corresponding author.
E-mail addresses: stevan.kordic@gmail.com (S. Kordić), tanjad@mi.sanu.ac.rs (T. Davidović), knatasa@ac.me (N. Kovač), branod@ac.me (B. Dragović).

http://dx.doi.org/10.1016/j.apm.2016.05.004
0307-904X/

In recent literature, exact approaches addressing BAP are rare, the majority of studies use heuristic or meta-heuristic methods to obtain suboptimal solutions of BAP. According to the recent surveys of BAP by Bierwirth and Meisel [2,4] exact methods are applied in 24% of approaches, while the rest of 76% approaches belongs to the heuristic and meta-heuristic methods.

An exact method for solving BAP can be found in Vacca et al. [5]. The authors proposed an exact algorithm for solving the *Tactical Berth Allocation Problem* (TBAP) defined by Giallombardo et al. [6]. The model for TBAP is based on an exponential number of variables, and it is solved via column generation. To obtain an integer solution, a branch-and-price scheme was applied along with several accelerating techniques specifically designed for solving TBAP.

Although BAP in container port is different from BAP in bulk port we list two works addressing BAP in bulk port. In Umang et al. [7] two exact methods, based on mixed integer programing and generalized set partitioning, and one heuristic methods to solve dynamic hybrid BAP in bulk ports are proposed. Robenek et al. [8] propose branch-and-price exact solution algorithm.

Heuristic and meta-heuristic methods for solving BAP are far more common. The following short review does not cover this topic completely. Its purpose is just to illustrate the variety of approaches.

Both static and dynamic discrete BAPs were examined by Imai et al. [9]. In both problem variants, the assignment and sequencing of vessels to berths was determined by minimizing the vessels' waiting and handling times. A Lagrangian relaxation-based heuristic was used to solve the problem. A similar approach, with a stronger Lagrangian relaxation because of the different formulation used, was applied by Monaco and Samara [10] for the dynamic version of DBAP. Cordeau et al. [11] modeled DBAP as a Multi-Depot Vehicle Routing Problem with Time Windows and applied the Tabu Search meta-heuristic to find good sub-optimal solutions for the problem. A similar approach was adopted by Mauri et al. [12] to solve DBAP. The set partition approach was used by Cristensen and Holst [13] to solve DBAP. Zhen et al. [14] applied a Simulated Annealing meta-heuristic, whereas de Oliveira et al. [15] applied Clustering Search method using Simulated Annealing for solutions generation. Lee and Chen [16] and Hansen et al. [17] used a Variable Neighborhood Search for the same variant of the problem. Genetic Algorithms were applied to several variants of DBAP by Imai et al. [18], Han et al. [19], Zhou et al. [20], and Nishimura et al. [21]. Iterated Greedy Heuristic for solving DBAP was used by Lin et al. [22]. Lalla-Ruiz and Voß [23] employ Partial Optimization Metaheuristic Under Special Intensification Condition Metaheuristic (POPMUSIC) for solving DBAP.

HBAP with fixed handling times was examined by Chen and Hsieh [24] using the MIP problem formulation. In order to solve HBAP Moorthly and Teo [25] used a precedence graph representation which is analyzed using the Project Evaluation Review Technique. Dai et al. [26] proposed Simulated Annealing algorithm for solving the same version of HBAP. Bee Colony Optimization was applied by Kovač [27] for solving the Minimum Cost Hybrid BAP with fixed handling times of vessels.

The HBAP formulations with position dependent vessel handling times are studied in several papers. Imai et al. [28] investigate indented berths HBAP. Also, Imai et al. [29] developed Genetic Algorithm for the berth allocation of the mega-ships served from two sides. Cordeau et al. [11] obtain HBAP from DBAP. The works of Nishimura et al. [21], Cheong et al. [30] and Hoffarth and Voß [31] include the vessels' draft into HBAP. The same work proposed a heuristic for solving HBAP.

In this paper the discrete BAP (DBAP) and the hybrid BAP (HBAP) are considered. We present an original exact approach for solving DBAP and HBAP implemented in two variants. The first variant, named *Sedimentation Algorithm* (SEDA), is a general combinatorial optimization algorithm adopted for solving BAP. SEDA is an exact solver and it works on the combinatorial branch and bound principles. The second variant differs from the first one because it uses a heuristic in the pre-processing phase to reduce the search space for SEDA. We name this algorithm the *Sedimentation Algorithm with an Estimation & Rearrangement Heuristic* (SEDA+ERH). To the best of our knowledge, these are the first exact combinatorial algorithms for solving the general DBAP and HBAP based on Rashidi and Tsang [32] model. Estimations of the complexity of the algorithms are given. Numerical experiments are conducted on four sets of test examples involving 5, 8 or 13 berths with one or two-week time horizon. SEDA+ERH enable us to find, in a very short CPU times, the optimal solution for the larger problem instances with up to 60 vessels to be scheduled during the time horizon of one or two weeks. In addition, we compare our combinatorial approach (realized through the implementation of SEDA and SEDA+ERH) against commercial MIP based exact solver CPLEX. Our computational results clearly prove the superiority of the proposed combinatorial algorithms.

The rest of this paper is organized as follows. At the beginning, SEDA and SEDA+ERH are described in Section 2. Then BAP notation and problem formulation are introduced in Section 3. BAP modeling for SEDA and the complexity of SEDA for solving BAP are described in Section 4. Computational results are presented in Section 5. Finally, Section 6 contains concluding remarks and directions for future research.

2. The sedimentation algorithm

Sedimentation Algorithm (SEDA) is a general combinatorial optimization algorithm introduced for the first time by the authors in Kordić et al. [33] for solving the Berth Allocation Problem (BAP) at *International Association of Maritime Economists Conference, IAME, Taipei, 2012*. Since the proceedings of this conference is not accessible to wider scientific community here we present in detail SEDA and SEDA+ERH. The initial version of SEDA and its preliminary computational results presented at [33] are significantly improved and presented here.

Please cite this article as: S. Kordić et al., Combinatorial approach to exactly solving discrete and hybrid berth allocation problem, Applied Mathematical Modelling (2016), http://dx.doi.org/10.1016/j.apm.2016.05.004

SEDA belongs to the class of branch-and-bound algorithms, which uses the backtracking mechanism combined with some look-ahead techniques for the exact solving of optimization problems. Here we present a recursive variant of the algorithm suitable for the minimization type of problems.

In order to solve a problem by SEDA, first we need to present the problem's model in the form required by SEDA. The problem modeling consists of: input parameters, internal structures, functions and procedures described in Sections 2.1, 2.2 and 2.3. The description of implementation is given in Section 2.4. SEDA and its key properties are presented in Section 2.5. and 2.6. SEDA+ERH is described in Section 2.7. Finally, in Subsection 2.8 advantages and disadvantages of SEDA are listed.

2.1. SEDA input parameters

The SEDA input parameters are the following:

1. $X = \{x_1, \ldots, x_l\}$ – a finite set of decision variables.
2. $Dom = \{D_1, \ldots, D_l\}$ – the set of domains (possible values) of decision variables. SEDA will work only if the domains of the decision variables D_i, for each $i \in \{1, \ldots, l\}$ are finite nonempty sets.
3. f – the objective function, depending on the values of decision variables, i.e. $f(x_1, \ldots, x_l)$. We assume that the objective function can be represented as:

$$f(x_1, \ldots, x_l) = \sum_{k=1}^{l} f_k(x_k), \tag{1}$$

where functions $f_k(x_k)$ are nonnegative for each $k \in \{1, \ldots, l\}$.

4. Ξ – the set of heuristic relations. Set $\Xi = \{\prec_1, \ldots, \prec_l\}$ consists of heuristic total order relations $\prec_i \subseteq D_i^2$, for each decision variable $x_i \in X$. For any two members $a, b \in D_i$, relation $\prec_i$ decides which value of decision variable x_i is better: a or b. The value a comes before b if $a \prec_i b$. Heuristic relations have to satisfy following condition:

$$(\forall i \in \{1, \ldots l\})(\forall a, b \in D_i) \quad a \prec_i b \Rightarrow \ f_i(a) \leq f_i(b). \tag{2}$$

2.2. Internal data structures used by SEDA

Internal structures used by SEDA are the following:

1. o – the sequence in which the values of decision variables corresponding to the current best value of the objective function are kept during the work of the algorithm. This is the sequence where current best solution is kept.
2. *minimum* – the variable for keeping the value of the objective function of the current best solution. The value of the variable *minimum* is completely determined by the values of the current best solution i.e., $minimum = f(o_1, \ldots, o_l)$ if o is determined, otherwise $minimum = +\infty$.
3. θ – the counter of the decision variables. We will also refer to variable θ as a construction step counter.

2.3. Functions and procedures used by SEDA

Procedures used by SEDA are the following:

1. FindSolution(θ, *Dom*) – the recursive procedure which finds the value for the decision variable x_θ, in the domain set $D_\theta \in Dom$, and then proceeds until a feasible solution is reached. The procedure will be described in more details in Subsection 2.5.
2. Estimation(θ, *Dom*) – the function which estimates the value of the objective function over the decision variables domain sets in *Dom*. If the values for the first $\theta - 1$ decision variables are determined i.e., values are determined for $\{x_1, \ldots, x_{\theta-1}\}$, then the function Estimation(θ, *Dom*) can be represented as:

$$\text{Estimation}(\theta, Dom) = \sum_{k=1}^{\theta-1} f_k(x_k) + \text{NonDetVarEstimation}(\theta). \tag{3}$$

The above sum is calculated according to the objective function definition for the decision variables with determined values: $\{x_1, \ldots, x_{\theta-1}\}$. For the decision variables $\{x_\theta, \ldots, x_l\}$ with still undetermined values we use the function NonDetVarEstimation(θ) to estimate minimal possible value to approximate objective function f. The quality of the approximation highly effects the SEDA running time. General assumption about the Estimation(θ, *Dom*) function is that its value is lower than or equal to the value of the objective function of the "optimal solution" with determined set of decision variables $\{x_1, \ldots, x_{\theta-1}\}$ if the following holds:

$$(\forall k \in \{\theta, \ldots, l\}) \ D_k \neq \emptyset. \tag{4}$$

Otherwise the value is $+\infty$. It will take value $+\infty$ if and only if $D_k = \emptyset$, for some $k \in \{\theta, \ldots, l\}$ i.e., feasible solution is not possible to construct for the given set of variable domains *Dom*.

Please cite this article as: S. Kordić et al., Combinatorial approach to exactly solving discrete and hybrid berth allocation problem, Applied Mathematical Modelling (2016), http://dx.doi.org/10.1016/j.apm.2016.05.004

```
1  Sediment(θ, x)
2      for i = θ + 1 to l do
3          "Remove elements in D_i which violates
               constrains of the problem if x_θ = x"
           "Remove elements in D_i determined by built in
               specific look-a-head techniques of the problem if x_θ = x"
4      endfor
5      return D
6  end
```

Fig. 1. The Sedimentation Function.

In this paper, in particular, we will assume that NonDetVarEstimation(θ) is calculated in the following way:

$$\text{NonDetVarEstimation}(\theta) = \sum_{k=\theta}^{l} \begin{cases} \min\limits_{y \in D_k} f_k(y) & \text{if} \quad D_k \neq \emptyset, \\ +\infty & \text{if} \quad D_k = \emptyset. \end{cases} \tag{5}$$

Therefore, the function Estimation(θ, *Dom*) can be represented as:

$$\text{Estimation}(\theta, Dom) = \sum_{k=1}^{\theta-1} f_k(x_k) + \sum_{k=\theta}^{l} \begin{cases} \min\limits_{y \in D_k} f_k(y) & \text{if} \quad D_k \neq \emptyset, \\ +\infty & \text{if} \quad D_k = \emptyset. \end{cases} \tag{6}$$

For the Estimation(θ, *Dom*) defined as above general assumption trivially holds.

3. Sediment(θ, x) – the function which propagates the assignment of x to the decision variable x_θ for all further exploration of the domains of the undetermined decision variables: $\{D_{\theta+1}, \ldots, D_l\}$. Also, it applies various general and problem specific *look-ahead* techniques for the further reduction of $\{D_{\theta+1}, \ldots, D_l\}$. Upon the *"sedimentation"*, the function returns the new value to the set of domains. The pseudocode of the Sediment(θ, x) is given in Fig. 1

   The name of the algorithm was inspired by the procedure resembling the natural phenomenon related to the sedimentation of particles in fluids.

4. ReportSolution() – the procedure which checks if the new feasible solution corresponds to a smaller value of the objective function f than the current best solution. If that is the case, then it becomes the new current best solution and it is saved in the variable o. Also, its objective function value is saved in the variable *minimum*.
5. CutOff(θ, ε) – the function which deletes elements from the domains of the undetermined decision variables $\{D_{\theta+1}, \ldots, D_l\}$ that correspond to high values of f_k function. Let us denote by M_k the sum of the minimal values of the functions f_k over the domain D_k for $k \in \{\theta+1, \ldots, l\}$ and a positive number ε i.e.,

$$M_k = \min_{y \in D_k} f_k(y) + \varepsilon, \quad k \in \{\theta+1, \ldots, l\}. \tag{7}$$

   The function CutOff(θ, ε) returns the new value of the set of domains: $\{D_1, \ldots, D_\theta, D'_{\theta+1}, \ldots, D'_l\}$. The domains of the determined decision variables remain unchanged, while the domains of the undetermined decision variables are calculated in the following way:

$$D'_k = \{y \in D_k \mid f_k(y) < M_k\}, \quad k \in \{\theta+1, \ldots, l\}. \tag{8}$$

6. Minx(D, $\prec$) – the function returns minimal $\prec$ element of the set D, i.e., it returns $a \in D$ such that:

$$(\forall y \in D) \;\; y \neq a \;\Rightarrow\; a \prec y. \tag{9}$$

2.4. Implementation of the functions and procedures used by SEDA

The main reason for efficiency of the SEDA algorithm lies in the way how functions Estimation(θ, *Dom*), Sediment(θ, a), CutOff(θ, ε) and Minx(D, $\prec$) are implemented. Internally we maintain so called ξ lists for each decision variable. Elements of the ξ_k list for the decision variable x_k are ordered pairs $(a, f_k(a))$ for all $a \in D_k$, sorted in ascending order according to the total ordering relation $\prec_k$. If we denote by $n_k = |D_k|$, then ξ_k list can be represented as:

$$\xi_k = \langle (a_{k,1}, f_k(a_{k,1})), (a_{k,2}, f_k(a_{k,2})), \ldots, (a_{k,n_k}, f_k(a_{k,n_k})) \rangle, \;\; k \in \{1, \ldots, l\}, \tag{10}$$

providing the following three formulas are true:

$$(\forall k \in \{1, \ldots, l\}) \;\; D_k = \{a_{k,1}, a_{k,2}, \ldots, a_{k,n_k}\}; \tag{11}$$

$$(\forall k \in \{1,\ldots,l\})(\forall i,j \in \{1,\ldots,n_k\}) \quad i \neq j \;\Leftrightarrow\; a_{k,i} \neq a_{k,j}; \tag{12}$$

$$(\forall k \in \{1,\ldots,l\})(\forall i,j \in \{1,\ldots,n_k\}) \quad i < j \;\Leftrightarrow\; a_{k,i} \prec_k a_{k,j}. \tag{13}$$

If we also introduce notation:

$$\begin{aligned} &\xi_k(i) = \big(a_{k,i}, f_k\big(a_{k,i}\big)\big), \\ &\xi_k(i,1) = a_{k,i}, \quad \xi_k(i,2) = f_k\big(a_{k,i}\big), \qquad k \in \{1,\ldots,l\}, \quad i \in \{1,\ldots,n_k\}, \end{aligned} \tag{14}$$

then we can describe implementation of SEDA function and procedures using ξ_k list. Implementation of the functions Estimation(θ, *Dom*) and Minx(D, $\prec$) are as follows:

$$\text{Estimation}(\theta, Dom) = \sum_{k=1}^{\theta-1} f_k(x_k) + \sum_{k=\theta}^{l} \begin{cases} \xi_k(1,2) & \text{if} \quad D_k \neq \emptyset, \\ +\infty & \text{if} \quad D_k = \emptyset, \end{cases} \tag{15}$$

$$\text{Minx}(D_k, \prec_k) = \xi_k(1,1). \tag{16}$$

For the implementation of the function CutOff(θ, ε) it is necessary to make search for ordered pair $(a, f_k(a))$ in the list ξ_k, having the highest value of the $f_k(a)$ less than M_k, for each decision variable x_k, $k \in \{\theta+1,\ldots,l\}$. Since ξ_k lists are ordered, binary search appears to be efficient algorithm for the necessary choice. If we introduce μ operator as:

$$\mu y_{y<z} R(z) - \text{The least} \;\; y < z \;\text{such that}\; R(y).\, \text{if}(\exists y \in \mathbb{N})\; y < z \Rightarrow R(y);\; \text{otherwise } z,$$

then the function CutOff(θ, ε) can be implemented as follows:

$$M_k = \xi_k(1,2) + \varepsilon, \quad k \in \{\theta+1,\ldots,l\}, \tag{17}$$

$$n'_k = \mu i_{i<n_k+1}[M_k \leq \xi_k(i,2)], \quad k \in \{\theta+1,\ldots,l\}, \tag{18}$$

$$D'_k = \left\{\xi_k(i,1) \mid i \in \left\{1,\ldots,n'_k - 1\right\}\right\}, \quad k \in \{\theta+1,\ldots,l\}. \tag{19}$$

The function CutOff(θ, ε) needs also to update ξ_k lists for the further calculations in the following way:

$$\xi'_k = \; \xi_k(i) \mid i \in \left\{1,\ldots,n'_k\right\}, \quad k \in \{\theta+1,\ldots,l\}. \tag{20}$$

Finally, the new values for n'_k and ξ'_k replace old. This step concludes the implementation of the CutOff(θ, ε) function:

$$n_k = n'_k \;\text{ and }\; \xi_k = \xi'_k, \quad k \in \{\theta+1,\ldots,l\}, \tag{21}$$

$$D'_k = \{\; \xi_k(i,1) \mid i \in \{1,\ldots,n_k\} \;\wedge\; \varphi(\theta, x, k, \xi_k(i,1)) \;\}, \quad k \in \{\theta+1,\ldots,l\}. \tag{22}$$

Function Sediment(θ, x) is implemented as the sequential search through the ξ_k lists. Elements satisfying constrains and heuristic conditions are kept in the list, the other ones are deleted. Predicate $\varphi(\theta, x, k, y)$ is defined to be true if all determined values of decision variables x_1, x_2,..., $x_{\theta-1}$ and $x_\theta = x$, $x_k = y$ are consistent with constraints and heuristic techniques of the problem (otherwise it is false). Then we can implement Sediment(θ, x) as follows:

At the end, new values of n_k and of ξ_k, should be updated according to D'_k for further calculations.

In this way all the most essential functions and procedures for SEDA can be reduced to the operations with ξ_k lists. The complexity of these operations does not exceed the general complexity of the algorithm, therefore we omit the proof of complexity for them.

Described functions and the ξ lists are integral part of SEDA. The ξ_k lists, for $k \in \{1,\ldots,l\}$ are pre calculated before SEDA starts to work and maintained as described during the work of SEDA, i.e., elements of the ξ lists are deleted in such a manner that their ascending order according to the total ordering relation $\prec_k$ is preserved. This feature inspired us to name this algorithm as the *Sedimentation Algorithm*.

In this subsection, we described the functions Estimation(θ, *Dom*), Sediment(θ, a), CutOff(θ, ε) and Minx(D, $\prec$) in a mathematical way which is more suitable for understanding how SEDA works. The actual programming implementation is, although equivalent, more sophisticated and optimized in order to increase the efficiency of the resulting algorithm.

2.5. SEDA description

The pseudo code of SEDA is given in Fig. 2. It consists of the two parts: the main body of the algorithm, lines **(19)–(23)** and the procedure FindSolution(θ, *Dom*), lines **(2)–(18)**.

In the main body of the algorithm the variables described in Sections 2.1 and 2.2 are initialized, lines **(19)–(21)**. Then procedure FindSolution(1, *Dom*) is called in line **(22)**. This procedure recursively determines the values of decision variables, starting from x_1, then x_2, and so on, until x_l is reached. After examining all possible values for decision variables, variable *minimum* contains the value of the objective function for optimal solution. The optimal solution itself is stored in the sequence *o*. At the end, the values *o* and *minimum* are returned from the main body of the algorithm, line **(23)**. If there is no feasible solution of the problem, then the return value of the variable *minimum* is the initial one i.e., $+\infty$.

As previously mentioned, the procedure FindSolution(θ, *Dom*) recursively examines the solution space. In order to describe how it works, let us suppose that the values of decision variables $\{x_1,\ldots,x_{\theta-1}\}$ are determined. The input variable

Please cite this article as: S. Kordić et al., Combinatorial approach to exactly solving discrete and hybrid berth allocation problem, Applied Mathematical Modelling (2016), http://dx.doi.org/10.1016/j.apm.2016.05.004

```
1  SedimentationAlgorithm(l, Dom, Ξ, f)
2      procedure FindSolution(θ, Dom)
3          Dom' = Dom
4          D = D_θ
5          while D ≠ ∅ and Estimation(θ, Dom) < minimum do
6              x_θ = Minx(D, ≺_θ)
7              D = D \ {x_θ}
8              Dom = Sediment(θ, x_θ)
9              ε = minimum − Estimation(θ + 1, Dom)
10             if ε > 0 then
11                 if θ = l then ReportSolution()
12                     else Dom = CutOff(θ, ε)
13                          FindSolution(θ + 1, Dom)
14                 endif
15             endif
16             Dom = Dom'
17         endwhile
18     end
19     o = ⟨ ∅ | i = 1, ..., l ⟩
20     x = ⟨ ∅ | i = 1, ..., l ⟩
21     minimum = +∞
22     FindSolution(1, Dom)
23     return ⟨o, minimum⟩
24 end
```

Fig. 2. The Sedimentation Algorithm.

Dom is the set of decision variables domains $\{D_1, \ldots, D_l\}$. We assume that these domains are consistent with the values of the decision variables $\{x_1, \ldots, x_{\theta-1}\}$. The next decision variable for which values is to be examined is x_θ.

In line **(3)** the original values of the whole set of domains *Dom* are saved in the auxiliary variable *Dom'*. Also, in particular, the value of the domain D_θ is saved in the auxiliary variable *D*, line **(4)**. If necessary, the algorithm will examine all the elements of the set *D* as the potential values of the decision variable x_θ. After examining a particular value $a \in D$, that value is removed from *D*. Therefore, the algorithm examines values for the decision variable x_θ while $D \neq \emptyset$ and there is a chance to improve the current best solution i.e., Estimation(θ, *Dom*) $<$ *minimum*, in the while loop lines **(5)–(17)**.

In line **(6)** we select the minimal $\prec_\theta$ element of set *D* as the value of the decision variable x_θ. After the selection, the value is removed from the set *D* in line **(7)**. The selection of the value for the decision variable x_θ will have consequences on the domains of decision variables yet undetermined: $\{D_{\theta+1}, \ldots, D_l\}$. These consequences are handled by the call of the Sediment(θ, x_θ) function in line **(8)** which returns new value for decision variables domains. Notice that only the domains of yet undetermined decision variables may be changed.

Since, the value of the decision variable x_θ is determined and all the consequences are reflected in the domains of yet undetermined decision variables it is befitting to estimate the new minimal possible value of the objective function f again. Instead of the direct comparison Estimation$(\theta + 1, Dom) <$ *minimum* in line **(10)**, the difference *minimum* − Estimation$(\theta + 1, Dom)$ is saved in auxiliary variable ε in line **(9)**, then equivalent comparison $\varepsilon > 0$ is performed in line **(10)**. If $\varepsilon > 0$ then the algorithm can proceed with construction of a feasible solution in lines **(11)–(14)**, otherwise a new value for decision variable x_θ has to be examined, if there is any left.

In line **(11)** condition $\theta = l$ is examined. In the case it is true, then the algorithm checks if the new current best solution is produced by calling the procedure ReportSolution(). In the case $\theta < l$ the algorithm proceeds with the construction of feasible solution. Firstly, it cuts off the domain elements with too high values of the objective function by calling the function CutOff(θ, ε) in line **(12)**. After the cutting off, the algorithm continues with the examination of the next decision variable by calling the procedure FindSolution$(\theta + 1, Dom)$ in line **(13).**

Finally, the original value of the decision variables domains is restored in line **(16)** in order to examine the remaining values for the decision variable x_θ.

The backtracking mechanism of SEDA is "hidden" by the recursive formulation of the procedure FindSolution(θ, *Dom*). The non-recursive formulation of SEDA is more complex, but also more efficient. This is why the recursive formulation of SEDA was used to describe the algorithm in this section and non-recursive implementation of SEDA for computational

Please cite this article as: S. Kordić et al., Combinatorial approach to exactly solving discrete and hybrid berth allocation problem, Applied Mathematical Modelling (2016), http://dx.doi.org/10.1016/j.apm.2016.05.004

results in Section 5. The transformation of any recursive function or procedure to a non-recursive one is, more or less, a technical matter. Therefore, the description of non-recursive SEDA is omitted here.

2.6. Key SEDA properties

In this section main properties of the proposed algorithm are described. First we prove total correctness of SEDA and then we address the issue of general complexity of SEDA.

2.6.1. Total correctness of SEDA

SEDA eliminates elements from the domains of the decision variables on four occasions, in lines **(5)**, **(8)**, **(10)** and **(12)**. To prove total correctness of SEDA it is sufficient to prove that mentioned eliminations do not prevent algorithm in finding the optimal solution.

Proposition 1. *If the values* $\{x_1, \ldots, x_{\theta-1}\}$ *of decision variables are determined in a feasible solution x, then x cannot be better than the current best solution o* if

$$\text{Estimation}(\theta, Dom) \geq minimum. \tag{23}$$

Proof. If the values $\{x_1, \ldots, x_{\theta-1}\}$ of decision variables are determined, function Estimation(θ, *Dom*) is calculated as follows:

$$\text{Estimation}(\theta, Dom) = \sum_{k=1}^{\theta-1} f_k(x_k) + \sum_{k=\theta}^{l} \begin{cases} \min\limits_{y \in D_k} f_k(y) & \text{if} \quad D_k \neq \emptyset, \\ +\infty & \text{if} \quad D_k = \emptyset. \end{cases} \tag{24}$$

If for some $k \in \{\theta, \ldots, l\}$ domain set D_k is empty then $\text{Estimation}(\theta, Dom) = +\infty$, so inequality Estimation(θ, *Dom*) $\geq$ *minimum* trivially holds.

If for all $k \in \{\theta, \ldots, l\}$ domain sets D_k are non-empty, then Estimation(θ, *Dom*) is calculated as follows:

$$\text{Estimation}(\theta, Dom) = \sum_{k=1}^{\theta-1} f_k(x_k) + \sum_{k=\theta}^{l} \min_{y \in D_k} f_k(y). \tag{25}$$

The first sum $\sum\limits_{k=1}^{\theta-1} f_k(x_k)$ in (25) is constant, since values $\{x_1, \ldots, x_{\theta-1}\}$ of decision variables are determined. The second sum $\sum\limits_{k=\theta}^{l} \min\limits_{y \in D_k} f_k(y)$ in (25) estimate the values of for the non-determined decision variables $\{x_\theta, \ldots, x_l\}$ with the minimal values of the objective function in their domains: $\min\limits_{y \in D_k} f_k(y)$. This estimation does not necessarily correspond to the feasible solution, but it has the minimal value. Therefore, any feasible solution has the value of objective function which is greater than or equal to Estimation(θ, *Dom*). Hence, if Estimation(θ, *Dom*) $\geq$ *minimum*, for the determined values of decision variables$\{x_1, \ldots, x_{\theta-1}\}$, there cannot exist a solution *x* containing the determined values for $\{x_1, \ldots, x_{\theta-1}\}$ that is better than the current best solution *o*. □

Consequence 1. The elimination of the elements of domain sets for the non-determined decision variables $\{x_\theta, \ldots, x_l\}$, in lines **(5)** and **(10)** of the SEDA pseudocode, does not prevent SEDA in finding optimal solution.

The eliminations of the elements of domain sets in line **(8)** occurs during the execution of the Sediment(θ, x_θ) function. This function eliminates elements of domain sets that does not fulfill constraints of the problem. In addition, it applies problem specific *look-ahead* techniques for further reduction of the domain sets. We will take for granted that these techniques do not prevent SEDA in finding optimal solution. Hence, the following consequence holds.

Consequence 2. The application of the Sediment(θ, x_θ) function, in line **(8)** of the SEDA pseudocode, does not prevent SEDA in finding the optimal solution.

In order to prove that the domains reductions in line **(12)** do not prevent SEDA finding the optimal solution we first prove the following proposition.

Proposition 2. *If there is an optimal solution x different from the current best solution o and the values* $\{x_1, \ldots, x_\theta\}$ *of decision variables in the optimal solution x are determined, then if we denote by*:

$$\varepsilon = minimum - \text{Estimation}(\theta + 1, Dom), \tag{26}$$

the following holds:

$$\varepsilon > 0 \;\Leftrightarrow\; (\forall k \in \{\theta + 1, \ldots, l\})\; f_k(x_k) \leq \min_{y \in D_k} f_k(y) + \varepsilon. \tag{27}$$

If we use the notation from the description of the CutOff(θ, ε) function in Subsection 2.3, the above equivalence can be formulated as:

$$\varepsilon > 0 \;\Leftrightarrow\; (\forall k \in \{\theta + 1, \ldots, l\})\; x_k \in D'_k. \tag{28}$$

Please cite this article as: S. Kordić et al., Combinatorial approach to exactly solving discrete and hybrid berth allocation problem, Applied Mathematical Modelling (2016), http://dx.doi.org/10.1016/j.apm.2016.05.004

Before the proof of the equivalence, note that, since the first θ decision variables of the optimal solution x are determined, the following must hold:

$$(\forall k \in \{\theta, \ldots, l\})\ D_k \neq \emptyset. \tag{29}$$

Proof. Let us first prove the ($\Leftarrow$) direction. By the definition of the variable *minimum* in Subsection 2.2, the $minimum = f(o_1, \ldots, o_l)$ holds. If there exists an optimal solution x different from the current best solution o, then $f(o_1, \ldots, o_l) > f(x_1, \ldots, x_l)$ i.e.,

$$minimum - f(x_1, \ldots, x_l) > 0 \quad \Leftrightarrow \quad minimum > f(x_1, \ldots, x_l). \tag{30}$$

By the general assumption for the function Estimation$(\theta + 1, Dom)$ and assumption $(\forall k \in \{\theta + 1, \ldots, l\})\ x_k \in D'_k$, it holds:

$$f(x_1, \ldots, x_l) \geq \text{Estimation}(\theta + 1, Dom). \tag{31}$$

From two previous inequalities (30) and (31) we conclude the ($\Leftarrow$) direction of the proposition equivalence (27):

$$minimum - \text{Estimation}(\theta + 1, Dom) > 0 \ \Leftrightarrow \ \varepsilon > 0. \tag{32}$$

The other direction ($\Rightarrow$) of equivalence is proved by *reduction ad absurdum*. First, let us introduce the following notation:

$$N_k = \min_{y \in D'_k} f_k(y), \quad k \in \{\theta + 1, \ldots, l\}; \tag{33}$$

$$N'_k = \min_{y \in D'_k} f_k(y), \quad k \in \{\theta + 1, \ldots, l\}. \tag{34}$$

From the description of SEDA we easily conclude that $(\forall k \in \{\theta + 1, \ldots, l\})\ D'_k \subseteq D_k$ holds, since the functions Sediment(θ, e_θ) and CutOff(θ, ε) eventually only reduce the elements of domains sets. Therefore, the following statement is true:

$$(\forall k \in \{\theta + 1, \ldots, l\})\ N'_k \geq N_k. \tag{35}$$

Suppose that $\varepsilon > 0$ and $(\exists k \in \{\theta + 1, \ldots, l\})\ x_k \notin D'_k$. Let $s \in \{\theta + 1, \ldots, l\}$ be such an index that:

$$x_s \notin D'_s \Leftrightarrow f_s(x_s) \geq \min_{y \in D'_s} f_s(y) + \varepsilon \Leftrightarrow f_s(x_s) \geq N_s + \varepsilon. \tag{36}$$

Since $minimum > f(x_1, \ldots, x_l)$ and taking into account the properties of the above introduced sequences N_k and N'_k, for $k \in \{\theta + 1, \ldots, l\}$ in formulas (35) and (36) the following sequence of equalities and inequalities holds:

$$\begin{aligned}
&minimum > f(x_1, \ldots, x_l) = \\
&= \sum_{k=1}^{l} f_k(x_k) = \sum_{k=1}^{\theta} f_k(x_k) + \sum_{k=\theta+1}^{l} f_k(x_k) = \\
&= \sum_{k=1}^{\theta} f_k(x_k) + \sum_{k=\theta+1}^{s-1} f_k(x_k) + f_s(x_s) + \sum_{k=s+1}^{l} f_k(x_k) \geq \\
&\geq \sum_{k=1}^{\theta} f_k(x_k) + \sum_{k=\theta+1}^{s-1} N'_k + \left(N'_s + \varepsilon\right) + \sum_{k=s+1}^{l} N'_k = \\
&= \sum_{k=1}^{\theta} f_k(x_k) + \sum_{k=\theta+1}^{l} N'_k + \varepsilon \geq \sum_{k=1}^{\theta} f_k(x_k) + \sum_{k=\theta+1}^{l} N_k + \varepsilon = \\
&= \sum_{k=1}^{\theta} f_k(x_k) + \sum_{k=\theta+1}^{l} \min_{y \in D_k} f_k(y) + \varepsilon = \text{Estimation}(\theta + 1, Dom) + \varepsilon = \\
&= \text{Estimation}(\theta + 1, Dom) + minimum - \text{Estimation}(\theta + 1, Dom) = \\
&= minimum.
\end{aligned} \tag{37}$$

If we connect the beginning and the end of the above formula, we obtain $minimum > minimum$, i.e., a contradiction. This concludes the proof of Proposition 2. □

Consequence 3. The application of the CutOff(θ, ε) function, in line **(12)** of the SEDA pseudocode, does not prevent SEDA in finding the optimal solution.

Proof. Let us suppose that there is an optimal solution x different from the current best solution o in some stage of SEDA's work. If the determined values of the decision variables differ from the corresponding values in the optimal solution x, then the optimal solution x cannot be found until SEDA equalises the values of the determined decision variables with the

optimal solution x. In that case any reductions in domain sets, including the CutOff(k, ε) will not have effect on finding the optimal solution.

On the other hand, if the determined values of the decision variables $\{x_1, \ldots, x_\theta\}$ are equal to the corresponding values in the optimal solution x, then Proposition 2 ensures that the application of the function CutOff(k, ε), in line **(12)** of the SEDA pseudocode, does not prevent SEDA in finding the optimal solution. □

Theorem 1. *SEDA is a totally correct algorithm.*

Proof. As it is already mentioned SEDA reduces the domain sets of the decisions variables on four occasions, in lines **(5)**, **(8)**, **(10)** and **(12)** of SEDA pseudocode. Without these reductions it would behave as a *brute force algorithm* searching whole space of feasible solutions. Therefore, it is sufficient and necessary to prove that these reductions do not prevent SEDA in finding optimal solution.

Consequence 1. proves that eliminations in lines **(5)** and **(10)** of SEDA pseudocode do not prevent SEDA in finding optimal solution, **Consequence 2.** proves the same for the line **(8)** and finally **Consequece 3.** provides the proof for the line **(12)**. Therefore, SEDA is a totally correct algorithm. □

2.6.2. General complexity of SEDA

Estimation of the complexity of SEDA is difficult as it is a general algorithm for solving problems of combinatorial type. In the general case it is not possible to determine, even not to estimate, amount of the reduction of domain space done in lines **(5)**, **(8)**, **(10)** and **(12)** of SEDA pseudocode. To illustrate that, let us suppose that all domain sets have cardinality of d. If there are no reductions, then the complexity of the SEDA is $O(d^l)$. That is also complexity of the brute force algorithm. If domain sets are reduced by one element in SEDA, then the complexity of SEDA is $O(d(d-1)(d-2)\cdots(d-l+1)) = O(\frac{d!}{l!})$, providing $d > l$. Finally, if we suppose that cardinalities of domain sets are reduced in SEDA by coefficient p, where $p \in (0, 1)$, then the complexity of SEDA is $O(d \cdot (p \cdot d) \cdot (p^2 \cdot d) \cdots (p^{l-1} \cdot d)) = O(d^l p^{\frac{l(l-1)}{2}})$. These three simple examples demonstrate differences in the complexity of SEDA which may occur depending on the number of reduced elements. In the Subsection 4.2, more precise estimation of SEDA for BAP is given.

2.7. Sedimentation algorithm with estimation & rearrangement heuristic

The efficiency of SEDA depends heavily on the order of the decision variables. Good ordering for the algorithm is when we sort the decision variables in a descending order according to their penalty costs in the optimal solution. The problem is that we do not know this order in advance. To obtain a good initial solution, we run SEDA using a few different orderings for a limited time. Then, we order the decision variables according to their penalty costs in the best solution obtained. Finally, we initialize SEDA with new ordering and run the algorithm until it stops. This simple heuristic yields a significant improvement in the efficiency of SEDA. We name this heuristic the *Estimation & Rearrangement Heuristic* (ERH). When ERH is applied before SEDA, a new algorithm is obtained; we name it the *Sedimentation Algorithm with an Estimation & Rearrangement Heuristic* (SEDA+ERH).

Let us now describe the SEDA+ERH in more details. By a straightforward modification of the described SEDA, algorithm can start from any predefined ordering ω of the decision variables. The variable ω is a permutation of the decision variables indices set $\{1, \ldots, l\}$. Instead of accessing the θth decision variable in the algorithm, we access decision variable $\omega(\theta)$. We refer to this SEDA modification as the SedimentationAlgorithmΩ(l, *Dom*, Ξ, ω, f) function. Also, by a straightforward modification of the described SEDA we delete lines **(19)** "$o = \langle \emptyset \mid i = 1, \ldots, l \rangle$" and **(21)** "*minimum* $= +\infty$" in the pseudo code of the Fig. 3. Instead, we assume that the variables o and *minimum* are global variables.

We assume the following functions and procedures for the SEDA+ERH:

1. RunForLimitedTime(*function, time*) – the function which terminates calculation of the *function* after *time* seconds of execution. It returns a pair *result, finished*, where *result* represents the value of the *function* and *finished* takes the value *true* if the function was completed during the *time* seconds of execution; otherwise, the function returns *false* as a value for the variable *finished*.
2. RandomPermutation(ω) – the function which returns a random permutation of the sequence ω.
3. SortAccordingTo(C) – the function which, for input sequence $C = \{c_1, \ldots, c_l\}$, returns the sequence ω such that:

$$c_{\omega(1)} \geq c_{\omega(2)} \geq \ldots \geq c_{\omega(l)}. \tag{38}$$

The input variable *SEDArestarts* determines the number of estimations, and the input variable *EstT* determines the estimation duration in seconds. The pseudo code of SEDA+ERH is given in Fig. 3.

Lines **(2)–(4)** initialize global variables in the algorithm. In lines **(5)–(10)**, the algorithm makes estimations. If SEDA reaches a feasible solution, the solution is saved in the global variable o. The value of the global variable *minimum* is also changed accordingly. If the estimations find at least one feasible solution, the value of *minimum* is less than $+\infty$, (line **(11)**). Then, we can sort the vessels according to their penalty costs in the feasible solution o by rearranging ω in lines **(12)–(13)**. Otherwise, we use the simplest ω, line **(15)**. Finally, the algorithm runs SEDA starting with the ω ordering of the vessels and, after the end of its execution, returns the optimal solution of BAP. The estimation of the complexity of SEDA+ERH is given in Subsection 4.2.

Please cite this article as: S. Kordić et al., Combinatorial approach to exactly solving discrete and hybrid berth allocation problem, Applied Mathematical Modelling (2016), http://dx.doi.org/10.1016/j.apm.2016.05.004

```
1  EstimationRearrangeHeuristic(l, Dom, Ξ, ω, f, SEDArestarts, EstT)
2      minimum = +∞
3      o = ⟨ ∅ | i = 1, ... l ⟩
4      ω = {1, ... , l}
5      for i = 1 to SEDArestarts do
6          ⟨⟨o, minimum⟩, completed⟩ =
7              RunForLimitedTime(SedimentationAlgorithmΩ(l, Dom, Ξ, ω, f), EstT)
8          if completed then return ⟨o, minimum⟩ endif
9          ω = RandomPermutation(ω)
10     endfor
11     if minimum < +∞ then
12         C = ⟨ f_k(o_k) | k ∈ {1, ... , l}⟩
13         ω = SortAccordingTo(C)
14     else
15         ω = {1, ... , l}
16     endif
17     return SedimentationAlgorithmΩ(l, Dom, Ξ, ω, f)
18 end
```

Fig. 3. The Estimation and Rearrange Heuristic.

2.8. Advantages and disadvantages of SEDA and SEDA+ERH

SEDA and SEDA+ERH are general branch-and-bound algorithms for solving combinatorial type of problems. Therefore, they have some general and some specific advantages and disadvantages. We will briefly list both properties.

Advantages of SEDA and consequently of SEDA+ERH are the following:

1. It can solve various type of problems. Author used it to solve Knapsack Problem, Robust Knapsack Problem, Beth Allocation Problem, Berth Allocation and Crain Assignment Problem, MAX-SAT, MAX-k-SAT, etc. Most of the problems can be relatively easy modeled for SEDA.
2. SEDA is proven to be totally correct, it will provide optimal solution if it exists. Also, it will halt if the optimal solution does not exist.
3. It can solve efficiently some type of combinatorial problems.
4. It have been designed to easily incorporate problem specific look-a-head techniques, which can significantly improve its efficiency.
5. It can be relatively easy implemented in the variety of programing languages.

Disadvantages of SEDA and consequently of SEDA+ERH are the following:

1. It can solve only problems with finite decision variables domains set. Moreover, for big finite domains set, SEDA will use a lot of memory for storing them, which will affect its performance.
2. In general, it is not easy to establish more precise its complexity, other than general exponential complexity of a combinatorial algorithm. The specific problem complexity for solver based on SEDA heavily depends on the modelling of the problem and look-a-head techniques implemented.
3. Same as for all combinatorial algorithms, SEDA has worst case problems, which are very hard (time consuming) for SEDA to solve.

The above lists of advantages and disadvantages are not complete, since this approach needs further developments. Both lists can serve as a starting points for future SEDA improvements.

3. Berth allocation problem formulation

The process of assigning vessels to berths is very complex; it consists of several problems, such as the *Berth Planning Problem, Berth Allocation Problem, Quay Crane Assignment Problem, and the Quay Crane Scheduling Problem*. These problems can be treated separately like Cordeau et al. [11], Imai et al. [28], Hansen et al. [17], Fu & Diabat [34] etc.; or jointly like Meisel and Bierwirth [35] and Vacca et al. [5]. In Dragović et al. [36], a comprehensive analytical description of vessel-berth link performance is given.

In this work, we concentrate on *minimum-cost* DBAP and HBAP. The model for these problems is a sub-model of the Rashidi & Tsang's [32] model. We use only the part of that model referring to the berth allocation and reproduced it in the

Please cite this article as: S. Kordić et al., Combinatorial approach to exactly solving discrete and hybrid berth allocation problem, Applied Mathematical Modelling (2016), http://dx.doi.org/10.1016/j.apm.2016.05.004

rest of this section, using exactly the same notation introduced in [32]. We omit the part of the model related to the crane assignment, assuming that each berth has exactly one crane.

The combinatorial approach adopted in this work for solving DBAP and HBAP directly uses only the input data and objective function of the Rashidi & Tsang's sub-model. The algorithms do not use other elements, including the decision variables and constraints, in the form presented in Rashidi & Tsang's [32]. Instead, the decision variables are implemented as functions, and they are used only for the calculation of the objective function value. In our algorithms, the constraints are implemented in the steps for constructing and maintaining the solution space in which the search for the optimal solution is conducted.

3.1. Assumptions

We made the following assumptions about berthing of vessel in container port:

Assumption 1. Each vessel has a pre-determined berthing time period. A cost penalty applies if the vessel berths early, tardy or departs late.

Assumption 2. Each vessel has a preferred berthing location. Preferred location of the vessel depends on some preference like: location of the storage area where inbound/outbound containers of the vessel are stacked, depth of water, strength and direction of currents or some other preference. A penalty cost applies if the vessel does not berth on its preferred location.

Assumption 3. At each time period only one container can be loaded/unloaded on a berth. Moreover, we assume that at each berth there is exactly one crane available for loading/unloading of the vessel.

The *Assumption 3* is necessary to establish compatibility with the Rashidi & Tsang's [32] model for BAP, since here we deal only with the BAP without crane assignment.

3.2. Input data

Our model and algorithms use the input data listed below:

T: The total number of time periods in the planning horizon.
m: The number of berths in the port.
l: The number of vessels in the planning horizon.
vessel: The sequence of data relevant for vessels, which has the following structure:

$$vessel = \{(\mathrm{ETA}_k,\ a_k,\ b_k,\ d_k,\ s_k, C_{1k},\ C_{2k},\ C_{3k},\ C_{4k}) \mid k = 1, \ldots, l\}. \tag{39}$$

The elements of a vessel 9-tuple represent the following data for each vessel:

ETA_k: The expected time of arrival of a $vessel_k$.
a_k: The processing time of the $vessel_k$.
b_k: The length of the $vessel_k$, expressed in the number of berths.
d_k: The required departure time for the $vessel_k$.
s_k: The least-cost berthing location of the $vessel_k$.
C_{1k}: The penalty cost for the $vessel_k$ if the vessel cannot dock at its preferred berth.
C_{2k}: The penalty cost for the $vessel_k$ per unit time for arrival before ETA_k.
C_{3k}: The penalty cost for the $vessel_k$ per unit time for arrival after ETA_k.
C_{4k}: The penalty cost for the $vessel_k$ per unit time of delayed departure after d_k.

In this paper we present the exact method for solving DBAP and HBAP. Since, in DBAP only one berth is allocated to a vessel, then the value of the b_k parameters must be 1, for all vessels. In the case of HBAP, the values of the b_k parameters range from 1 to m.

3.3. Decision variables and domains

The formulation of BAP given in Rashidi & Tsang's [32] uses decision variables. Although they are not important for combinatorial algorithms, we list them here for a proper definition of BAP:

At_k: The berthing time of a $vessel_k$ to the corresponding berth, $At_k \in \{1, \ldots, T\}$.
Dt_k: The departing time of the $vessel_k$ from the corresponding berth, $Dt_k \in \{1, \ldots, T\}$.
Bi_k: The lowest berth index allocated to the $vessel_k$ from the corresponding berth, $Bi_k \in \{1, \ldots, m\}$.
X_{itk}: If berth i at the time t is allocated to the $vessel_k$, X_{itk} takes the value 1; otherwise, its value is 0. Obviously, $X_{itk} \in \{0, 1\}$.

Please cite this article as: S. Kordić et al., Combinatorial approach to exactly solving discrete and hybrid berth allocation problem, Applied Mathematical Modelling (2016), http://dx.doi.org/10.1016/j.apm.2016.05.004

3.4. Constraints

A feasible solution of BAP is subject to two sets of constraints.

Constraints 1. At a time t, each berth can be assigned to only one vessel:

$$(\forall i \in \{1,\ldots,m\})(\forall t \in \{1,\ldots,T\}) \sum_{k=1}^{l} X_{itk} \leq 1. \tag{40}$$

Constraints 2. A berth is allocated to the vessel only between vessels arrival and departure times and where vessel fits:

$$\begin{aligned}&(\forall k \in \{1,\ldots,l\})(\forall t \in \{1,\ldots,T\})(\forall i \in \{1,\ldots,m\})\\ &(At_k \leq t \leq Dt_k \ \wedge \ \ Bi_k \leq i < Bi_k + b_k \ \Rightarrow X_{itk} = 1) \wedge\\ &(t < At_k \ \vee \ Dt_k < t \ \vee \ i < Bi_k \ \vee \ Bi_k + b_k \leq i \ \Rightarrow X_{itk} = 0).\end{aligned} \tag{41}$$

3.5. Objective function

Let us first introduce the auxiliary variable Z_k, which represents the sum of the absolute distances between the preferred location of the $vessel_k$ and the berths allocated to the $vessel_k$:

$$Z_k = \sum_{t=1}^{T}\sum_{i=1}^{m} \begin{cases} |i - s_k| & \text{if} \quad X_{itk} = 1, \\ 0 & \text{if} \quad X_{itk} = 0. \end{cases} \tag{42}$$

The objective function for the minimization of the port penalty cost can be formulated as follows:

$$VesselCost = \sum_{k=1}^{l} \left\{ C_{1k} Z_k + C_{2k}(\mathrm{ETA}_k - At_k)^+ + C_{3k}(At_k - \mathrm{ETA}_k)^+ + C_{4k}(Dt_k - \mathrm{d}_k)^+ \right\}. \tag{43}$$

The objective function minimize the cost of berth position, vessels waiting time, speed up time and tardiness. According to Meisel [2,3] and the above problem formulation, our BAP can be classified as:

$$disc\ or\ hybrid|stat|fix|\Sigma(w_1 wait + w_2 speed + w_3 tard + w_4 pos).$$

Original formulation of the objective function (43) in Rashidi & Tsang's [32] model does not follow the order of *"performance measure"* in Maisel [2,3] classification. In (43) order of *"performance measure"* is $\Sigma(w_4\ pos + w_1\ wait + w_2\ speed + w_3\ tard)$.

From the above problem formulation, it is also evident that we consider *minimum-cost* variant of the BAP problem either as DBAP or as HBAP.

The described formulation of the problem is used with the CPLEX 11.2 commercial MIP solver for the comparison with the SEDA and the SEDA+ERH proposed here.

3.6. Few notes on BAP modeling and Rashidi & Tsang's model in particular

Rashidi & Tsang's [32] model is intended for discrete, continuous and hybrid form of BAP, as are the most of the other models of BAP. Mathematically model remains the same, only the interpretation of the berth and time units differ. The interpretation for the discrete, continuous and hybrid form of BAP is given in Table 1.

From the Table 1. it is evident that CBAP is not truly continuous. It is also discrete, but the interpretation of berth and time unit is different, as it is previously stated.

Table 1
The interpretation of berth and time units in BAP.

Form of BAP	Berth unit interpretation	Time unit interpretation
Discrete (DBAP)	The quay is partitioned into a number of sections, called berths. Only one vessel can be served at each single berth i.e., vessel is serviced on one and only one berth. Therefore $b_k = 1$, for all vessels in the case of DBAP.	Time unit usually takes a few hours' time interval. In our test instances we use 3-h time interval to be time unit. Plan horizon is usually one or two weeks i.e. 56 or 112 time units, see [6].
Continuous (CBAP)	The quay is dividend in units of the certain length, in [2,3] the length of the unit is 10 m. For each vessel b_k is calculated as the length of the vessel measured in the segments of 10 m.	Time unit is usually 1 h, see [2,3].
Hybrid (HBAP)	As in the case of DBAP the quay is partitioned into a number of sections. Large vessels can occupy more than one berth. In our test instances, in the case of HBAP, vessels can occupy up to 3 berths. Some models of HBAP allow small vessels to share a berth. That is not the case with Rashidi & Tsang's model.	Same as for DBAP, time unit is usually a few hours' time interval. In our test instances we use 3-h time interval and time horizon of one or two weeks, see [6].

Please cite this article as: S. Kordić et al., Combinatorial approach to exactly solving discrete and hybrid berth allocation problem, Applied Mathematical Modelling (2016), http://dx.doi.org/10.1016/j.apm.2016.05.004

Berths										
m	0	0	0	0	...	0	0	0	0	
⋮	⋮	⋮	...	⋮	...	⋮	...	⋮	⋮	
$Bi_k + b_k - 1$	0	0	0	1	...	1	0	0	0	
⋮	⋮	⋮	⋮	⋮	...	⋮	⋮	⋮	⋮	
Bi_k	0	0	0	1	...	1	0	0	0	
⋮	⋮	⋮	...	⋮	...	⋮	...	⋮	⋮	
2	0	0	0	0	...	0	0	0	0	
1	0	0	0	0	...	0	0	0	0	
$\boldsymbol{X_{itk}}$	1	2	...	At_k	...	$At_k + a_k - 1$	Dt_k	...	T	Time

Fig. 4. Representation of a X_{itk} decision variable of the BAP model.

In the case of DBAP and HBAP berths are regarded as having the same length (size), although physical length may not be exactly the same size. In the case of HBAP, vessels occupying more than one berth will always add to penalty cost, even if the least-cost berth location is allocated to them. That is the consequence of the definition of the auxiliary variable Z_k in the formula (42). This feature of the Rashidi & Tsang's [32] model is justified by the necessity for additional transportation costs for moving containers from the neighboring berths to the least-cost berth of the vessels.

4. BAP modeling for SEDA

In order to solve BAP by SEDA we have to model BAP to meet SEDA requirements described in Subsection 2.1 and 2.3. The BAP model exposed in Section 3 is the starting point in BAP modeling for SEDA.

4.1. Model of BAP for SEDA

The decision variables for each vessel are: At_k, Dt_k, Bi_k and X_{itk} for $k \in \{1, \ldots, l\}$. Fig. 4 represents the values of a decision variable X_{itk} from which we can easily calculate At_k, Dt_k and Bi_k decision variables for a vessel.

From the Fig. 4 it is evident that it is sufficient to know the values of At_k and Bi_k to calculate Dt_k and X_{itk} for $k \in \{1, \ldots, l\}$:

$$Dt_k = At_k + a_k; \tag{44}$$

$$X_{itk} = \begin{cases} 1 & \text{if} \quad At_k \le t < At_k + a_k \ \wedge \ Bi_k \le i < Bi_k + b_k, \\ 0 & \text{if} \quad \text{otherwise.} \end{cases} \tag{45}$$

Therefore, it is sufficient to use a pair (At_k, Bi_k) to uniquely represent the position of the vessel in the berth-time grid. The pair (At_k, Bi_k) will be called the vessel's position in the time-berth grid. This fact is used for the definition of the domain set $Dom = \{D_1, \ldots, D_l\}$. Since we can represent vessels as rectangles of $a_k \times b_k$ dimensions in the grid of dimension $T \times m$, the domain sets D_k for $k \in \{1, \ldots, l\}$ are:

$$D_k = \{ \ (t, b) \mid t \in \{1, \ldots, T - a_k + 1\} \ \wedge \ b \in \{1, \ldots, m - b_k \ + 1\}\}. \tag{46}$$

Each vessel will be associated with one decision variable, so that the set of decision variables in BAP modeling for SEDA is $X = \{x_1, \ldots, x_l\}$. SEDA finds a value for each decision variable in the corresponding decision variable set i.e., $x_k = (t, b) = (At_k, Bi_k) \in D_k$, for $k \in \{1, \ldots, l\}$.

We define f_k for $k \in \{1, \ldots, l\}$ as:

$$f_k((At_k, Bi_k)) = C_{1k}Z_k + C_{2k}(\mathrm{ETA}_k - At_k)^+ + C_{3k}(At_k - \mathrm{ETA}_k)^+ + C_{3k}(Dt_k - \mathrm{d}_k)^+. \tag{47}$$

The auxiliary variable Z_k is calculated as it is described in Subsection 3.5. The objective function f, we define as follows:

$$f(x_1, \ldots, x_l) = f((At_1, Bi_1), \ldots, (At_l, Bi_l)) = \sum_{k=1}^{l} f_k((At_k, Bi_k)) = \sum_{k=1}^{l} f_k(x_k). \tag{48}$$

Finally, we define set of heuristic relations $\Xi = \{\prec_1, \ldots, \prec_l\}$. For $k \in \{1, \ldots, l\}$ and (t_1, b_1), $(t_2, b_2) \in D_k$ we define $\prec_k$ as total order closure of the relation $\ll_k$ defined as follows:

$$\begin{aligned} &(t_1, b_1) \ll_k (t_2, b_2) \ \Leftrightarrow \\ &\quad f_k((t_1, b_1)) < f_k((t_2, b_2)) \ \vee \\ &\quad (f_k((t_1, b_1)) = f_k((t_2, b_2)) \ \wedge \ |b_1 - s_k| < |b_2 - s_k| \ ). \end{aligned} \tag{49}$$

Please cite this article as: S. Kordić et al., Combinatorial approach to exactly solving discrete and hybrid berth allocation problem, Applied Mathematical Modelling (2016), http://dx.doi.org/10.1016/j.apm.2016.05.004

Berths											
m											
⋮						⋮					
s_k+2				4	3	2	3	4			
s_k+1				3	2	1	2	3			
s_k			...	2	1	0	1	2	...		
s_k-1				3	2	1	2	3			
s_k-2				4	3	2	3	4			
⋮						⋮					
2											
1											
f_k	1	2	...	ETA_k-2	ETA_k-1	ETA_k	ETA_k+1	ETA_k-2	...	T	Time

Fig. 5. Representation of a f_k function.

The procedures and functions: FindSolution(θ, *Dom*), Estimation(θ, *Dom*), NonDetVarEstimation(θ), ReportSolution() and CutOff(θ, ε) are exactly the same as described in Subsection 2.3. The function Sediment(θ, x_θ) deletes from the domains of the undetermined decision variables all the positions occupied by setting the vessel θ at the position x_θ.

From the described BAP modelling for SEDA and the SEDA properties, we can easily conclude that this modeling meets all the SEDA requirements. Therefore, when these two combine we get an exact solver for BAP.

4.2. Complexity of SEDA for BAP

To estimate the complexity of SEDA for BAP we consider the case of DBAP. Similar estimation can be conducted for the case of HBAP, therefore it is omitted from this paper. Without losing generality we can assume that processing time for each vessel is 1-time unit. We also assume that the penalty costs of not berthing the vessel at its preferred berth, earliness and tardiness is 1. The penalty cost for delay departure is neglected to make estimation easier. These assumptions we can sum in the following formula:

$$(\forall k \in \{1,\ldots,l\}) \quad a_k = 1 \ \wedge \ C_{1k} = 1 \ \wedge \ C_{2k} = 1 \ \wedge \ C_{3k} = 1 \ \wedge \ C_{4k} = 0. \tag{50}$$

Input data satisfying the above condition represent worst case scenario of DBAP, since 1-time unit processing time vessels makes minimal reductions in the domain sets of its positions. For such scenario a function f_k for a vessel $k \in \{1,\ldots,l\}$ is shown in the Fig. 5.

From the above figure we trivially calculate that there is one position with cost (value of function f_k) equal to 0 and $4i$ positions with the cost (value of function f_k) equal to i, for $i > 1$.

For establishing the complexity of the algorithm we denote by M the estimation of the BAP solution that has value greater than or equal to optimum. The complexity of the SEDA for BAP can be expressed as the number of leaves (branches) in the solution search space tree constructed during the work of SEDA. The number of the leaves in the solution search space tree for l vessels and the estimation M we denote by $\Psi(l, M)$. The $\Psi(l, M)$ can be calculated recursively in the following way:

$$\begin{aligned} &\Psi(0,M) = 1, \qquad \Psi(l,0) = 1 \qquad \text{and} \\ &\Psi(l,M) = \Psi(l-1,M) + 4\sum_{i=1}^{M} i\Psi(l-1,M-i). \end{aligned} \tag{51}$$

The leaf of the solution search space tree is reached if $l = 0$, therefore $\Psi(0, M) = 1$. If the estimation of the optimal solution is $M = 0$, then search is concluded, since solution of the problem consists of the least cost positions of the vessels, therefore also $\Psi(l, 0) = 1$.

The number of the leaves for the $\Psi(l, M)$ is calculated as the number of the leaves in the solution search space subtree if vessel l takes position with the cost 0 i.e. $\Psi(l-1, M)$, plus all the solution search space subtrees, if vessel l, takes positions with the cost i, where i ranges from 1 to maximally M. If vessel l takes positions with the cost i, then the number of the leaves of the solution search space subtree is calculate as $\Psi(l-1, M-i)$. Since there are $4i$ positions with the cost i, for $i \geq 1$, we easily derive the sum in the formula (51).

Lemma 1. *The* $\Psi(1, M)$ *can be calculated as* $\Psi(1, M) = 2M^2 + 2M + 1$.

Proof. From the definition of the $\Psi(l, M)$, given in formula (51), we calculate:

$$\begin{aligned} \Psi(1,M) &= \Psi(0,M) + 4\sum_{i=1}^{M} i\Psi(0,M-i) = 1 + 4\sum_{i=1}^{M} i * 1 = 1 + 4\sum_{i=1}^{M} i = \\ &= 1 + 4 \cdot \tfrac{M(M+1)}{2} = 1 + 2M(M+1) = 2M^2 + 2M + 1. \end{aligned} \tag{52}$$

□

Lemma 2. *Let* $P(M) = \alpha_n M^n + \alpha_{n-1} M^{n-1} + \ldots \alpha_1 M + \alpha_0$ *be a polynomial of the degree* n, i.e., $\deg(P) = n$. *Then the degree of the polynomial*:

$$Q(M) = \sum_{i=1}^{M} iP(M-i), \tag{53}$$

is $\deg(Q(M)) = \deg(P(M)) + 2 = n + 2$.

Proof. By expanding expression $iP(M-i)$ in formula (53), we get M^n and i^{n+1}, as the highest degrees of M and i. The M is a free variable in the sum, so, we can express $iP(M-i)$ as a polynomial of i as:

$$iP(M-i) = \beta_{n+1} i^{n+1} + \beta_n i^n + \ldots \beta_1 i, \tag{54}$$

where coefficients β_j, $j \in \{1, \ldots n+1\}$ depend on M and α_j, $j \in \{0, \ldots n\}$. Polynomial $Q(M)$, then can be written as:

$$Q(M) = \sum_{i=1}^{M} \left(\beta_{n+1} i^{n+1} + \beta_n i^n + \ldots \beta_1 i\right) = \beta_{n+1} \sum_{i=1}^{M} i^{n+1} + \beta_n \sum_{i=1}^{M} i^n + \ldots + \beta_1 \sum_{i=1}^{M} i. \tag{55}$$

From the above formula it is evident that $\deg(Q(M)) = \deg(\sum_{i=1}^{M} i^{n+1})$. Using the well-known Faulhaber's formula for the sum of the $n+1$ powers of the first M positive integers we obtain:

$$\sum_{i=1}^{M} i^{n+1} = \frac{1}{n+2} \sum_{j=0}^{n+1} (-1)^j \binom{n+2}{j} B_j M^{n+2-j}, \tag{56}$$

where B_j are first Bernoulli numbers, with $B_1 = -\frac{1}{2}$. From the above formula it is evident that $\deg(Q(M)) = \deg(\sum_{i=1}^{M} i^{n+1}) = \deg(M^{n+2}) = n+2$. □

Lemma 3. *The* $\deg(\Psi(l, M)) = 2l$, *for any* $M \geq 0$.

Proof. The proof of the lemma is conducted by mathematical induction.

(1) For $l = 1$, by the Lemma 1 we have $\Psi(1, M-i) = 2M^2 + 2M + 1$, from which we conclude that $\deg(\Psi(1, M)) = 2$.
(2) The induction hypothesis: $\deg(\Psi(l, M)) = 2l$, for any $M \geq 0$.
(3) The inductive step: $\deg(\Psi(l+1, M)) = 2(l+1)$.

Let us expand $\Psi(l+1, M)$:

$$\Psi(l+1, M) = \Psi(l, M) + 4 \sum_{i=1}^{M} i\Psi(l, M-i). \tag{57}$$

By the induction hypothesis the following holds:

$$\deg(\Psi(l, M)) = 2l \quad \wedge \quad (\forall i \in \{1, \ldots, M\})\ deg(\Psi(l, M-i)) = 2l. \tag{58}$$

Then, by the Lemma 2 we obtain that $\deg(\sum_{i=1}^{M} i\Psi(l, M-i)) = 2l + 2$.

Finally, we get:

$$\begin{aligned} \deg(\Psi(l+1, M)) &= \max\left(\deg(\Psi(l, M)),\ \deg\left(\sum_{i=1}^{M} i\Psi(l, M-i)\right)\right) \\ &= \max(2l,\ 2l+2) = 2l+2 = 2(l+1). \end{aligned} \tag{59}$$

□

Theorem 3. *The complexity of the SEDA is* $O(M^{2l})$.

Proof. The proof is immediate consequence of the fact that $\Psi(l, M)$ is polynomial of M and Lemma 3 which states that $\deg(\Psi(l, M)) = 2l$. □

Complexity $O(M^{2l})$ of SEDA is achieved for the worst case. We neglected delay departure penalty and elimination of the elements of domain sets of the vessels. Delay departure penalty can be included into the calculus of estimation, but the estimation of complexity remains the same $O(M^{2l})$. Therefore, it is not presented it in this paper. The boundaries of planning horizon and the number of berths in this estimation are also neglected. The complexity estimation is obtained under the assumption that they are unlimited. All this factors influence runtimes, so in the most of the tests and real-life examples the maximal complexity is reached only for a large number of vessels.

Complexity depends on the estimation M of the optimal solution. The quality of the estimation effects the runtime of SEDA. We conducted complexity estimation for the fixed value of M, although that is not the general case. SEDA finds initial suboptimal solution very quickly and then improves it until it reaches optimal solution. The most of the runtime, SEDA

Please cite this article as: S. Kordić et al., Combinatorial approach to exactly solving discrete and hybrid berth allocation problem, Applied Mathematical Modelling (2016), http://dx.doi.org/10.1016/j.apm.2016.05.004

uses to prove that there is no better solution than the optimal solution. Hence, the quality of the initial estimation M is important, but not critical for the work of SEDA.

The SEDA+ERH complexity is much smaller than the complexity of the sole SEDA. ERH puts ahead vessels which contribute in the value of objective function with penalties greater than 0. Since these vessels are at the beginning of the search they reach their optimal values sooner than other vessels. Because they participate in the value of the objective function with the values greater than 0, they reduce search space for other vessels coming after them. Once the optimal values are reached, the only possible positions for the remaining vessels are those with the value 0 of the objective function. Therefore, only one branch is constructed and the search for the optimal solution is complete. This is the most effective situation when ERH provides sequence of vessels having at the beginning all the vessels which contribute in the value of objective function with penalties greater than 0. Although the current version of ERH is not much effective in the cases with a big number of vessels, nevertheless it significantly improves the running time of SEDA.

Having in mind the above mentioned features of ERH, if ω is a sequence provided by ERH, o is the optimal solution of the problem and k_ω is defined as:

$$k_\omega = \max\left\{i \in \{1,\ldots,l\} \mid f_{\omega(i)}(o_i) > 0\right\}, \tag{60}$$

then the complexity of SEDA+ERH is $O(M^{2k_\omega})$. Sometimes it is more convenient to consider quotient $D_\omega = l/k_\omega$, which we named *slowdown coefficient*. The complexity of SEDA+ERH can be equivalently formulated as $O(M^{\frac{2l}{D_\omega}})$. Usually D_ω ranges between 2 and 5, which explains much better performance of SEDA+ERH over SEDA.

5. Computational results

In this section, we present the test instances and computational results of SEDA and SEDA+ERH. Finally, we select 10 examples and compare the runtimes of both sedimentation algorithms with that of the CPLEX commercial solver.

5.1. Test instances

The experimental evaluation is performed on three classes of instances that are similar to those introduced by Giallombardo et al. [6]. We consider test instances with 5, 8 and 13 berths and time horizons of 1 or 2 weeks. The time horizon is divided into 3-h time units. Thus, one week has 56 time units and two weeks are divided into 112 time units. The number of vessels ranges from 5 to 65, with an increment of 5 vessels, and it is specific for each test class we consider.

The classes of test instances are the following:

Class I: 5 berths, 1 week, and 5 to 30 vessels.
Class II: 8 berths, 2 weeks, and 5 to 60 vessels.
Class III: 13 berths, 2 weeks, and 5 to 40 vessels.

The information required to specify various types of vessels are presented in Table 1. The specifications resemble those used by Meisel [2,3]; however, here they are adjusted to DBAP and HBAP. Three types of vessels are present in the test population: feeder, medium and mega. For each type, the corresponding percentage of the test population, handling time range, penalty amounts (in units of US$ 1000) and number of berths occupied in the case of DBAP and HBAP are listed in Table 2.

The distribution of the least-cost berthing location for vessels is homogeneous. For each instance and the number of vessels, 500 tests were randomly generated. We recorded the percentage of the tests that were solved in a half-an-hour period, i.e., 1800 s. For all the tests that were solved in a half-an-hour period, we also recorded the minimum, average and maximum time required to find the solution. Tests that were not solved in a half-an-hour period were interrupted and they are not included into the calculation of the minimal, average and maximal time for finding solution. All the times in the following tables are expressed in seconds.

Number of estimations of the optimal solution and their duration in ERH depended on the number of vessels l. Number of estimations was set to $10 + 2l$. Duration of the estimation, for any l, never exceeded 0.03 s. Overall duration of the ERH in all test instances for each example was always less than 4.2 s. The running times of ERH were included in the half-an-hour time period limitation.

Table 2
Test vessel specifications.

Size, handling times, penalties and number of berths for test vessels								
Vessel type	*Percentage of the test population*	*Handling time range*	C_1	C_2	C_3	C_4	*Number of berths DBAP*	*Number of berths HBAP*
Feeder	60%	1–3	2	3	3	9	1	1
Medium	30%	4–5	3	6	6	18	1	2
Mega	10%	6–8	4	9	9	27	1	3

Please cite this article as: S. Kordić et al., Combinatorial approach to exactly solving discrete and hybrid berth allocation problem, Applied Mathematical Modelling (2016), http://dx.doi.org/10.1016/j.apm.2016.05.004

Table 3
Computational results for the Class I test instances of DBAP.

l Number of vessels	Class I: 5×56 | 500 samples | DBAP									
	Sedimentation Algorithm					Sedimentation Algorithm with Estimation & Rearrangement Heuristic				
	$t \leq ½h$	min	avg	max	$½h < t$	$t \leq ½h$	min	avg	max	$½h < t$
5	100.0%	0.000	**0.020**	0.047	0.0%	100.0%	0.000	**0.018**	0.063	0.0%
10	100.0%	0.015	**0.027**	0.078	0.0%	100.0%	0.015	**0.024**	0.047	0.0%
15	100.0%	0.015	**0.044**	1.531	0.0%	100.0%	0.015	**0.032**	0.157	0.0%
20	99.8%	0.018	**0.705**	63.999	0.2%	100.0%	0.015	**0.058**	0.218	0.0%
25	96.8%	0.015	**24.828**	1158.455	3.2%	100.0%	0.021	**0.149**	7.410	0.0%
30	–	–	**–**	–	–	100.0%	0.031	**1.328**	80.444	0.0%
35	–	–	**–**	–	–	96.6%	0.047	**17.311**	1399.490	3.4%

Table 4
Computational results for the Class II test instances of DBAP.

l Number of vessels	Class II: 8×112 | 500 samples | DBAP									
	Sedimentation Algorithm					Sedimentation Algorithm with Estimation & Rearrangement Heuristic				
	$t \leq ½h$	min	avg	max	$½h < t$	$t \leq ½h$	min	avg	max	$½h < t$
5	100.0%	0.000	**0.021**	0.094	0.0%	100.0%	0.003	**0.018**	0.034	0.0%
10	100.0%	0.015	**0.030**	0.110	0.0%	100.0%	0.014	**0.022**	0.047	0.0%
15	100.0%	0.015	**0.036**	0.172	0.0%	100.0%	0.015	**0.030**	0.094	0.0%
20	100.0%	0.022	**0.056**	4.115	0.0%	100.0%	0.015	**0.043**	0.485	0.0%
25	100.0%	0.030	**3.100**	1278.34002	0.0%	100.0%	0.031	**0.069**	0.772	0.0%
30	100.0%	0.031	**5.580**	1127.433	0.0%	100.0%	0.031	**0.132**	0.843	0.0%
35	98.0%	0.014	**16.227**	998.805	2.0%	100.0%	0.031	**0.300**	1.469	0.0%
40	–	–	**–**	–	–	100.0%	0.046	**0.606**	1.875	0.0%
45	–	–	**–**	–	–	100.0%	0.047	**0.903**	2.273	0.0%
50	–	–	**–**	–	–	100.0%	0.047	**1.518**	22.098	0.0%
55	–	–	**–**	–	–	100.0%	0.066	**4.453**	385.473	0.0%
60	–	–	**–**	–	–	100.0%	0.240	**6.510**	780.002	0.0%
65	–	–	**–**	–	–	99.4%	0.407	**41.819**	1753.066	0.6%

5.2. Comparison between SEDA And SEDA+ERH

SEDA and SEDA+ERH were coded in the *C* programming language. Code was compiled by *Microsoft C/C++ Optimizing Compiler version 18.00.31101 for x86*. The tests were conducted on a computer with an *Intel Core i7-4500 U @* 1.80 GHz—2.40 GHz CPU and 8GB of RAM running the *Microsoft Windows 8.1* 64-bit operating system.

Table 3 shows the computational results for the Class I test instances of DBAP. In the trivial cases of 5 and 10 vessels, both algorithms perform equally well. In the cases of 25 vessels, SEDA+ERH solves problems on average 166.63 times faster than SEDA. The results of the SEDA execution for the cases with the number of vessel larger than or equal to 30 are not presented in Table 2 because the time needed for solving 500 examples was too long. Note that the average runtime of SEDA+ERH for 35 vessels is still lower than the average runtime of SEDA for 25 vessels. Also, the percent of problems solved in a half-an-hour period is almost the same if we compare SEDA for 25 vessels (96.8%) and SEDA+ERH for 35 vessels (96.6%).

Table 4 shows the computational results for the Class II test instances of DBAP. In the trivial cases of 5, 10, 15 and 20 vessels, both algorithms perform equally well. In the cases of 25 or more vessels, SEDA+ERH solves problems faster than SEDA. In the case of 25 vessels it is 44.92 times faster, in the case of 30 vessels it is 42.27 times faster and in the case of 35 vessels it is 54.09 times faster. The results of the SEDA execution for the cases with the number of vessels greater than or equal to 40 are not presented in Table 3 because the time needed for solving 500 examples was too long. Note that the average runtime of SEDA+ERH for 55 vessels is still lower than the average runtime of SEDA for 30 vessels. The percentage of problems solved in a half-an-hour period of SEDA+ERH for 60 vessels is 100.0%, while SEDA can solve 100.0% problems only for 30 or less vessels. The results in Table 3 prove the ability of SEDA+ERH of solving DBAP much faster than SEDA: for the approximately same amount of time, SEDA+ERH can solve problems with twice as many vessels.

Table 5 shows the computational results for the Class II test instances of HBAP. Only in the trivial cases of 5 vessels, both algorithms perform equally well. In the cases of 10 or more vessels, SEDA+ERH solves problems faster than SEDA. In the case of 10 vessels it is 77.33 times faster and in the case of 15 vessels it is 27.06 times faster. In the case of 20 vessels it is at least 286.916 times faster, since the percentage of problems solved in a half-an-hour period by SEDA is 97.2% and percentage of problems solved by SEDA+ERH is 100%. The results of the SEDA execution for the cases with the number of

Please cite this article as: S. Kordić et al., Combinatorial approach to exactly solving discrete and hybrid berth allocation problem, Applied Mathematical Modelling (2016), http://dx.doi.org/10.1016/j.apm.2016.05.004

Table 5
Computational results for the Class II test instances of HBAP.

l Number of vessels	Class II: 8×112 | 500 samples | HBAP									
	Sedimentation Algorithm					Sedimentation Algorithm with Estimation & Rearrangement Heuristic				
	$t \le \frac{1}{2}h$	min	avg	max	$\frac{1}{2}h < t$	$t \le \frac{1}{2}h$	min	avg	max	$\frac{1}{2}h < t$
5	100.0%	0.015	**0.018**	0.032	0.00%	100.0%	0.015	**0.018**	0.032	0.0%
10	100.0%	0.015	**2.088**	743.645	0.00%	100.0%	0.015	**0.027**	0.157	0.0%
15	100.0%	0.014	**1.461**	227.430	0.00%	100.0%	0.015	**0.054**	0.469	0.0%
20	97.2%	0.015	**37.586**	1761.454	2.80%	100.0%	0.015	**0.131**	0.953	0.0%
25	–	–	**–**	–	–	99.8%	0.027	**0.257**	2.174	0.2%
30	–	–	**–**	–	–	99.8%	0.031	**0.707**	32.563	0.2%
35	–	–	**–**	–	–	99.6%	0.037	**9.257**	1312.607	0.4%
40	–	–	**–**	–	–	97.0%	0.063	**24.939**	1462.369	3.0%

Table 6
Computational results for the Class III test instances of HBAP.

l Number of vessels	Class III: 13×112 | 500 samples | HBAP									
	Sedimentation Algorithm					Sedimentation Algorithm with Estimation & Rearrangement Heuristic				
	$t \le \frac{1}{2}h$	min	avg	max	$\frac{1}{2}h < t$	$t \le \frac{1}{2}h$	min	avg	max	$\frac{1}{2}h < t$
5	100.0%	0.004	**0.019**	0.062	0.0%	100.0%	0.000	**0.018**	0.078	0.0%
10	100.0%	0.014	**0.030**	1.563	0.0%	100.0%	0.014	**0.025**	0.125	0.0%
15	100.0%	0.015	**0.405**	131.856	0.0%	100.0%	0.015	**0.052**	0.625	0.0%
20	98.6%	0.015	**14.655**	1756.791	1.4%	100.0%	0.020	**0.126**	1.188	0.0%
25	–	–	**–**	–	–	100.0%	0.031	**0.268**	1.955	0.0%
30	–	–	**–**	–	–	100.0%	0.033	**0.521**	2.782	0.0%
35	–	–	**–**	–	–	100.0%	0.047	**0.947**	4.125	0.0%
40	–	–	**–**	–	–	100.0%	0.109	**2.243**	66.046	0.0%
45	–	–	**–**	–	–	99.8%	0.114	**14.778**	1434.768	0.2%
50	–	–	**–**	–	–	98.2%	0.187	**24.883**	1338.496	1.8%

vessels greater than or equal to 25 are not presented in Table 5 because the time needed for solving 500 examples was too long. Note that the average runtime of SEDA+ERH for 40 vessels is still lower than the average runtime of SEDA for 20 vessels. The percent of problems solved in a half-an-hour period is almost the same if we compare SEDA for 20 vessels (97.2%) and SEDA+ERH for 40 vessels (97.0%). SEDA+ERH was not able to solve all 500 problems in a half-an-hour period for the cases of 25 and more vessels, but the number of unsolved problems is small. Out of 500 problems for 25 and 30 vessels only one problem (0.2%) was unsolved, for 35 vessels 2 problems (0.4%) were unsolved and for 40 vessels 15 problems (3.0%) were unsolved. Table 5 also shows that, compared to SEDA, SEDA+ERH can solve problems with twice as many vessels for the approximately same amount of time.

Table 6 shows the computational results for the Class III test instances for HBAP. Again, in the trivial case of 5 and 10 vessels, both algorithms perform equally well. In the case of 15 vessels, SEDA+ERH is 7.79 times faster than SEDA and in the case of 20 vessels SEDA+ERH is at least 116.31 times faster than SEDA. The results of the SEDA execution for the cases with the number of vessels greater than or equal to 25 are not presented in Table 6 because the time needed for solving 500 examples was too long. However, note that the average runtime of SEDA+ERH for 45 vessels almost equal to the average runtime of SEDA for 20 vessels. The percent of problems solved in a half-an-hour period is very close if we compare SEDA for 20 vessels (98.6%) and SEDA+ERH for 50 vessels (98.2%). Similar to previous example, out of 500 problems only 1 problem (0.2%) was unsolved in a half-a-hour time period by SEDA+ERH for 45 vessels and 9 problems (1.8%) were unsolved for 50 vessels.

From the above examples, it is clear that there is a significant difference between the minimum and maximum runtimes of both algorithms in all examples. This tendency increases as the number of vessels increases. The existence of the *"best-case"* and *"worst-case"* input data, which is typical for combinatorial algorithms, is also evident for SEDA and SEDA+ERH. The *"best-case"* input data are when the vessels do not occupy the same berths at the same time. In all the classes of test instances, it is possible to generate the *"best-case"* input data and make the minimum runtimes short. On the other hand, the *"worst-case"* input data for both algorithms occur when vessels tend to occupy the same berth at the same time. As the number of vessels increases, the probability of the *"worst-case"* input data also increases. As the results indicate, SEDA+ERH is much better when treating the *"worst-case"* input data than SEDA. From the above examples, we can conclude that SEDA is powerful enough to solve the problems that include up to 20 vessels. Depending on the number of berths and the length of the time horizon, SEDA+ERH can solve problems that include 40 to 60 vessels in the acceptable solution time (less than

Please cite this article as: S. Kordić et al., Combinatorial approach to exactly solving discrete and hybrid berth allocation problem, Applied Mathematical Modelling (2016), http://dx.doi.org/10.1016/j.apm.2016.05.004

Table 7
Distribution of below-average solution times.

Distribution of SEDA+ERH runtime for the most complex cases of test instances					
Class of test instance	*L Number of vessels*	*Average runtime*	$t \leq avg$	$avg < t \leq ½h$	$½h < t$
Class I | DBAP	35	17.311	**88.0%**	8.6%	3.4%
Class II | DBAP	65	41.819	**89.9%**	9.6%	0.6%
Class II | HBAP	40	24.939	**90.0%**	7.0%	3.0%
Class III | HBAP	50	24.883	**91.20%**	7.0%	1.8%

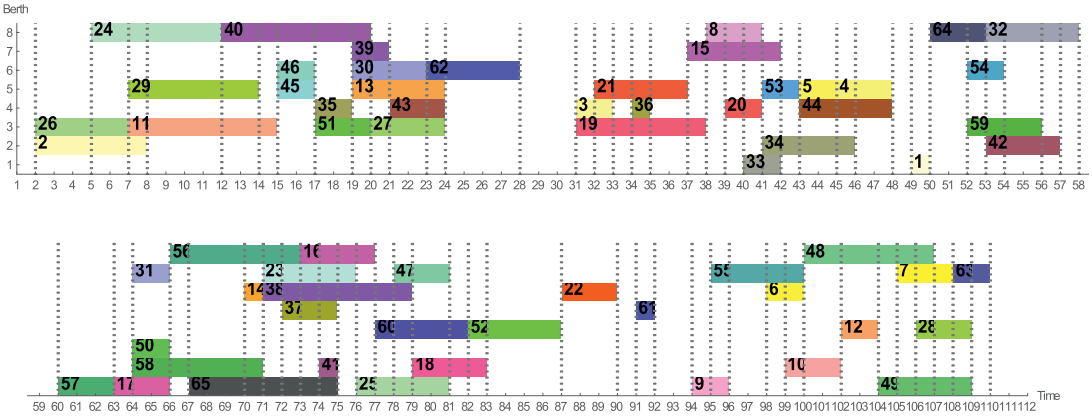

Fig. 6. Optimal solution of the problem No. 76 with 65 vessels, 8 berths (vertical axis) and 112 time units (horizontal axis). Time horizon is split in two parts. Vessels are represented as the rectangles in the bert×time grid. Each vessel has its identifier in the top left corner and its unique color.

1800 s). The occurrences of the *"worst-case"* data for this range are rare because the average solution time is less than or equal to 41 s. Table 7 contains the SEDA+ERH solution time distributions for the most complex test instances.

For the most complex cases, 88.0% to 91.20% of the tests were solved in a time less than the average. Approximately 7.0% to 9.6% of the tests required times between the average solution time of the class and 1800 s but were solved within the 1800-s time limit. Finally, 0.6% to 3.4% of the tests required more than 1800 s to solve the problem. From the percentage of the unsolved problems, within the time limit of 1800 s, we can easily conclude which class of test instances is the most challenging for SEDA+ERH.

5.3. The analysis of one "worst-case" problem

Among the problems that were not solved within the 1800 s time limit we will discuss in more detail the test problem No. 76 of DBAP with 65 vessels, 8 berths and 112 time units (Class II). The optimal solution of this problem is 122 (122 000 $).

During the attempt to solve the problem, ERH started with the solution having the cost of 206 and provided initial suboptimal solution for SEDA with the cost equal to 122, which is also the optimal solution the problem. ERH took 2.593 seconds to reach upper bound 122. After that SEDA was not able to search whole solution space with the upper bound 122 within time limit of 1800 s. Without time limit this example was solved in 2698.728 s. The more sophisticated version of SEDA and ERH proved in 247.586 s that 122 is the optimal solution of the problem No. 76 (Fig. 6).

What makes the problem No. 76 difficult to solve is 13 pairs of mutually conflicting vessels. In the optimal solution one vessel, in conflicting pair, takes position with no penalty and the other with a penalty. This has as a consequence the generation of the "bad" ω ordering in ERH, which is not good enough for SEDA to solve problem within time limit of 1800 s. This example shows that despite the good upper bound, there are still cases which are hard to solve by SEDA.

5.4. Comparison between sedimentation algorithms and CPLEX

In this subsection, we selected first 10 DBAP test instances with 25 vessels, 5 berths and 56 time units (Class I) to compare our approach with the CPLEX commercial solver. For the comparison of SEDA and SEDA+ERH with CPLEX, we used the Version 11.2 of CPLEX, running on a computer with a 2.66-GHz *Intel Core 2 Duo E6750 CPU*, 8GB of RAM, and the *Linux Slackware 12* (kernel version 2.6.21.5) operating system. According to the PassMark (www.passmark.com) CPU Benchmark single thread rating of the *Intel Core 2 Duo E6750* processor is 1002, while for *Intel Core i7-4500 U* processor single thread rating is 1578. From this fact we can conclude that *Intel Core 2 Duo E6750* processor is approximately 0.63498 times slower compared to *Intel Core i7-4500 U* processor. This fact we will include into our comparison of CPLEX, SEDA and SEDA+ERH,

Please cite this article as: S. Kordić et al., Combinatorial approach to exactly solving discrete and hybrid berth allocation problem, Applied Mathematical Modelling (2016), http://dx.doi.org/10.1016/j.apm.2016.05.004

Table 8
Computational results for first 10 Class I test instances of DBAP.

No.	Class I: 5×56 | First 10 problems | DBAP						
	Optimum	SEDA	SEDA+ERH	CPLEX	CPLEX equ	×SEDA	×SEDA+ERH
1	68	2.281	0.218	15316.500	9725.686	**4263.782**	**44613.240**
2	24	0.125	0.035	6577.400	4176.524	**33412.192**	**119329.256**
3	54	0.083	0.203	6187.890	3929.193	**47339.669**	**19355.628**
4	25	0.546	0.047	4758.500	3021.557	**5533.987**	**64288.448**
5	52	0.593	0.219	11063.300	7024.985	**11846.518**	**32077.558**
6	21	0.047	0.047	2440.720	1549.811	**32974.698**	**32974.698**
7	60	8.076	0.218	16385.400	10404.417	**1288.313**	**47726.686**
8	16	0.096	0.096	4678.640	2970.847	**30946.328**	**30946.328**
9	27	0.203	0.039	134.279	85.265	**420.023**	**2186.272**
10	29	0.047	0.187	2764.770	1755.576	**37352.689**	**9388.109**
	Σ	12.097	1.309	70307.399	44643.862	**3690.490**	**34105.318**

by multiplying runtimes of CPLEX with calculated factor to get approximately equivalent times as if CPLEX was running on the *Intel Core i7-4500 U*.

Table 8 shows the number of problem, the value of the optimal solution, runtimes for SEDA, SEDA+ERH and CPLEX. Adjusted CPLEX runtimes by coefficient 0.63498 are given in the sixth column of Table 8. All runtimes are expressed in seconds. In the last two columns, we present the ratios between the runtimes required by adjusted CPLEX and SEDA (SEDA+ERH). Both SEDA and SEDA+ERH outperform CPLEX significantly. SEDA is between 420.023 and 47339.669 times faster, while SEDA+ERH is 2186.272 to 119329.256 times faster than CPLEX. Overall times for solving all 10 problems are given in the row below Table 8. SEDA solved all 10 problems in 12.097 s, SEDA+ERH solved them in 1.309 s and adjusted CPLEX in 44643.862 s, which makes SEDA 3690.490 times faster and SEDA+ERH 34105.318 times faster than CPLEX.

The most recent version of the CPLEX is 12.6.1. According to the IBM ILOG CPLEX Optimizer page (http://www-01.ibm.com/software/commerce/optimization/cplex-performance/#improvements), and performance benchmarks for MIP CPLEX 12.6.1. it is approximately up to 40 times faster than CPLEX version 11.2. used for our comparison with SEDA and SED+ERH. This approximation is calculated by multiplying version-to-version improvements factor from version 11 to version 12.6.1. If we adopt 40 as the maximal speed up factor from CPLEX version 11.2. to version 12.6.1, then SEDA is at least 92.250 times faster and SEDA+ERH is at least 852.625 times faster. This proves the superiority of both dedicated combinatorial algorithms compared to a general MIP solver such as CPLEX. Computational experiments including a larger number of vessels were not conducted because even for some instances with 25 vessels runtimes required by CPLEX were already very long (almost 5 h). The MILP model, used for this test, suggested by Davidović et al. [37] is too complex and obviously does not provide adequate results for the examples with more than 25 vessels, both with the framework of CPLEX commercial software and the application of MIP-based meta-heuristics.

5.5. Note on the sedimentation algorithms and other exact BAP solvers

The most prominent recent approach for exact solving of BAP can be found in Vacca et al. [5]. The authors consider *Tactical Berth Allocation Problem* (TBAP), variant of BAP defined by Giallombardo et al. [6] which integrate planning of berth allocation and crane assignment (BACAP). Instead of exploring all feasible crane assignments for a vessel, only some subsets, called profiles, are considered in TBAP. Objective function maximize the total value of selected profile and the total housekeeping cost generated by the berth allocation plan. On the other hand, SEDA and SEDA+ERH are purely BAP solving algorithms and their objective function minimize the cost of vessels waiting time, speed up time, tardiness and berth position. These two facts:

1. difference of the problem (model) formulation: BAP vs BACAP and
2. difference of the objective functions;

makes direct comparison of these two approaches impossible. Simply, they are designed to solve different type of the BAP. Therefore, their comparison is omitted. Our impression is that TBAP can be modeled for SEDA and SEDA+ERH, and then direct comparison with the approach of Vacca et al. [5] will be possible.

However, it is worth noticing that SEDA+ERH is capable of solving very efficiently test instances having the same number of berths, time horizon and number of vessels as the test instances used in Giallombardo et al. [6]. As we previously explained runtimes for solving are not directly comparable. In the Table 9 average runtimes of SEDA+ERH are given for Giallombardo et al. [6] tests instances sizes. Times are given only for DBAP, since in Giallombardo et al. [6] only DBAP is considered. These times illustrate capability of SEDA+ERH to efficiently solve problems of the Giallombardo et al. [6] tests instances sizes.

Other recent exact solver for BAP, like Umang et al. [7] and Robenek et al. [8] address BAP in bulk ports. That is similar, but yet different problem then BAP in container port which is addressed in our work. Therefore, their comparison is omitted.

Please cite this article as: S. Kordić et al., Combinatorial approach to exactly solving discrete and hybrid berth allocation problem, Applied Mathematical Modelling (2016), http://dx.doi.org/10.1016/j.apm.2016.05.004

Table 9
Average runtimes of SEDA+ERH for Giallombardo tests instances sizes.

Average runtimes of SEDA+ERH for some size instances as in Giallombardo				
Class of test instance	*Number of berths*	*Time horizon in weeks (time units)*	*Number of vessels*	*Average runtime of SEDA+ERH in seconds*
Class I | DBAP	5	1 (56)	30	**1.328**
Class II | DBAP	8	2 (112)	40	**0.606**
Class II | DBAP	8	2 (112)	50	**1.518**

6. Conclusion

We considered the Discrete (minimum-cost) Berth Allocation Problem (DBAP) and Hybrid (minimum-cost) Berth Allocation Problem (HBAP) with the static arrival of vessels and fixed vessel handling times. The computational experiments performed fully justify the design and further development of *Sedimentation Algorithm*, the exact combinatorial approach for solving DBAP and HBAP. When combined with a simple heuristic like *Estimation & Rearrange Heuristic*, the algorithm can be used for instances with large number of vessels as a standalone method or as a part of some more complex heuristic or meta-heuristic approach to solving DBAP or HBAP. The most difficult problems of our test instances can be solved within the time limit of 1800 s. Actually, majority of them are solved in less than 6.65 s on average, including the cases with 30 up to 60 vessels. These results indicate that this method can be used for solving real-life medium and big size instances of BAP, depending on container port layout.

The differences between the minimum and maximum solving times for a large number of vessels in the test instances indicate that *Sedimentation Algorithm* is worth further development. We find it especially worthwhile to investigate the possibility of solving sub-problems of DBAP or HBAP that contain conflicting vessels and then combining these partial solutions to obtain the optimal solution of the entire problem. Moreover, the generalization of the algorithm to the continuous BAP and the inclusion of crane assignment to *Sedimentation Algorithm* are natural avenues for further development of the algorithms presented here.

References

[1] A. Lim, The berthing planning problem, Oper. Res. Lett. 22 (2) (1998) 105–110.
[2] C. Bierwirth, F. Meisel, A survey of berth allocation and quay crane scheduling problems in container terminals, Eur. J. Oper. Res. 202 (3) (2010) 615–627.
[3] F. Meisel, Seaside Operations Planning in Container Terminals, Physica Verlag, Berlin, 2009.
[4] C. Bierwirth, F. Meisel, A follow-up survey of berth allocation and quay crane scheduling problems in container terminals, Eur. J. Oper. Res. 244 (3) (2015) 675–689.
[5] I. Vacca, M. Salani, M. Bierlaire, An exact algorithm for integrated planning of berth allocation and quay crane assignment, Rep. TRANSP-OR 110323 (2011).
[6] G. Giallombardo, L. Moccia, M. Salani, I. Vacca, Modeling and solving the tactical berth allocation problem, Transp. Res. B 44 (3) (2010) 400–415 (2010).
[7] N. Umang, M. Bierlaire, I. Vacca, Exact and heuristic methods to solve berth allocation problems in bulk ports, Transp. Res. E 54 (2013) 14–31.
[8] T. Robenek, N. Umang, M. Bierlaire, S. Ropke, A branch-and-price algorithm to solve the integrated berth allocation and yard assignment problem in bulk ports, Eur. J. Oper. Res. 235 (2014) 399–411.
[9] A. Imai, E. Nishimura, S. Papadimitriou, The dynamic berth allocation problem for container port, Transp. Res. B 35 (4) (2001) 401–417.
[10] F.M. Monaco, M. Samara, The berth allocation problem: a strong formulation solved by a lagrangian approach, Transp. Sci. 41 (2) (2007) 256–280.
[11] J.F. Cordeau, G. Laporte, P. Legato, L. Moccia, Models and tabu search heuristics for the berth-allocation problem, Transp. Sci. 39 (4) (2005) 526–538.
[12] G.R. Mauri, A.C.M. Oliveira, A.N. Lorena, A hybrid column generation approach for the berth allocation problem, in: EvoCOP, 4972, 2008, p. 110–122.
[13] C.G. Cristensen, C.T. Holst, Berth allocation in container terminals, Master's thesis, Department of Informatics and Mathematical Modelling, Technical University of Denmark, Lyngby, 2008.
[14] L. Zhen, L.H. Lee, E.P. Chew, A decision model for berth allocation under uncertainty, Eur. J. Oper. Res. 212 (1) (2011) 54–68.
[15] R.M. de Oliveira, G.R. Mauri, L.A.N. Lorena, Clustering search for the berth allocation problem, Expert Syst. Appl. 39 (2012) 5499–5505.
[16] Y. Lee, C.Y. Chen, An optimization heuristic for the berth scheduling problem, Eur. J. Oper. Res. 196 (2) (2009) 500–508.
[17] P. Hansen, C. Oğuz, N. Mladenović, Variable neighbourhood search for minimum cost berth allocation, Eur. J. Oper. Res. 191 (3) (2008) 636–649.
[18] A. Imai, E. Nishimura, S. Papadimitriou, Berthing ships at a multi-user container terminal with a limited quay capacity, Transp. Res. E 44 (1) (2008) 136–151.
[19] M. Han, P. Li, J. Sun, The algorithm for berth scheduling problem by the hybrid optimization strategy GASEDA, in: Proceedings of the 9th International Conference of Control, Automation, Robotics and Vision, ICARCV '06 Washington, IEEE Computer Society, Washington, 2006, p. 1–4.
[20] P. Zhou, H. Kang, L. Lin, A dynamic berth allocation model based on stochastic consideration, in: Proceedings of the 6th World Congress on Intelligent Control and Automation, Vol 2, Washington DC: IEEE Computer Society, Washington, 2006, p. 7297–7301.
[21] E. Nishimura, A. Imai, S. Papadimitriou, Berth allocation planning in the public berth system by genetic algorithms, Eur. J. Oper. Res. 131 (2) (2001) 282–292.
[22] S.W. Lin, K.C. Ying, S.Y. Wan, Minimizing the total service time of discrete dynamic berth allocation problem by an iterated greedy heuristic, Sci. World J. 2014. http://dx.doi.org/10.1155/2014/218925
[23] E. Lalla-Ruiz, S. Voß, POPMUSIC as a metaheuristic for the berth allocation problem, Ann. Math. Artif. Intell. 76 (1) (2014) 173–189 http://dx.doi.org/10.1007/s10472-014-9444-4 .
[24] C.Y. Chen, T.W. Hsieh, A time-space network model for the berth allocation problem, in: 19th IFIP TC7 Conference on System Modelling and Optimization, Cambridge, 1998.
[25] R. Moorthy, C.P. Teo, Berth management in container terminal: the template design problem, OR Spectr. 28 (4) (2006) 495–427.
[26] J. Dai, W. Lin, R. Moorthy, C.P. Teo, Berth allocation planning optimization in container terminals, in: Supply chain analysis: a handbook on the interaction of information, system and optimization, Springer, New York, 2008, p. 69–105.

Please cite this article as: S. Kordić et al., Combinatorial approach to exactly solving discrete and hybrid berth allocation problem, Applied Mathematical Modelling (2016), http://dx.doi.org/10.1016/j.apm.2016.05.004

[27] N. Kovač, Bee Colony Optimization Algorithm for the Minimum Cost Berth Allocation Problem, in: Proceedings of the XI Balkan Conference on Operational Research, Belgrade-Zlatibor, 2013, p. 245–254.
[28] A. Imai, E. Nishimura, M. Hattori, S. Papadimitriou, S, Berth allocation at indented berths for mega-containerships, Eur. J. Oper. Res. 179 (2) (2007) 579–593 (2007).
[29] A. Imai, E. Nishimura, S. Papadimitriou, Marine container terminal configurations for efficient handling of mega-containerships, Transp. Res. E 49 (2013) 141–158.
[30] C.Y. Cheong, C.J. Lin, K.C. Tan, D.K. Liu, A multi-objective evolutionary algorithm for berth allocation in the container port, in: IEEE Congress on Evolutionary Computation, IEEE Computer Society, Washington DC, 2007, p. 927–934.
[31] L. Hoffarth, S. Voß, Berth allocation in container terminal – development of decision support system (in German), in: Operations Research Proceedings 1993, Berlin, Springer, 1994, p. 89–95.
[32] H. Rashidi, E.P.K. Tsang, Novel constrains satisfaction models for optimization problems in container terminals, Appl. Math. Model. 37 (2013) 3601–3634.
[33] S. Kordić, B. Dragović, T. Davidović, N. Kovač, N., A Combinatorial Algorithm for Berth Allocation Problem in Container Port, in: Proceedings of International Association of Maritime Economists Conference, Taipei, IAME, 2012 2012, pp. 1–15 (MD-065).
[34] Y.M. Fu, A. Diabat, A Lagrangian relaxation approach for solving the integrated quay crane assignment and scheduling problem, Appl. Math. Model. 39 (2015) 1194–1201.
[35] F. Meisel, C. Bierwirth, A framework for integrated berth allocation and crane operations planning in seaport container terminals, Transp. Sci. 47 (2) (2013) 131–147 http://dx.doi.org/10.1287/trsc.1120.0419 .
[36] B. Dragović, N.K. Park, Z. Radmilović, Ship-berth link performance evaluation: simulation and analytical approaches, Marit. Policy Manag 33 (3) (2006) 281–299.
[37] T. Davidović, J. Lazić, N. Mladenović, S. Kordić, N. Kovač, B. Dragović, MIP-Heuristics for Minimum Cost Berth Allocation Problem, in: Proceedings of International Conference on Traffic and Transport Engineering, ICTTE 2012, Belgrade, 2012, p. 21–28.

Please cite this article as: S. Kordić et al., Combinatorial approach to exactly solving discrete and hybrid berth allocation problem, Applied Mathematical Modelling (2016), http://dx.doi.org/10.1016/j.apm.2016.05.004