Received 24 August 2019
Revised 27 September 2019
Accepted 20 October 2019

Berth scheduling at marine container terminals

A universal island-based metaheuristic approach

Masoud Kavoosi
Florida A&M University-Florida State University College of Engineering, Tallahassee, Florida, USA

Maxim A. Dulebenets
Department of Civil and Environmental Engineering, Florida A&M University-Florida State University College of Engineering, Tallahassee, Florida, USA

Olumide Abioye, Junayed Pasha, Oluwatosin Theophilus, Hui Wang and Raphael Kampmann
Florida A&M University-Florida State University College of Engineering, Tallahassee, Florida, USA, and

Marko Mikijeljević
University of Montenegro, Podgorica, Montenegro

Abstract

Purpose – Marine transportation has been faced with an increasing demand for containerized cargo during the past decade. Marine container terminals (MCTs), as the facilities for connecting seaborne and inland transportation, are expected to handle the increasing amount of containers, delivered by vessels. Berth scheduling plays an important role for the total throughput of MCTs as well as the overall effectiveness of the MCT operations. This study aims to propose a novel island-based metaheuristic algorithm to solve the berth scheduling problem and minimize the total cost of serving the arriving vessels at the MCT.

Design/methodology/approach – A universal island-based metaheuristic algorithm (UIMA) was proposed in this study, aiming to solve the spatially constrained berth scheduling problem. The UIMA population was divided into four sub-populations (i.e. islands). Unlike the canonical island-based algorithms that execute the same metaheuristic on each island, four different population-based metaheuristics are adopted within the developed algorithm to search the islands, including the following: evolutionary algorithm (EA), particle swarm optimization (PSO), estimation of distribution algorithm (EDA) and differential evolution (DE). The adopted population-based metaheuristic algorithms rely on different operators, which facilitate the search process for superior solutions on the UIMA islands.

Findings – The conducted numerical experiments demonstrated that the developed UIMA algorithm returned near-optimal solutions for the small-size problem instances. As for the large-size problem instances, UIMA was found to be superior to the EA, PSO, EDA and DE algorithms, which were executed in isolation, in terms of the obtained objective function values at termination. Furthermore, the developed UIMA algorithm outperformed various single-solution-based metaheuristic algorithms (including variable neighborhood

Maritime Business Review
Vol. 5 No. 1, 2020
pp. 30-66
Emerald Publishing Limited
2397-3757
DOI 10.1108/MABR-08-2019-0032

This work has been partially supported by the National Science Foundation grant CMMI-1901109. The opinions, findings and conclusions, expressed in this publication, are those of the authors and do not necessarily reflect the views of the National Science Foundation.

search, tabu search and simulated annealing) in terms of the solution quality. The maximum UIMA computational time did not exceed 306 s.

Research limitations/implications – Some of the previous berth scheduling studies modeled uncertain vessel arrival times and/or handling times, while this study assumed the vessel arrival and handling times to be deterministic.

Practical implications – The developed UIMA algorithm can be used by the MCT operators as an efficient decision support tool and assist with a cost-effective design of berth schedules within an acceptable computational time.

Originality/value – A novel island-based metaheuristic algorithm is designed to solve the spatially constrained berth scheduling problem. The proposed island-based algorithm adopts several types of metaheuristic algorithms to cover different areas of the search space. The considered metaheuristic algorithms rely on different operators. Such feature is expected to facilitate the search process for superior solutions.

Keywords Optimization, Supply chains, Marine transportation, Berth scheduling, Island-based algorithms, Marine container terminals, Parallel algorithms

Paper type Research paper

1. Introduction

During the past three decades, the offshoring of manufacturing activities (i.e. relocation of the production facilities or supply basins to a foreign country, aiming to decrease the production expenses) has become a popular approach and was adopted by a lot of firms (Da Silveira, 2014). This tendency has created an increasing demand for containerized marine transportation and global trade. Hence, the marine container terminal (MCT) operators are expected to enhance the MCT operations, given the available handling resources, to increase the MCT throughput and effectively serve the growing demand. The basic operations at each MCT can be classified into the seaside operations, the marshaling yard operations, and the landside operations (Dulebenets *et al.*, 2018). This study investigates the seaside operations with a primary focus on the berth scheduling problem (BSP). The BSP is a decision problem, in which the arriving vessels have to be assigned to the available berthing positions, and the order of vessel service has to be determined for each berthing position (Bierwirth and Meisel, 2015).

The berth scheduling process can be classified into three different levels, including (Zhen *et al.*, 2017):

(1) monthly berth planning;

(2) weekly berth planning; and

(3) daily berth planning.

The monthly berth planning is based on the initial information regarding the vessels' monthly arrival plans. Also, the information regarding the physical specifications of the arriving vessels is exchanged between the liner shipping companies and the MCT operator. In the weekly berth planning, the liner shipping companies update the estimated arrival times and departure times of vessels. The MCT operator assigns a berthing position to each vessel with approximate start and finish service times. Moreover, the marshaling yard planning can be performed based on the assigned berthing positions for the arriving vessels. As for the daily berth planning, the liner shipping companies are required to provide more precise vessel arrival times and expected departure times to the MCT operator to avoid potential delays in the vessel service. Based on the obtained information, the MCT operator determines the start service times and finish service times of the arriving vessels as well as

allocates the required handling equipment and resources. Considering the aforementioned berth planning classifications, this study focuses on daily berth scheduling.

The exact optimization and metaheuristic algorithms can be used to solve the BSP. However, the BSP literature shows that the exact optimization algorithms cannot solve the realistic-size problem instances in a reasonable computational time (Ting *et al.*, 2014; Dulebenets *et al.*, 2018). Therefore, the heuristic and metaheuristic algorithms have been commonly adopted to tackle the realistic-size problem instances of BSPs (Bierwirth and Meisel, 2015). In 1980s, island-based metaheuristic algorithms started receiving more attention of the community (Cohoon *et al.*, 1987). The population of candidate solutions to the problem of interest, initialized in the island-based metaheuristic, is divided into several sub-populations to establish a set of islands. Each island is assigned a separate metaheuristic algorithm (Alba and Tomassini, 2002). In the canonical island-based metaheuristic algorithms, all the islands are searched using the same type of metaheuristic algorithm (Alba and Tomassini, 2002). Island-based metaheuristics allow maintaining the population diversity as different islands explore various domains of the search space (Eiben and Smith, 2003), which further allows avoiding a premature convergence. However, none of the previous BSP studies applied island-based metaheuristic algorithms, despite their proven effectiveness for various combinatorial optimization problems.

In this study, an innovative universal island-based metaheuristic algorithm (UIMA) is developed, in which the population is divided into four sub-populations (which are also referred to as "islands"). Unlike the canonical island-based algorithms that execute the same metaheuristic on each island, four different population-based metaheuristics are adopted within the developed algorithm to search the islands, including the following:

(1) evolutionary algorithm (EA);
(2) particle swarm optimization (PSO);
(3) estimation of distribution algorithm (EDA); and
(4) differential evolution (DE).

Each algorithm deploys a unique set of operators, which is expected to facilitate the search process for superior solutions and improve the quality of berth schedules. The developed solution algorithm is compared against the alternative algorithms in terms of different performance indicators. The remainder of the manuscript includes the following sections. A concise review of the studies related to the BSP and island-based algorithms is presented in the second section. A description of the problem of interest and the developed mathematical model is provided in the third and fourth sections, respectively. The fifth section describes the proposed UIMA. The conducted numerical experiments are summarized in the sixth section. The last section presents the conclusions and future extensions of this study.

2. Review of related studies

Bierwirth and Meisel (2015) conducted a comprehensive literature survey, which covers the studies related to berth allocation, quay crane assignment, and quay crane scheduling problems. Carlo *et al.* (2015) also published a survey study, covering the studies related to the seaside MCT operations. The literature review under this study will focus on two study groups:

(1) some of the recent BSP studies published after 2015; and
(2) the representative studies that relied on island-based algorithms.

Moreover, this section presents the literature summary and contributions of this study to the state-of-the-art.

2.1 Berth scheduling problem literature

Hsu (2016) developed a model for a simultaneous optimization of berth allocation and quay crane assignment. A Hybrid PSO was combined with an event-based heuristic to solve the formulated problem. The conducted numerical experiments showed that the developed solution approach outperformed a canonical EA and a Hybrid EA in terms of the objective function values. Mauri *et al.* (2016) proposed a BSP formulation and considered both continuous and discrete berthing layouts. The problem of interest was solved by an Adaptive Large Neighborhood Search. The capability of the proposed algorithm to return high-quality solutions was demonstrated throughout the conducted numerical experiments. Tsai *et al.* (2016) proposed a mathematical model for a BSP, aiming to reduce the total waiting time of vessels. The considered problem was solved using a Wharf-based genetic algorithm (GA). The results showed a superior performance of the proposed algorithm against three other algorithms in terms of the objective function values and computational time.

Dulebenets *et al.* (2017) developed a mixed-integer mathematical model to minimize the total service cost of the arriving vessels, which also included the total carbon dioxide emission cost. The developed mathematical model was solved using a Hybrid EA. The computational experiments proved that consideration of the emission cost could influence the berth scheduling plans. Dulebenets (2017) developed a berth scheduling model to minimize the total service cost of vessels. A Memetic Algorithm with a deterministic parameter control was presented as a solution approach for the developed optimization model. The results showcased that the proposed algorithm was able to provide promising solutions in a reasonable computational time. Venturini *et al.* (2017) considered a collaborative relationship between the MCTs and the liner shipping companies. The speed of vessels was optimized at all the sailing legs. CPLEX was adopted to solve the developed mathematical model. The results showed that the computational time of the proposed solution approach was affected with the problem size.

Li *et al.* (2017) considered a multi-objective coordinated berth and quay crane scheduling problem, which aimed to minimize the port time of the arriving vessels and the additional trucking distance. A Chaos Cloud Particle Swarm Algorithm was developed for solving the proposed model. The results verified that the proposed solution approach could greatly decrease both trucking distance and port time of vessels. Zhen *et al.* (2017) conducted a study for an operational-level berth allocation and quay crane assignment problem, constrained by the tidal patterns and the channel flow control. The initially developed mathematical model was reformulated based on a set partitioning approach. A Column Generation approach was adopted to solve the formulated problem. The results showed the capability of the proposed approach to solve the real-world problem instances with up to 80 vessels, 40 berthing positions, and 120 quay cranes within an hour. Jiao *et al.* (2018) proposed an integrated berth allocation and time-variant quay crane scheduling problem, considering the tidal effects in the approach channel. The objective minimized the total vessel turnaround time. The computational experiments demonstrated the remarkable effects of tidal patterns on the objective function values.

2.2 Literature on island-based algorithms

Island-based metaheuristic algorithms have been applied for a wide range of optimization problems. For example, Lardeux and Goëffon (2010) developed a Dynamic Island-based EA

Algorithm for combinatorial decision problems. In that study, the last migration impact was taken into account to dynamically update the migration probabilities. The numerical experiments showed efficiency of the proposed strategy for the 0/1 Knapsack problem and the MAX-SAT problem. Gong and Fukunaga (2011) proposed a novel approach for determining the values of algorithmic parameters within an Island-based GA. In that study, instead of using a parameter tuning or a self-adaptive parameter control strategy, parameters were statistically assigned random control parameter values. The numerical experiments showed that the proposed approach could return the solutions, which were competitive to a homogeneously distributed GA with the parameters that were tuned specifically for each one of the benchmark problem instances. Osaba *et al.* (2013) focused on different variations of the routing problem and used a Multi-Crossover and Adaptive Island-based GA as a solution approach. In the developed algorithm, each one of the sub-populations was assigned a specific crossover type, which could be switched based on its efficiency.

Ammi and Chikhi (2015) proposed a model for a capacitated vehicle routing problem (CVRP). The study aimed to minimize the cost of a reliable product distribution over a large distribution network. An island-based metaheuristic algorithm was developed to solve the problem. A GA and an Ant Colony Optimization Algorithm were adopted to cover different areas of the search space throughout the search process. The numerical experiments, conducted for a large-scale CVRP, showed superiority of the solutions, provided by the developed algorithm, as compared to the well-known benchmark values that had been used in the CVRP literature. Kurdi (2015) developed a new Hybrid Island-based GA for a job shop scheduling problem with the overall objective of minimizing the makespan. A naturally inspired self-adaptation phase strategy was embedded within the algorithm to improve performance of the island-based algorithm. The considered numerical experiments showcased efficiency of the proposed self-adaptation strategy in terms of the solution quality. Al-Betar *et al.* (2015) evaluated performance of the Harmony Search Algorithm in an island-based framework. The performance of the developed island-based algorithm was evaluated for a group of benchmark functions, which had been previously used in the literature. It was found that application of Harmony Search improved effectiveness of the algorithm.

Magalhaes *et al.* (2015) studied 10 strategies that could be used to conduct migration of individuals between the islands in the island-based metaheuristic algorithm. The islands were searched by a DE and a GA. A total of 30 different scalable problem instances were developed to perform the numerical experiments. Advantages and disadvantages of the developed strategies were evaluated and discussed based on the quality of results. Brester *et al.* (2017) developed a decision support tool, aiming to assist with decision making in the field of project management. The objectives of the study maximized the income and minimized the risks that resulted from the decisions made by a given project manager. Three different algorithms were adopted in that study, and various combinations of them were considered to search the islands. All the developed combinations were evaluated in terms of the objective function values as well as the computational time. The results confirmed the efficiency of introducing the island-based framework.

2.3 Literature summary and contributions

Based on the conducted literature review, island-based metaheuristic algorithms have never been used in the BSP literature so far (Bierwirth and Meisel, 2015), but have been found to be efficient for various challenging decision problems. Given potential advantages of the island-based framework, this study proposes a novel UIMA for the BSP, which executes

four different population-based metaheuristic algorithms on its islands to search more efficiently for promising solutions. Unlike typical island-based metaheuristic algorithms, which use the same metaheuristic on different islands, the developed UIMA relies on four different metaheuristics (EA, PSO, EDA, and DE) to effectively search the islands by taking unique advantages of these metaheuristics.

3. Problem description

In this section of the manuscript, the BSP, which is investigated in this study, is described in detail. Several types of MCT berthing layouts have been defined in the BSP literature, including discrete, continuous, indented, hybrid, and channel berthing layouts (Bierwirth and Meisel, 2015). In this study, a discrete berthing layout was adopted, based on which the wharf is divided into a set of berthing positions ($B = \{1, \ldots, n\}$), and each one of them can serve one vessel at a time. Figure 1 illustrates the berthing layout of the MCT, where a group of vessels ($V = \{1, \ldots, m\}$) are served at n discrete berthing positions. The vessel arrival patterns in BSPs can be categorized into dynamic and static. The former arrival pattern was adopted in this study. Based on the dynamic vessel arrival case, the vessels have not arrived at the MCT yet, but the liner shipping companies have already informed the MCT operator regarding the expected vessel arrival times. Let $T = \{1, \ldots, p\}$ be the set of discrete periods in the considered planning horizon, and $\theta_v^a, v \in V$(period) be the period when vessel v arrives at the MCT. Once a vessel is getting close to the MCT, a number of tug boats tow the vessel to the assigned berthing position. If the assigned berthing position cannot serve the vessel immediately upon its arrival, it will be towed to the waiting area of the MCT. Let $\boldsymbol{\theta}_v^{wt}, v \in V$ (periods) be the number of periods that vessel v should wait for service (periods), and $\boldsymbol{\theta}_b^{ber}, b \in B$ (period) be the period when berthing position b is available for the first time in the considered planning horizon. The service of a given vessel cannot start before its assigned berthing position becomes available for the first time in the planning horizon.

The MCT operator prepares a berth scheduling plan, in which each one of the vessels is assigned to one of the available berthing positions, in advance. The berthing position, assigned based on a preliminary berth scheduling plan, is referred to as a "preferred berthing position", where a given vessel receives the highest handling rate (i.e. the shortest handling time). In fact, the preferred berthing position is the closest berthing position to the storage area, which was assigned for the containers to be delivered by the given vessel in the marshaling yard (Bierwirth and Meisel, 2015). However, the assigned berthing position may

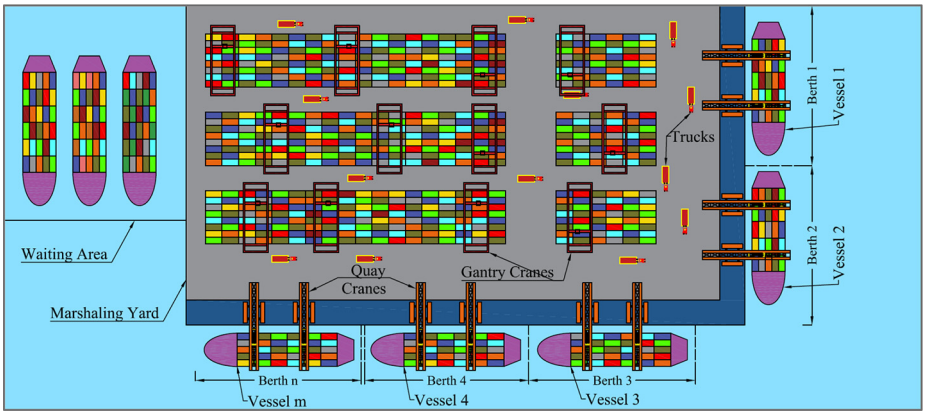

Figure 1.
The MCT berthing layout

not necessarily be the same as the preferred one (e.g. a vessel can be re-assigned to another berthing position to reduce its waiting time). The handling time of each vessel can be calculated using this relationship (Dulebenets *et al.*, 2018): $\theta_{vb}^{ht} = \widetilde{\theta_v^{ht}} \cdot (1 + \xi \cdot abs(B_v^{pr} - B_v^{as}))\ \forall v \in V, b \in B$, where $\theta_{vb}^{ht}, v \in V, b \in B$ (periods) are the periods required for handling vessel v at berthing position b; $\widetilde{\theta_v^{ht}}, v \in V$ (periods) are the periods required for handling vessel v at its preferred berthing position; $B_v^{pr}, v \in V$ is the index for the preferred berthing position of vessel v; $B_v^{as}, v \in V$ is the index for the berthing position assigned to vessel v; and ξ is a coefficient of the handling time increase (per cent) due to diversion of vessel v from its preferred berthing position to the other MCT berthing position (assumed to be 3 per cent in this study).

The BSP spatial requirements are also modeled throughout this study. In detail, the length of each vessel ($L_v^{ves}, v \in V$– feet) and the draft of each vessel ($D_v^{ves}, v \in V$ – feet) should be compatible with the length of the assigned berthing position ($L_b^{ber}, b \in B$ – feet) and the depth of the assigned berthing position ($D_b^{ber}, b \in B$ – feet), considering a horizontal clearance (P_v^{hor}, $v \in V$ – feet) and a vertical clearance ($P_v^{ver}, v \in V$ – feet), respectively. A specific departure time is assigned to each one of the arriving vessels and is determined based on the agreements between the liner shipping companies and the MCT operator. Let $\theta_v^d, v \in V$ (period) be the negotiated departure period for vessel v. If a given vessel departs the MCT later than the pre-determined departure period, the entire voyage of the vessel can be interrupted as the vessel would arrive at the next ports of call with a delay. This study assumes that the MCT operator will incur a monetary penalty for the late departures of vessels. The MCT operator is expected to allocate adequate handling equipment (e.g. straddle carriers, yard trucks, automated lifting vehicles, automated guided vehicles) and sufficient storage area in the marshaling yard for each one of the vessels to be able to have all the vessels served based on their specific arrival times and negotiated departure times. Some of the BSP studies modeled uncertain vessel arrival times and/or handling times (Bierwirth and Meisel, 2015), while this study assumed the vessel arrival and handling times to be deterministic. The considered BSP aims to minimize the total vessel waiting cost, the total vessel handling cost, and the total vessel late departure cost.

4. Mathematical formulation of the optimization model

Under this section of the manuscript, the mathematical model, developed for the Spatially Constrained BSP (SCBSP), is described in a comprehensive manner. A mixed-integer linear mathematical formulation was adopted for the considered BSP. In Table I, the nomenclature, which will be used from now on, is presented.

Equation (1) represents the objective function, which was adopted for the SCBSP mathematical model. This equation minimizes the summation of the total waiting cost, the total handling cost, and the total late departure cost of the arriving vessels at the MCT.

$$min\ \boldsymbol{Z} = \left[\sum_{v \in V}\left(\varphi_v^{wt}\theta_v^{wt}\right) + \sum_{v \in V}\sum_{b \in B}\sum_{t \in T}\left(\varphi_v^{ht}\theta_{vb}^{ht}\boldsymbol{x}_{vbt}\right) + \sum_{v \in V}\left(\varphi_v^{lt}\theta_v^{lt}\right)\right] \tag{1}$$

A group of constraint sets are defined in the optimization model, which correspond to real-world operational features of the problem studied herein. Constraint set (2) guarantees that each vessel will be assigned to a berthing position just once during the considered planning horizon.

Model component Type	Nomenclature	Description
Sets	$V = \{1, \ldots, m\}$	the set of arriving vessels at the MCT (vessels)
	$B = (1, \ldots, n\}$	the set of available berthing positions at the MCT (berthing positions)
	$T = \{1, \ldots, p)$	the set of discrete periods (periods)
Decision Variables	$\boldsymbol{x}_{vbt}, v \in V,$ $b \in B, t \in T$	=1 if vessel v is assigned for service at berthing position b at period t (=0 otherwise)
Auxiliary Variables	$\boldsymbol{\theta}_{vbt}^{it}, \ v \in V,$ $b \in B, \ t \in T$	the number of idling periods for berthing position b between service of vessel v and its immediate predecessor served during period $(t-1)$ (periods)
	$\boldsymbol{q}_{vbt}, v \in V,$ $b \in B, t \in T$	=1 if vessel v leaves berthing position b at period t (=0 otherwise)
	$\boldsymbol{\theta}_v^{st}, v \in V$	the start service period for vessel v (period)
	$\boldsymbol{\theta}_v^{ft}, v \in V$	the finish service period for vessel v (period)
	$\boldsymbol{\theta}_v^{wt}, v \in V$	the number of periods that vessel v should wait for service (periods)
	$\boldsymbol{\theta}_v^{lt}, v \in V$	the number of periods that vessel v leaves later than the negotiated departure period (periods)
Parameters	m	the total number of arriving vessels at the MCT to be served (vessels)
	n	the total number of available berthing positions at the MCT (berthing positions)
	p	the total number of periods in the planning horizon (periods)
	$\theta_v^a, v \in V$	the period when vessel v arrives at the MCT (period)
	$\theta_b^{ber}, b \in B$	the period when berthing position b is available for the first time in the considered planning horizon (period)
	$\widetilde{\theta_v^{ht}}, \ v \in V$	the periods required for handling vessel v at its preferred berthing position (periods)
	$\theta_{vb}^{ht}, v \in V, b \in B$	the periods required for handling vessel v at berthing position b (periods)
	$\theta_v^d, v \in V$	the negotiated departure period for vessel v (period)
	$D_v^{ves}, \ v \in V$	the draft of vessel v (feet)
	$D_b^{ber}, \ b \in B$	the depth of berthing position b (feet)
	$L_v^{ves}, v \in V$	the length of vessel v (feet)
	$L_b^{ber}, b \in B$	the length of berthing position b (feet)
	$P_v^{ver}, v \in V$	the minimum vertical clearance for vessel v (feet)

(*continued*)

Table I. Nomenclature adopted for the mathematical model

Model component Type	Nomenclature	Description
	$P_v^{hor}, v \in V$	the minimum horizontal clearance for vessel v (feet)
	$\varphi_v^{wt}, v \in V$	the unit waiting cost component for vessel v (US$/period)
	$\varphi_v^{ht}, v \in V$	the unit handling cost component for vessel v (US$/period)
	$\varphi_v^{lt}, v \in V$	the unit late departure cost component for vessel v (US$/period)
	Y	a large positive number

Table I.

$$\sum_{b \in B} \sum_{t \in T} \boldsymbol{x}_{vbt} = 1 \ \forall v \in V \tag{2}$$

Constraint set (3) prevents assigning more than one vessel to the same berthing position at the same period.

$$\sum_{v \in V} \boldsymbol{x}_{vbt} \leq 1 \ \forall b \in B, \ t \in T \tag{3}$$

Constraint set (4) guarantees that the start service period for a vessel should be greater than the period when the assigned berthing position is available for service.

$$\sum_{b \in B} \sum_{t \in T} \left(\theta_b^{ber} \boldsymbol{x}_{vbt} \right) \leq \boldsymbol{\theta}_v^{st} \ \forall v \in V \tag{4}$$

Constraint set (5) ensures to set the start service period for a vessel to be greater than its arrival period at the MCT.

$$\theta_b^{ber} \boldsymbol{x}_{vbt} + \sum_{v' \in V: v' \neq v} \sum_{t' \in T: t' < t} \left(\theta_{v'b}^{ht} \boldsymbol{x}_{v'bt'} + \boldsymbol{\theta}_{v'bt'}^{it} \right) + \boldsymbol{\theta}_{vbt}^{it} \geq \theta_v^{a} \boldsymbol{x}_{vbt} \ \forall v \in V, b \in B, \ t \in T \tag{5}$$

Constraint set (6) calculates the start service period for each arriving vessel at the MCT.

$$\boldsymbol{\theta}_v^{st} \geq \theta_b^{ber} \boldsymbol{x}_{vbt} + \sum_{v' \in V: v' \neq v} \sum_{t' \in T: t' < t} \left(\theta_{v'b}^{ht} \boldsymbol{x}_{v'bt'} + \boldsymbol{\theta}_{v'bt'}^{it} \right) + \boldsymbol{\theta}_{vbt}^{it} - \Upsilon(1 - \boldsymbol{x}_{vbt}) \quad \forall v \in V,$$

$$b \in B, \quad t \in T \tag{6}$$

Constraint set (7) shows that if the service of a vessel at the assigned berthing position starts at period $\boldsymbol{\theta}_v^{st}$, the vessel must leave that berthing position at period $\boldsymbol{\theta}_v^{st} + \theta_{vb}^{ht}$.

$$\boldsymbol{x}_{vb\left(\boldsymbol{\theta}_v^{st}\right)} = \boldsymbol{q}_{vb\left(\boldsymbol{\theta}_v^{st} + \theta_{vb}^{ht}\right)} \ \forall v \in V, b \in B, t \in T \tag{7}$$

The horizontal and vertical spatial requirements are imposed by constraint sets (8) and (9), respectively. Specifically, the length of the assigned berthing position should be longer than the vessel length plus the considered minimum horizontal clearance due to safety issues. Moreover, the depth of the assigned berthing position should be greater than the summation of the draft of the vessel and the considered minimum vertical clearance.

$$\left(L_v^{ves} + P_v^{hor}\right)\boldsymbol{x}_{vbt} \leq L_b^{ber} \quad \forall v \in V, b \in B, \quad t \in T \tag{8}$$

$$\left(D_v^{ves} + P_v^{ver}\right)\boldsymbol{x}_{vbt} \leq D_b^{ber} \quad \forall v \in V, b \in B, \quad t \in T \tag{9}$$

Constraint set (10) estimates the waiting periods for each arriving vessel at the MCT.

$$\boldsymbol{\theta}_v^{wt} \geq \boldsymbol{\theta}_v^{st} - \theta_v^a \quad \forall v \in V \tag{10}$$

Constraint set (11) calculates the finish service period for each arriving vessel at the MCT.

$$\boldsymbol{\theta}_v^{ft} = \boldsymbol{\theta}_v^{st} + \sum_{b \in B}\sum_{t \in T}\left(\theta_{vb}^{ht}\boldsymbol{x}_{vbt}\right) \quad \forall v \in V \tag{11}$$

Constraint set (12) computes the late departure periods for each arriving vessel at the MCT.

$$\boldsymbol{\theta}_v^{lt} \geq \boldsymbol{\theta}_v^{ft} - \theta_v^d \quad \forall v \in V \tag{12}$$

The nature of all the decision variables, auxiliary variables, and parameters of the SCBSP mathematical model is defined in constraint sets (13) to (15).

$$\boldsymbol{x}_{vbt}, \boldsymbol{q}_{vbt} \in \mathbb{B} \quad \forall v \in V, \quad b \in B, \quad t \in T \tag{13}$$

$$\boldsymbol{\theta}_{vbt}^{it}, \boldsymbol{\theta}_v^{st}, \boldsymbol{\theta}_v^{ft}, \boldsymbol{\theta}_v^{wt}, \boldsymbol{\theta}_v^{lt}, m, n, p, \theta_v^a, \quad \theta_b^{ber}, \quad \widetilde{\theta_v^{ht}}, \theta_{vb}^{ht}, \quad \theta_v^d \quad \in \mathbb{N} \quad \forall v \in V, \quad b \in B, \quad t \in T \tag{14}$$

$$D_v^{ves}, \quad D_v^{ber}, \quad L_v^{ves}, \quad L_v^{ber}, \quad P_v^{ver}, P_v^{hor}, \quad \varphi_v^{wt}, \quad \varphi_v^{ht}, \quad \varphi_v^{lt}, \quad \Upsilon \in \mathbb{R}^+ \quad \forall v \in V \tag{15}$$

5. Solution strategy

In this study, the developed SCBSP mathematical model was solved using the proposed UIMA. The key steps of UIMA are illustrated in Figure 2. The data, required for the execution of UIMA, are loaded in the first step. Then, the initial population is generated based on the two strategies:

(1) a First Come First Served with Spatial Requirements (FCFS-SR) heuristic for the first half of the whole population; and

(2) the next half of the population is generated randomly (more details can be found in Dulebenets *et al.*, 2018).

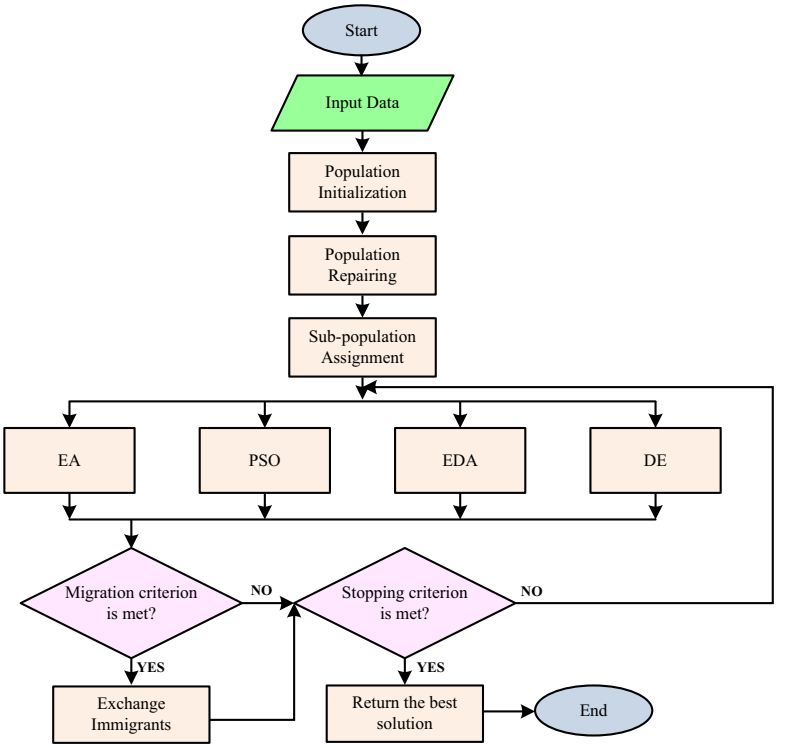

Figure 2.
The basic UIMA steps

Each solution in the population is checked by a repairing operator to be repaired in case of infeasibility. Then, the generated initial population is split into four sub-populations. The first half of each sub-population is selected from the solutions generated using FCFS-SR, and the next half is chosen from the randomly generated solutions. Each island is assigned one of the generated sub-populations. Afterwards, UIMA starts the execution of all four metaheuristics at the same time. Then, the defined migration criterion is checked, and in case of satisfaction, the migration process is conducted. Throughout the migration process, a specific number of solutions (called "immigrants"), selected from each island, are exchanged between all the possible pairs of the islands. In the next step, the stopping criterion is checked, and in case of satisfaction, the algorithm returns the best solution discovered over all the islands; otherwise, the four developed algorithms will continue the search on their specific islands for another iteration.

A population in UIMA includes a number of candidate solutions to the SCBSP mathematical model. Figure 3 illustrates a two-dimensional solution (as it consists of two rows) adopted in this study. The row at the top includes the indices of vessels, and the lower row accommodates the berthing positions, where the vessels at the row above are assigned

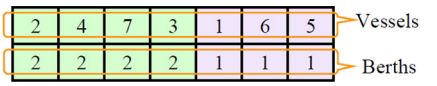

Figure 3.
The solution representation

to. For example, vessels "7" and "5" are assigned for service at berthing positions "2" and "1", respectively.

5.1 The topology of universal island-based algorithm

As discussed earlier, UIMA splits the population into four sub-populations, and each sub-population is assigned to one of the UIMA islands. After that, each island is searched using one of the population-based metaheuristic algorithms, which have been widely used in the BSP and EA literature (i.e. EA, PSO, EDA and DE). This study refers to Dulebenets *et al.* (2018), Ting *et al.* (2014), Izquierdo *et al.* (2012), and Arabani *et al.* (2011) for a detailed description of the EA, PSO, EDA, and DE algorithms, respectively. The UIMA topology is illustrated in Figure 4. A particular area of the search space (i.e. island) is explored and exploited by one of the adopted algorithms independently. The UIMA algorithm allows migration of individuals between the islands once a pre-defined migration criterion is satisfied (i.e. the islands are not fully isolated). While the canonical island-based metaheuristics execute the same type of metaheuristic algorithm on each one of the islands (Alba and Tomassini, 2002), UIMA deploys various types of metaheuristics for different islands that rely on different operators to explore and exploit the search space.

In detail, each metaheuristic algorithm plays a specific role in UIMA, given its unique features. The EA and DE algorithms are founded based on the principles of natural selection and survival of the fittest. The EA and DE algorithms select a specific number of solutions using the selection operators. The selected solutions are used by the crossover and mutation operators to generate the new solutions (Eiben and Smith, 2003). Crossover is used for exploration of various domains of the search space, while mutation is used to exploit the promising domains for superior solutions. The EA and DE algorithms are controllable search tools in the UIMA algorithm, as they can discover new areas in the search space by adopting the appropriate values for the crossover and mutation probabilities. The EA and DE algorithms have been widely used in the BSP literature (Liu *et al.*, 2016; Şahin and Kuvvetli, 2016; Tsai *et al.*, 2016; Dulebenets *et al.*, 2018).

On the other hand, the PSO algorithm is recognized as a swarm-intelligence algorithm, which plays the role of a fast local search tool in UIMA (Kennedy and Eberhart, 1997). The PSO algorithm is inspired by flocking behavior of birds and schooling behavior of fish (Ting *et al.*, 2014; Arabani *et al.*, 2011). The population in PSO is composed of a group of particles. A position and a velocity are assigned to each particle. These two features of particles enable them to move across the search space and discover the new areas. Ting *et al.* (2014) and Wang *et al.* (2012) adopted a PSO to solve BSPs. While the EA, PSO, and DE algorithms stochastically explore and exploit the search space, the EDA algorithm deploys statistical tools throughout the search process (Izquierdo *et al.*, 2012). A number of studies on scheduling problems used EDA as a solution methodology (Izquierdo *et al.*, 2012). The

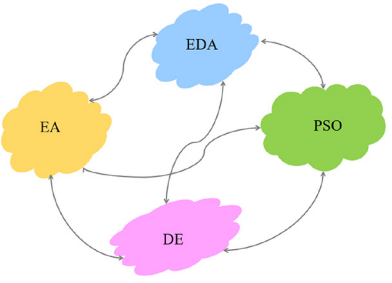

Figure 4.
A schematic illustration of the UIMA topology

metaheuristic algorithms, adopted for the UIMA islands, are further described in Sections 5.1.1-5.1.4 of the manuscript.

5.1.1 Evolutionary algorithm. The EAs, classified as population-based metaheuristic algorithms, are inspired by the principles of nature and evolve based on the survival of the fittest individual concept (Eiben and Smith, 2003). In EAs, a specific number of solutions, which is called a "population", can be generated randomly or considering the problem-specific properties. The fitness values of the generated solutions are evaluated based on the values of the objective function, adopted for the problem. The parent chromosomes are selected from the population (typically, based on their fitness) and undergo the crossover and mutation operations to produce and mutate the offspring chromosomes. The offspring chromosomes are further evaluated, and only a portion of them will be moved to the next generation (typically, based on their fitness). In general, the EA algorithm converges once a certain stopping criterion is met (e.g. maximum number of generations). In this study, the EA algorithm will be executed on one of the UIMA islands. Therefore, the EA stopping criterion will be defined by UIMA (similar to PSO, EDA, and DE, which will be executed on the other UIMA islands). Figure 5 presents the main EA steps to be performed on a given island in each UIMA iteration.

In step 0 of Figure 5, the data structure, required for storing the EA sub-population fitness values, is initialized. The fitness values of the chromosomes, assigned to the EA sub-population, are calculated in step 1 using equation (1) – see Section 4 of the manuscript. A specific number of chromosomes (equal to the sub-population size) are selected as parents in step 2. The crossover and mutation operators are applied to the selected parents in steps 3 and 4, respectively, to produce and mutate the offspring. The newly generated offspring are checked in step 5 to be repaired in case of infeasibility. The fitness value of each offspring is calculated in step 6. After that, another selection operator is applied to the generated and repaired offspring to establish a new EA sub-population for the next iteration of UIMA (step 7). The EA sub-population, selected for the next iteration of UIMA, is updated by applying the elitism strategy in step 8. In step 9, the EA sub-population is returned to UIMA. The Stochastic Universal Sampling and Binary Tournament selection operators were adopted to identify the parent and offspring chromosomes, respectively. The order crossover and swap mutation operators were applied to produce and mutate the offspring chromosomes,

Algorithm 1: Evolutionary Algorithm (EA)

$\boldsymbol{EA}(EASubPop, \sigma^c, \sigma^m, TourSize, InData)$

in: $EASubPop$ – the assigned EA sub-population; σ^c – crossover probability; σ^m – mutation probability; $TourSize$ – the number of chromosomes attending each tournament; $InData$ – the SCBSP input data

out: $EASubPop$

0: $|FitVals| \leftarrow |EASubPop|$ ◁ Initialization

1: $FitVals \leftarrow \boldsymbol{Fitness}(EASubPop, InData)$ ◁ Estimates the fitness value for each chromosome

2: $Parents \leftarrow \boldsymbol{SUS}(EASubPop, FitVals)$ ◁ Selects the parents

3: $Offspring \leftarrow \boldsymbol{Crossover}(Parents, \sigma^c)$ ◁ Applies the crossover operator

4: $Offspring \leftarrow \boldsymbol{Mutation}(Offspring, \sigma^m)$ ◁ Applies the mutation operator

5: $Offspring \leftarrow \boldsymbol{Repair}(Offspring)$ ◁ Repairs the infeasible offspring

6: $FitVals \leftarrow \boldsymbol{Fitness}(Offspring, InData)$ ◁ Estimates the fitness value for each offspring

7: $EASubPop \leftarrow \boldsymbol{TourSel}(Offspring, FitVals, TourSize)$ ◁ Establishes the next generation sub-population

8: $EASubPop \leftarrow \boldsymbol{Elitism}(EASubPop)$ ◁ Applies the elitism strategy

9: **return** $EASubPop$

Figure 5.

respectively. For a detailed description of the adopted selection, crossover, and mutation operators this study refers to Eiben and Smith (2003) (Figure 5).

5.1.2 Particle swarm optimization. The PSO algorithm is another population-based metaheuristic, introduced by Kennedy and Eberhart (1997). The PSO algorithm is inspired by schooling behavior of fish and flocking behavior of birds. In detail, when a given particle is moving in a group of particles (a bird in a flock), its move is remarkably affected by the distances between that particle and the other particles. A position and a velocity are the two features of a given particle, which have the major effects on the aforementioned distances. Two factors are defined in PSO to adjust the position of each particle:

(1) *pBest* – the best position that a particle has experienced so far throughout the search process; and

(2) *gBest* – the best position achieved so far by all the particles.

The values of *pBest* and *gBest* are updated throughout the search process. Figure 6 presents the main PSO steps to be performed on a given island in each UIMA iteration.

In step 0 of Figure 6, the data structure, required for storing the PSO sub-population fitness values, is initialized. In step 1, the solutions with the integer-coded representation, adopted within UIMA, are mapped to the solutions with the real-coded representation, which will be further used by PSO. The fitness values of the particles, composing the PSO sub-population, are calculated in step 2 using equation (1). The two key PSO parameters (i.e. *pBest* and *gBest*) are recorded in steps 3 and 4. The current velocity and position of each particle are retrieved in step 5. The velocity and the position of each particle are updated in steps 6 and 7, respectively. The PSO sub-population is updated in step 8. The newly generated particles are checked in step 9 to be repaired in case of infeasibility. The fitness values of the newly generated particles are calculated in step 10. In step 11, the solutions with the real-coded representation, used by PSO, are mapped back to the solutions with the integer-coded representation. In step 12, the PSO sub-population is returned to UIMA.

Algorithm 2: Particle Swarm Optimization (PSO)

$\boldsymbol{PSO}(PSOSubPop, C_1, C_2, W, InData)$
in: $PSOSubPop$ – the assigned PSO sub-population; C_1 – cognition component; C_2 – social component; W – inertia weight; $InData$ – the SCBSP input data
out: $PSOSubPop$

0: $|FitVals| \leftarrow |PSOSubPop|$ ◁ Initialization
1: $PSOSubPop \leftarrow \boldsymbol{RealMapping}(PSOSubPop)$ ◁ Mapping to the real-coded representation
2: $FitVals \leftarrow \boldsymbol{Fitness}(PSOSubPop, InData)$ ◁ Estimates the fitness value for each solution
3: $pBest \leftarrow \boldsymbol{pBestFinder}(PSOSubPop, FitVals)$ ◁ Records $pBest$ for each particle
4: $gBest \leftarrow \boldsymbol{gBestFinder}(PSOSubPop, FitVals)$ ◁ Records $gBest$ over all the particles
5: $[PrtclVel, PrtclPos] \leftarrow \boldsymbol{Retrieve}(PSOSubPop)$ ◁ Retrieves the current velocity and position
6: $PrtclVel \leftarrow \boldsymbol{VelUpdate}(PSOSubPop, PrtclVel, pBest, gBest, C_1, C_2, W)$ ◁ Updates the particle velocity
7: $PrtclPos \leftarrow \boldsymbol{PosUpdate}(PSOSubPop, PrtclPos, PrtclVel)$ ◁ Updates the particle position
8: $PSOSubPop \leftarrow \boldsymbol{SubPopUpdate}(PSOSubPop, PrtclVel, PrtclPos)$ ◁ Updates the PSO sub-population
9: $PSOSubPop \leftarrow \boldsymbol{Repair}(PSOSubPop)$ ◁ Repairs the infeasible solutions
10: $FitVals \leftarrow \boldsymbol{Fitness}(PSOSubPop, InData)$ ◁ Estimates the fitness value for each particle
11: $PSOSubPop \leftarrow \boldsymbol{IntMapping}(PSOSubPop)$ ◁ Mapping to the integer-coded representation
12: **return** $PSOSubPop$

Figure 6.

Since the PSO algorithm conducts operations with the real-coded solutions, the solutions, provided by UIMA that have the integer-coded representation, should be mapped to the solutions with the real-coded representation. Figure 7 illustrates a particle representation, adopted within PSO (similar to Ting *et al.*, 2014). There are 7 arriving vessels in Figure 7, which should be served at two berthing positions. To encode the berthing positions in the lower row, a group of real numbers are randomly generated in the range [0, 2]. The rounded integer part of the real number in the lower row stands for the berthing position that the vessel in the upper row is assigned to. The fractional part determines the service order of vessels at each berthing position. In particular, vessels "1", "6", and "5" should have service orders "2", "1", and "3" at berthing position "1" (i.e. the associated fractional parts are sorted in the ascending order).

The velocity and the position of each particle are updated (steps 6 and 7 of Figure 6) based on the following two equations (Arabani *et al.*, 2011; Ting *et al.*, 2014):

$$V_p^{new} = W \cdot V_p^{old} + C_1 \cdot R_1 \cdot \left(pBest_p - P_p^{old}\right) + C_2 \cdot R_2 \cdot \left(gBest - P_p^{old}\right) \tag{16}$$

$$P_p^{new} = P_p^{old} + V_p^{new} \tag{17}$$

where: V_p^{new} and V_p^{old} are the new and current velocities of particle p, respectively; P_p^{old} and P_p^{new} are the new and current positions of particle p, respectively; W is the inertia weight (the value of the inertia weight generally varies from 0.50 to 1.00 – Ting *et al.*, 2014); C_1 and C_2 are the cognition and social components, respectively; R_1 and R_2 are two random numbers in the range [0, 1]; $pBest_p$ is the best position that particle p has experienced so far throughout the search process; $gBest$ is the best position achieved so far by all the particles.

5.1.3 Estimation of distribution algorithm. The EDA population-based metaheuristic was developed by Larrañaga and Lozano (2001). Unlike the other three algorithms that were adopted in this study for the UIMA islands (i.e. EA, PSO, and DE), EDA uses statistical tools (i.e. a probability distribution) to generate new solutions. On the other hand, EA, PSO and DE generate new solutions stochastically (i.e. using stochastic operators for crossover and mutation or stochastic operators for updating velocities and positions of particles). The statistical information is collected from a group of promising solutions in the population (Izquierdo *et al.*, 2012). In EDA, a probability matrix should be established, which includes the probability of assigning a vessel to each one of the available berthing positions in the developed SCBSP mathematical model. Figure 8 presents the main EDA steps to be performed on a given island in each UIMA iteration.

In step 0 of Figure 8, the data structure, required for storing the EDA sub-population fitness values, is initialized. The fitness values of the solutions, assigned to the EDA sub-population, are calculated in step 1 using equation (1). The probability matrix is updated and shaken in steps 2 and 3, respectively. A portion of the elite solutions is transferred from the previous EDA sub-population into the current EDA sub-population in step 4. In step 5, the remainder of the population is filled based on the probability matrix. The newly generated solutions are checked in step 6 to be repaired in case of infeasibility. The fitness

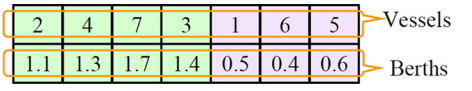

Figure 7.
The PSO particle representation

Algorithm 3: Estimation of Distribution Algorithm (EDA)

$\boldsymbol{EDA}(EDASubPop, EDASubPop^{g-1}, \varepsilon, \psi, InData)$

in: $EDASubPop$ – the assigned EDA sub-population; $EDASubPop^{g-1}$ – the assigned EDA sub-population in the previous generation; ε – shaking coefficient; ψ – elitism coefficient; $InData$ – the SCBSP input data

out: $EDASubPop$

0: $|FitVals| \leftarrow |EDASubPop|$ ◁ Initialization

1: $FitVals \leftarrow \boldsymbol{Fitness}(EDASubPop, InData)$ ◁ Estimates the fitness value for each solution

2: $PrbSet \leftarrow \boldsymbol{UpdatePrb}(EDASubPop, FitVals, \psi)$ ◁ Updates the probability matrix

3: $PrbSet \leftarrow \boldsymbol{ShakePrb}(EDASubPop, PrbSet, \varepsilon)$ ◁ Shakes the probability matrix

4: $EDASubPop \leftarrow \boldsymbol{Elitism}(EDASubPop^{g-1}, FitVals, \psi)$ ◁ Transfers a portion of the elite solutions

5: $EDASubPop \leftarrow \boldsymbol{PrbExplore}(EDASubPop, PrbSet, \psi)$ ◁ Fills the remainder of the sub-population

6: $EDASubPop \leftarrow \boldsymbol{Repair}(EDASubPop)$ ◁ Repairs the infeasible solutions

7: $FitVals \leftarrow \boldsymbol{Fitness}(EDASubPop)$ ◁ Estimates the fitness value for each solution

8: **return** $EDASubPop$

Figure 8.

value of each solution is calculated in step 7. In step 8, the EDA sub-population is returned to UIMA. The EDA probability matrix can be represented using matrix (18), which contains the number of arriving vessels (m) and the number of available berthing positions (n) in generation g – i.e. $m \times n$ matrix with m rows and n columns. More specifically, $P_{mn}(g)$ denotes the probability of assigning vessel m to berthing position n in generation g. Note that the summation of probabilities of assigning a given vessel to the available berthing positions should be equal to 1 (Izquierdo *et al.*, 2012).

$$P(g) = \begin{bmatrix} P_{11}(g) & \cdots & P_{1n}(g) \\ \vdots & \ddots & \vdots \\ P_{m1}(g) & \cdots & P_{mn}(g) \end{bmatrix} \tag{18}$$

The EDA probability matrix is updated in each generation in the following manner. First, the fittest solutions from the current sub-population are selected. The number of the fittest solutions will be set to $round(\psi \cdot |EDASubPop|)$ – i.e. ψ percent of the EDA sub-population. The selected solutions will be further used to update the probability matrix based on the following equation:

$$P_{mn}(g) = \frac{Num_{mn}(g)}{\sum_{k \in B} Num_{mk}(g)} \tag{19}$$

where: $Num_{mn}(g)$ is the number of times vessel m has been assigned to berthing position n in the solutions selected in generation g; and the denominator represents the number of times vessel m has been assigned to all the available berthing positions in generation g.

The shaking operator is implemented in EDA (step 3) to increase the population diversity, which further facilitates exploration and exploitation in the search space and decreases the probability of the premature convergence. The shaking operator selects a specific number of vessels ($Num^{vs} < m$). Then, the rows, corresponding to the selected vessels in the probability matrix, are independently perturbed (i.e. the probability values of assigning a vessel to different berthing positions are exchanged randomly). The elitism operator is implemented in each iteration of EDA (step 4) to generate a portion of the new

sub-population. In particular, the new sub-population is composed of $round(\psi \cdot |EDASubPop|)$ solutions – i.e. ψ percent of the fittest solutions from the EDA sub-population in the previous generation. Function *PrbExplore* will be used to fill out the remainder of the new EDA sub-population. Figure 9 illustrates the process of assigning a vessel to a berthing position by function *PrbExplore*. First of all, the corresponding row for a given vessel in the probability set is considered. As it can be observed, there are 7 berthing positions for the case, which is considered in Figure 9. Then, the list of berthing positions is sorted based on the assignment probability in the ascending order. The probabilities are cumulatively summed up. A random number is generated in the range [0, 1] (e.g. 0.48). Berthing position "1" is selected, as 0.48 is less than 0.56 and greater than 0.32. The same procedure is repeated for the other vessels as well.

5.1.4 Differential evolution. As indicated earlier, DE has some similarities with EAs, as it is inspired by the principles of nature and evolves based on the survival of the fittest individual concept. The DE algorithm was introduced by Storn and Price (1997) for the first time. Similar to EAs, DE relies on the crossover and mutation operators to perform the search process. However, unlike typical EAs that can work with both integer- and real-coded chromosomes, DE particularly relies on real-coded chromosomes (that are also called as "target vectors" – Arabani *et al.*, 2011). Figure 10 presents the main DE steps to be performed on a given island in each UIMA iteration.

In step 0 of Figure 10, the data structures, required for storing the DE sub-population fitness values, are initialized. In step 1, the solutions with the integer-coded representation, adopted within UIMA, are mapped to the solutions with the real-coded representation, which will be further used by DE. The fitness values of the target vectors, composing the DE sub-population, are calculated in step 2 using equation (1). The mutation and crossover operators are applied to the target vectors in steps 3 and 4, respectively, to produce the trial vectors. The newly generated trial vectors are checked in step 5 to be repaired in case of

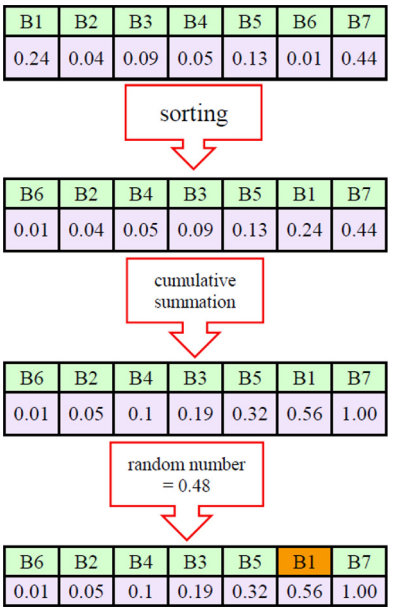

Figure 9.
The new EDA sub-population generation

Algorithm 4: Differential Evolution (DE)

$\boldsymbol{DE}(DESubPop, \alpha, \sigma^c, InData)$

in: $DESubPop$ – the assigned DE sub-population; α – mutation coefficient; σ^c – crossover probability; $InData$ – the SCBSP input data

out: $DESubPop$

0: $|FitVals^{tar}| \leftarrow |DESubPop|$; $|FitVals^{tri}| \leftarrow |DESubPop|$ ◁ Initialization

1: $DESubPop^{tar} \leftarrow \boldsymbol{RealMapping}(DESubPop)$ ◁ Mapping to the real-coded representation

2: $FitVals^{tar} \leftarrow \boldsymbol{Fitness}(DESubPop^{tar}, InData)$ ◁ Estimates the fitness value for each target vector

3: $DESubPop^{tri} \leftarrow \boldsymbol{Mutation}(DESubPop^{tar}, \alpha)$ ◁ Applies the mutation operator

4: $DESubPop^{tri} \leftarrow \boldsymbol{Crossover}(DESubPop^{tar}, DESubPop^{tri}, \sigma^c)$ ◁ Applies the crossover operator

5: $DESubPop^{tri} \leftarrow \boldsymbol{Repair}(DESubPop^{tri})$ ◁ Repairs the infeasible solutions

6: $FitVals^{tri} \leftarrow \boldsymbol{Fitness}(DESubPop^{tri}, InData)$ ◁ Estimates the fitness value for each trial vector

7: $DESubPop \leftarrow \boldsymbol{Selection}(DESubPop^{tar}, FitVals^{tar}, DESubPop^{tri}, FitVals^{tri})$ ◁ Selection

8: $DESubPop \leftarrow \boldsymbol{Elitism}(DESubPop)$ ◁ Applies the elitism strategy

9: $DESubPop \leftarrow \boldsymbol{IntMapping}(DESubPop)$ ◁ Mapping to the integer-coded representation

10: **return** $DESubPop$

Figure 10.

infeasibility. The fitness value of each trial vector is calculated in step 6. After that, the selection operator is applied to the target vectors and the trial vectors to establish a new DE sub-population for the next iteration of UIMA (step 7). The DE sub-population, selected for the next iteration of UIMA, is updated by applying the elitism strategy in step 8. In step 9, the solutions with the real-coded representation, used by DE, are mapped back to the solutions with the integer-coded representation. In step 10, the DE sub-population is returned to UIMA.

Since the DE algorithm conducts operations with the real-coded solutions, the solutions, provided by UIMA that have the integer-coded representation, should be mapped to the solutions with the real-coded representation. In this study, Relative Position Indexing (RPI) will be used to map the integer-coded solutions into the real-coded solutions for the DE algorithm (Lichtblau, 2002). In Figure 11, the integer-coded vectors are mapped into the real-coded vectors using the RPI-format based on the following relationship: $RPI_{inx} = IC_{inx}/MAX_{IC}$, where: RPI_{inx} is the index of the vessel or the berthing position in the RPI-format; IC_{inx} is the integer-coded index of the vessel or the berthing position; and MAX_{IC} is the maximum value of the integer-coded vector for the vessels or the berthing positions. In the presented example, the RPI value for vessel "7" was estimated as 7/7 = 1.00 ("7" is the maximum value of the integer-coded vector, representing the arriving vessels), while the RPI value for berthing position "1" was estimated as 1/2 = 0.50 ("2" is the maximum value of the integer-coded vector, representing the available berthing positions). For a detailed description of typical mutation, crossover, and selection operators, which were adopted within the DE algorithm, this study refers to Arabani *et al.* (2011).

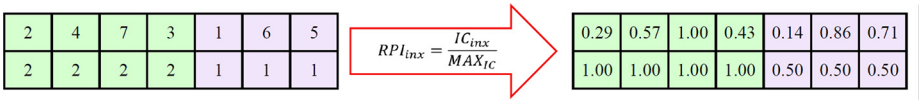

Figure 11.
Mapping the integer-coded vectors to the real-coded vectors

5.2 The major parameters of universal island-based algorithm

In addition to typical algorithmic parameters, which are used within the population-based metaheuristic algorithms that were adopted to search the UIMA islands, there are four other parameters/procedures, required to perform migration in island-based metaheuristic algorithms. One of the most important parameters is the frequency of migration, which determines how often the islands exchange the solutions between each other. The frequency of migration directly impacts performance of island-based metaheuristic algorithms (Alba and Tomassini, 2002). Moreover, there are three other UIMA parameters/procedures that should be properly set, including the following (Eiben and Smith, 2003):

(1) the number of immigrants;

(2) the individuals, which should migrate (i.e., immigrant selection); and

(3) the individuals, which must be removed on the destination island to make enough room for the migrating individuals.

A detailed description of the strategies adopted for the aforementioned parameters is provided in Sections 5.2.1-5.2.4 of the manuscript.

5.2.1 The frequency of migration. A deterministic approach is widely used for setting the frequency of migration (Eiben and Smith, 2003). The deterministic approach assumes that the migration process is implemented based on a specific number of algorithmic iterations/generations (a pre-determined number of fitness function evaluations or a target computational time can be used as deterministic migration criteria as well). However, if the migration is implemented too frequently, the population diversity decreases. On the other hand, if the migration occurs very few times throughout the algorithmic run, some sub-populations may converge to a local optimum and negatively affect efficiency of all the conducted computational efforts. To address the aforementioned drawbacks, some studies relied on an adaptive strategy to set the migration frequency (Kurdi, 2015). An adaptive approach was also used in this study as well, where the migration process was conducted when the improvements in the objective function values on two or more than two islands were less than $\lambda = 0.1$ per cent (determined based on the conducted parameter tuning analysis – see Section 6.2 of the manuscript) as compared to the objective function values in the first iteration after the previous migration.

5.2.2 The number of immigrants. The number of immigrants (N^{img}) is the number of individuals that should migrate from one island to the other ones. The number of immigrants should be determined properly, as a high number of immigrants may lead to a quick convergence to a similar solution on all the islands. Therefore, it is recommended to exchange a few solutions between the islands (Eiben and Smith, 2003). The number of immigrants in this study was set to $N^{img} = 4$ individuals (determined based on the conducted parameter tuning analysis – see Section 6.2 of the manuscript) for each migration process within UIMA.

5.2.3 Selection of the migrating individuals. Selection of the migrating individuals can be conducted based on different strategies (Eiben and Smith, 2003). In this study, a Roulette Wheel Selection operator was used to perform the immigrant selection. In the Roulette Wheel Selection, a selection probability is assigned to each one of the solutions in the sub-population [calculated using equation (20)]. The selection probability has a reverse relationship with the objective function value of a given solution (as the SCBSP mathematical model aims to minimize the objective function value, i.e. minimize the total vessel service cost).

$$\boldsymbol{Pr}_i = \frac{1/\boldsymbol{Z}_i}{\sum_{j \in subpop} 1/\boldsymbol{Z}_j} \tag{20}$$

where $\boldsymbol{Pr}_i$ is the selection probability of solution i; and $\boldsymbol{Z}_i$ is the objective function value, associated with solution i.

The probabilities, estimated for all the solutions, are cumulatively summed up. Each one of the solutions is allocated a portion of the range [0, 1]. A random number in the range [0, 1] is generated, and the corresponding solution that belongs to a portion of the roulette wheel is identified and chosen as an immigrant. This process is repeated until the required number of immigrants are selected (Eiben and Smith, 2003).

5.2.4 Removal of the individuals. To make enough room on the destination islands for the migrating solutions, a strategy should be developed to select a specific number of individuals for removal on the destination islands. A random selection of the solutions to be removed may lead to some negative consequences (e.g. a loss of some promising solutions on the destination islands). To avoid any potential negative effects on the algorithmic performance, an alternative Roulette Wheel Selection procedure was developed in this study. To select low-quality solutions for removal, the selection probability of the solutions is calculated using equation (21).

$$\boldsymbol{Pr}_i = \frac{\boldsymbol{Z}_i}{\sum_{j \in subpop} \boldsymbol{Z}_j} \tag{21}$$

Equation (21) was developed by making some revisions in equation (20). Application of equation (21) increases the probability of selecting less promising solutions for removal on the destination islands. To select a specific number of solutions for removal, all the steps that were required for the immigrant selection should be conducted in the same way (except the removal probability estimations for the solutions on the destination islands).

6. Numerical experiments

In this section of the manuscript, a detailed evaluation of the developed UIMA algorithm is conducted. UIMA will be compared to the EA, PSO, EDA, and DE algorithms, which are executed separately (i.e. without using an island concept). Moreover, the developed SCBSP mathematical model will be solved using a group of single-solution-based metaheuristic algorithms, which have been widely used in the BSP literature, including the following:

- variable neighborhood search – VNS;
- tabu search – TS; and
- simulated annealing – SA.

A detailed description of the VNS, TS, and SA algorithms can be found in Hansen *et al.* (2008), Cordeau *et al.* (2005), and Emde and Boysen (2016), respectively. The SCBSP mathematical model will be encoded in the General Algebraic Modeling System (GAMS) environment and solved using CPLEX to the global optimality. The global optima, obtained by CPLEX, will be further used to assess the quality of solutions, obtained by UIMA and other candidate metaheuristic algorithms (i.e. EA, PSO, EDA, DE, VNS, TS and SA). Furthermore, the scope of numerical experiments includes the analysis of managerial

insights, aiming to determine how changes in the unit cost components of the SCBSP mathematical model (i.e. unit waiting cost, unit handling cost, and unit late departure cost) could impact the berth scheduling decisions.

All the metaheuristic algorithms were encoded in the MATLAB 2016a environment in this study, and the numerical experiments were conducted using an Alienware CPU with an Intel® Core™ i7-7700K processor, 32 GB of RAM, and Windows 10 Operating System. The following sections of the manuscript elaborate on the key steps of the conducted numerical experiments, including:

- input data generation;
- algorithmic parameter tuning;
- solution quality evaluation against exact optimization;
- UIMA evaluation against the alternative metaheuristics; and
- managerial insights.

6.1. Input data generation

Table II provides detailed information regarding the values that were adopted for the parameters of the developed SCBSP mathematical model. The adopted parameter values were based on the BSP literature, freight operations literature and relevant online sources (Imai *et al.*, 2008; Mauri *et al.*, 2016; Venturini *et al.*, 2017; Dulebenets *et al.*, 2018; Eniram, 2019). The arrival pattern of vessels was modeled using an exponential distribution with an average inter-arrival time of 2 periods. Each period was assumed to be equal to 1 hour. The following types of vessels were modeled as a part of this study:

- Panamax (draft: 41.0 feet, length: 820.2 feet);
- Panamax Max (draft: 41.0 feet, length: 951.4 feet);
- Post Panamax (draft: 42.7 feet, length: 935.0 feet);
- Post Panamax Plus (draft: 47.6 feet, length: 984.3 feet);
- New Panamax (draft: 49.9 feet, length: 1200.8 feet); and
- Triple E (draft: 50.9 feet, length: 1312.3 feet).

The handling time of the arriving vessels at their preferred berthing positions was assumed to vary from 10 periods to 24 periods. The negotiated departure period for a given vessel was set based on its vessel arrival period and the number of periods, required for service of that vessel at its preferred berthing position.

The minimum vertical clearance for the arriving vessels was assumed to vary from 4 feet to 8 feet. The minimum horizontal clearance for the arriving vessels was assumed to vary from 50 feet to 100 feet. The unit waiting cost of vessels ranged from US$1,000/period to US$5,000/period. On the other hand, the unit handling cost of vessels was assumed to vary from US$50,000/period to US$70,000/period. The unit late departure cost of vessels ranged from US$5,000/period to US$10,000/period. Based on the values adopted for the parameters of the SCBSP mathematical model, the small-size problem instances (P1-P24 – with 5 to 20 vessels and 2 to 5 berthing positions) and the large-size problem instances (P25-P48 – with 65 to 110 vessels and 4 to 10 berthing positions) were developed to conduct the numerical experiments.

Parameter	Selected Value
Number of arriving vessels: m (vessels)	Values are changed based on the problem instance
Number of available berthing positions: n (berthing positions)	Values are changed based on the problem instance
Number of periods: p (periods)	Values are changed based on the problem instance
Arrival time of vessel v at the MCT: $\theta_v^a,\ v \in V$ (period)	$\theta_{v+1}^a = round^{+}\left(\theta_v^a + \Delta\theta\right)\ \ \forall v \in V$[a]
Average inter-arrival time of vessels: $\Delta\theta$ (periods)	$\Delta\theta = exp[2]$[b]
Period when berthing position b is available for the first time in the planning horizon: $\theta_b^{ber},\ b \in B$ (period)	$\theta_b^{ber} = 0\ \ \forall b \in B$
Periods required for handling vessel v at its preferred berthing position: $\widetilde{\theta_v^{ht}},\ v \in V$ (periods)	$\widetilde{\theta_v^{ht}} = round(U[10;24])\ \ \forall v \in V$[c,d]
Periods required for handling vessel v at berthing position b: $\theta_{vb}^{ht},\ v \in V,\ b \in B$ (periods)	$\theta_{vb}^{ht} = round^{+}\left(\widetilde{\theta_v^{ht}} \cdot \left(1 + 0.03 \cdot abs\left(B_v^{pr} - B_v^{as}\right)\right)\right)\ \ \forall v \in V,\ b \in B$
Negotiated departure period for vessel v: $\theta_v^d,\ v \in V$ (period)	$\theta_v^d = round\left(\theta_v^a + \widetilde{\theta_v^{ht}} \cdot U[1.20;1.50]\right)\ \ \forall v \in V$
Draft of vessel v: $D_v^{ves},\ v \in V$ (feet)	[41.0; 41.0; 42.7; 47.6; 49.9; 50.9]
Depth of berthing position b: $D_b^{ber},\ b \in B$ (feet)	$D_b^{ber} = 1.20 \cdot \min_v\left(D_v^{ves}\right) + U\left[0; 1.20 \cdot \max_v\left(D_v^{ves}\right) - \min_v\left(D_v^{ves}\right)\right]\ \ \forall b \in B$
Length of vessel v: $L_v^{ves},\ v \in V$ (feet)	[820.2; 951.4; 935.0; 984.3; 1200.8; 1312.3]
Length of berthing position b: $L_b^{ber},\ b \in B$ (feet)	$L_b^{ber} = 1.20 \cdot \min_v\left(L_v^{ves}\right) + U\left[0; 1.20 \cdot \max_v\left(L_v^{ves}\right) - \min_v\left(L_v^{ves}\right)\right]\ \ \forall b \in B$
Minimum vertical clearance requirement for vessel v (feet): $P_v^{ver},\ v \in V$ (feet)	$P_v^{ver} = U[4;8]\ \ \forall v \in V$
Minimum horizontal clearance requirement for vessel v: $P_v^{hor},\ v \in V$ (feet)	$P_v^{hor} = U[50;100]\ \ \forall v \in V$
Unit waiting cost for vessel v: $\varphi_v^{wt},\ v \in V$ (US$/period)	$\varphi_v^{wt} = U[1000;5000]\ \ \forall v \in V$
Unit handling cost for vessel v: $\varphi_v^{ht},\ v \in V$ (US$/period)	$\varphi_v^{ht} = U[50000;70000]\ \ \forall v \in V$
Unit late departure cost for vessel v: $\varphi_v^{lt},\ v \in V$ (US $/period)	$\varphi_v^{lt} = U[5000;10000]\ \ \forall v \in V$
Large positive number: Υ	10000

Notes: [a]Notation $round^{+}(Val_1)$ is for the nearest integer larger than Val_1; [b]notation $exp[Val_2]$ is for the exponentially distributed pseudorandom numbers with an average of Val_2; [c] notation $round(Val_3)$ is for the nearest integer to Val_3; [d]notation $U[Val_4; Val_5]$ is for the uniformly distributed pseudorandom numbers that range from Val_4 to Val_5

Table II. Data generation for the SCBSP mathematical model

While the small-size problem instances were used to compare the candidate metaheuristic algorithms against exact optimization, the large-size problem instances were used for a detailed evaluation of UIMA against the alternative metaheuristics based on different performance indicators.

6.2 Algorithmic parameter tuning

All the algorithms developed in this study have some parameters, which should be assigned the appropriate values to ensure adequate performance of the algorithms (Eiben and Smith, 2003; Dulebenets *et al.*, 2019; Dulebenets, 2019; Kavoosi *et al.*, 2019). The process of algorithmic parameter selection is called "parameter tuning" (Eiben and Smith, 2003). Two approaches have been commonly used in the literature for conducting the parameter tuning (Eiben and Smith, 2003):

(1) full factorial design; and

(2) Taguchi's scheme.

The Taguchi's scheme is used when the number of parameters to be tuned is significant, and consideration of all the possible combinations of the parameter values may be prohibitive in terms of the computational time required; therefore, only "the most favorable parameter combinations" are considered (Dulebenets, 2019). Given the fact that the number of parameters does not exceed 5 parameters for each one of the considered algorithms, the full factorial design methodology was used for the parameter tuning in this study. Based on the full factorial design methodology, all the possible combinations of the parameter values have to investigated to determine the best one, taking into account the tradeoff between the solution quality at termination and the computation time incurred.

A total of 4 large-size problem instances were randomly selected from all the generated large-size problem instances, described in Section 6.1 of the manuscript, to conduct the parameter tuning. Given stochastic nature of the candidate metaheuristic algorithms, a total of 10 replications were performed to determine the average objective function and computational time values for each problem instance. Table III summarizes the parameter tuning results for UIMA, EA, PSO, EDA, DE, VNS, TS, and SA. Note that the considered single-solution-based algorithms (VNS, TS, and SA) had to change two vessel to berth assignments (randomly selected) or two vessel to service order assignments (randomly selected) to create a new solution during the local search. The maximum number of iterations was used as a stopping criterion for UIMA, PSO, VNS, TS, and SA, while the maximum number of generations was used as a stopping criterion for EA, EDA, and DE.

6.3 Solution quality evaluation against exact optimization

CPLEX and all the candidate metaheuristic algorithms were executed for all the developed small-size problem instances, and the results are reported in Table AI and AII. Note that Table AI and AII are provided in Appendix, which accompanies this manuscript. The computational time limit and the relative optimality gap for CPLEX were set to 7200 seconds and 0.01, respectively, while the remainder of the CPLEX parameters remained default. A total of 10 replications were performed for each algorithm to determine the average objective function and computational time values for each small-size problem instance. Based on the results from the conducted analysis, CPLEX could return the global optimum only for the first 19 small-size problem instances. The computational time limit was not sufficient for CPLEX to solve problem instances P20-P24 to the global optimality. It was observed that the considered metaheuristic algorithms were able to return either optimal or near-optimal solutions for a significant number of the developed small-size problem instances. The maximum optimality gap of the developed UIMA algorithm did not exceed 2.50 per cent, which demonstrates a high accuracy of UIMA. The other

Algorithm	Parameter	Candidate values	Selected value
UIMA	Population size (*PopSize*) [a]	[240; 280; 320]	240
UIMA	Migration criterion (λ)	[0.1; 0.2; 0.3]	0.1
UIMA	Number of immigrants (N^{img})	[3; 4; 5]	4
UIMA	Stopping criterion (*MaxIter*)	[1200; 1400; 1600]	1600
EA	Population size (*PopSize*)[b]	[40; 50; 60]	60
EA	Crossover probability (σ^c)	[0.3; 0.5; 0.8]	0.8
EA	Mutation probability (σ^m)	[0.01; 0.04; 0.06]	0.06
EA	Number of chromosomes attending each tournament (*TourSize*)	[20; 30; 40]	40
EA	Stopping criterion (*MaxGen*)[b]	[4000; 5000; 6000]	6000
PSO	Population size (*PopSize*)[b]	[40; 50; 60]	60
PSO	Cognition component (C_1)	[1.5; 2.0; 2.5]	2.0
PSO	Social component (C_2)	[1.5; 2.0; 2.5]	1.5
PSO	Inertia weight (W)	[0.3; 0.5; 0.8]	0.5
PSO	Stopping criterion (*MaxIter*)[b]	[4000; 5000; 6000]	6000
EDA	Population size (*PopSize*)[b]	[40; 50; 60]	60
EDA	Shaking coefficient (ε)	[0.1; 0.3; 0.5]	0.1
EDA	Elitism coefficient (ψ)	[0.4; 0.6; 0.8]	0.6
EDA	Stopping criterion (*MaxGen*)[b]	[4000; 5000; 6000]	6000
DE	Population size (*PopSize*)[b]	[40; 50; 60]	60
DE	Mutation coefficient (α)	[0.4; 0.6; 0.8]	0.8
DE	Crossover probability (σ^c)	[0.3; 0.5; 0.6]	0.3
DE	Stopping criterion (*MaxGen*)[b]	[4000; 5000; 6000]	6000
VNS	Local search size $\left(S_{ls}^{VNS}\right)$	[15; 20; 25]	20
VNS	Stopping criterion (*MaxIter*)	[4000; 5000; 6000]	6000
TS	Tabu list size $\left(S_{tl}^{TS}\right)$	[10; 15; 20]	10
TS	Local search size $\left(S_{ls}^{TS}\right)$	[15; 20; 25]	20
TS	Stopping criterion (*MaxIter*)	[4000; 5000; 6000]	6000
SA	Initial Boltzmann temperature (τ)	[2000; 3000; 3500]	3500
SA	Temperature interval ($\Delta\tau$)	[0.10; 0.30; 0.50]	0.50
SA	Stopping criterion (*MaxIter*)	[4000; 5000; 6000]	6000

Note: [a]The UIMA population was evenly distributed between its four islands (i.e. the UIMA population of 240 individuals was distributed between its four islands in the way that the EA, PSO, EDA, and DE sub-populations would have 60 individuals per sub-population); [b]the population size and stopping criterion that were set for EA, PSO, EDA, and DE when they were executed independently

Table III. The algorithmic parameter tuning results based on the full factorial design methodology

population-based metaheuristic algorithms (EA, PSO, EDA, and DE) and single-solution-based algorithms (VNS, TS, and SA) showed an acceptable accuracy with the optimality gaps, which generally did not exceed ≈5 per cent for the generated small-size problem instances. Also, the considered metaheuristic algorithms were able to converge much faster as compared to CPLEX. Specifically, the computational time of the candidate metaheuristic algorithms did not exceed 60 sec over the generated small-size problem instances.

6.4 Universal island-based algorithm evaluation against the alternative metaheuristics

All the candidate metaheuristic algorithms were executed for all the developed large-size problem instances, and the results are reported in Table AIII and AIV. Note that Table AIII and AIV are provided in Appendix, which accompanies this manuscript. A

total of 10 replications were performed for each algorithm to determine the average objective function and computational time values for each large-size problem instance. Based on the results from the conducted analysis, the objective values by UIMA were superior to the ones, obtained by EA, PSO, EDA, and DE, on average by 11.03 per cent, 12.03 per cent, 12.92 per cent, and 11.76 per cent, respectively. Such outcome can be explained by the fact that UIMA uses EA, PSO, EDA, and DE as search tools to cover different areas (i.e. islands) of the search space and is able to more effectively explore various domains of the search space.

Moreover, migration of individuals from one UIMA island to the other islands enables the algorithm to effectively exploit the promising search space domains for superior solutions. On the other hand, execution of the considered population-based metaheuristic algorithms in isolation (i.e. the EA, PSO, EDA, and DE algorithms) imposes limitations on the search process, which further negatively affects the solution quality at termination of such algorithms. The numerical experiments show that the considered single-solution-based algorithms (VNS, TS and SA) generally performed worse in the terms of the solution quality as compared to UIMA and the considered population-based metaheuristic algorithms. Specifically, the objective values by UIMA were superior to the ones, obtained by VNS, TS, and SA, on average by 19.01 per cent, 20.21 per cent, and 19.13 per cent, respectively. Such finding can be supported by the fact that, unlike population-based metaheuristic algorithms, single-solution-based metaheuristic algorithms perform operations with just one solution in a given domain of the search space and its neighbor(s), which limits their explorative and exploitative capabilities.

In terms of the computational efforts, the average computational times, obtained over 10 replications for UIMA, EA, PSO, EDA, DE, VNS, TS, and SA, are equal to 211.81 sec, 155.60 sec, 70.29 sec, 132.26 sec, 86.34 sec, 11.62 sec, 12.74 sec, and 11.85 sec, respectively, for the generated large-size problem instances. Furthermore, the average computational times per iteration/generation, obtained over 10 replications for UIMA, EA, PSO, EDA, DE, VNS, TS, and SA, comprise 0.1324 sec, 0.0259 sec, 0.0117 sec, 0.0220 sec, 0.0144 sec, 0.0019 sec, 0.0021 sec, and 0.0020 sec, respectively, for the generated large-size problem instances. The average computational time did not fluctuate substantially from one iteration/generation to another for each one of the considered solution algorithms. The highest computational time was recorded for UIMA. Such pattern can be justified by the fact that the UIMA algorithm performs more steps throughout the search process as compared to the alternative population-based metaheuristic algorithms, which were executed independently, and the considered single-solution-based metaheuristic algorithms. However, the maximum UIMA computational time did not exceed 306 sec.

The scope of numerical experiments also includes the analysis of algorithmic performance throughout the search process. The average objective function values over 10 replications, returned by the considered metaheuristic algorithms in each generation/iteration (the term "generation" is related to EA, EDA, and DE, while the term "iteration" is related to UIMA, PSO, VNS, TS, and SA), are plotted in Figure 12 against the number of generations/iterations mapped to the range [0, 1]. Such mapping was adopted due to the differences in the stopping criteria, which were set for the considered metaheuristic algorithms. Specifically, UIMA was terminated after 1,600 iterations, while PSO, VNS, TS, and SA were terminated after 6,000 iterations. On the other hand, EA EDA, and DE were terminated after 6,000 generations. The convergence pattern diagrams for the considered algorithms are illustrated in Figure 12 for the 12 largest problem instances (i.e. P37-P48). Similar convergence patterns were observed for the remainder of the large-size problem

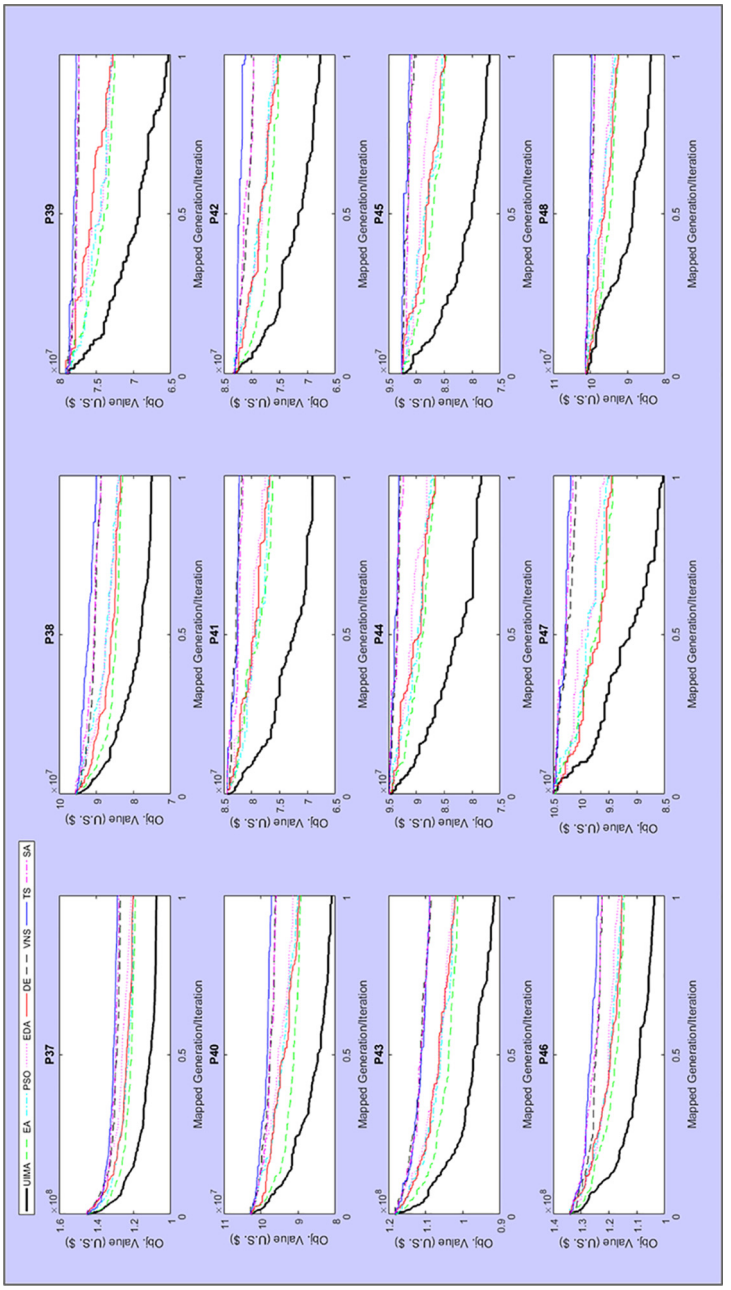

Figure 12.
The convergence patterns of all the considered algorithms for the large-size problem instances P37 to P48

instances. All the diagrams start from the same point as the same initial population generation strategy was adopted for all the considered metaheuristic algorithms (see Section 5 of the manuscript for more details). The convergence patterns demonstrate that the developed UIMA algorithm was able to move more effectively along the search space and discover superior solutions as compared to the alternative population-based and single-solution-based metaheuristic algorithms.

The numerical experiments, conducted as a part of this study, demonstrate a clear superiority of the developed UIMA algorithm over the population-based and single-solution-based metaheuristic algorithms, which have been commonly applied in the BSP literature, in terms of the solution quality at convergence. Moreover, the proposed solution approach can be considered as competitive in terms of the computational time as well, since the maximum UIMA computational time did not exceed 306 sec over the generated large-size problem instances (with up to 10 berthing positions and 110 vessels calling for service at the MCT). Therefore, the developed UIMA algorithm can be used by the MCT operators as an efficient decision support tool and assist with a cost-effective design of berth schedules within an acceptable computational time. In the meantime, the berth schedules, suggested by UIMA, capture preferences of liner shipping companies as well; since they directly account for the total late departure cost of the arriving vessels at the MCT (see Section 4 of the manuscript). Timely departures of vessels from MCTs are critical for effective liner shipping.

6.5 Managerial insights

This section of the manuscript focuses on the analysis of managerial insights, aiming to determine how changes in the unit cost components of the SCBSP mathematical model (i.e. unit waiting cost, unit handling cost, and unit late departure cost) could impact the berth scheduling decisions. The analysis was conducted for the problem instance P48 (i.e. the problem instance with the largest number of the arriving vessels and the largest number of the berthing positions, which are available for vessel service at the considered MCT). UIMA was used as a solution approach for the SCBSP mathematical model. A total of 10 replications were performed for the considered scenarios with different values of the unit cost components to estimate the average values of the following performance indicators:

- the SCBSP objective function value (i.e. the total vessel service cost);
- the total vessel waiting time;
- the total vessel handling time; and
- the total vessel late departure time.

First, a total of 21 scenarios were developed by increasing the unit waiting cost ranges from US$1,000 and US$5,000 to US$5,000 and US$9,000 with an increment of US$200/period (i.e. the unit waiting cost for scenario "1" was generated as $\varphi_v^{wt1} = U[1000; 5000] \ \forall v \in V$, while the unit waiting cost for scenario "21" was generated as $\varphi_v^{wt21} = U[5000; 9000] \ \forall v \in V$). The values of the remaining parameters of the SCBSP mathematical model (see Table II) were assumed to be unchanged for all the generated unit waiting cost scenarios. The results of the conducted unit waiting cost sensitivity analysis are presented in Figure 13. As the unit handling cost and the unit late departure cost components were constant, and the unit waiting cost increased from scenario "1" to "21", the total vessel service cost increased as well (the total vessel service cost increased by ≈0.55 per cent from scenario "1" to "21"). The numerical experiments show that the total vessel waiting time generally decreased after increasing the unit waiting cost from one scenario to another by assigning the arriving vessels to the first available berthing positions. A reduction in the total vessel waiting time

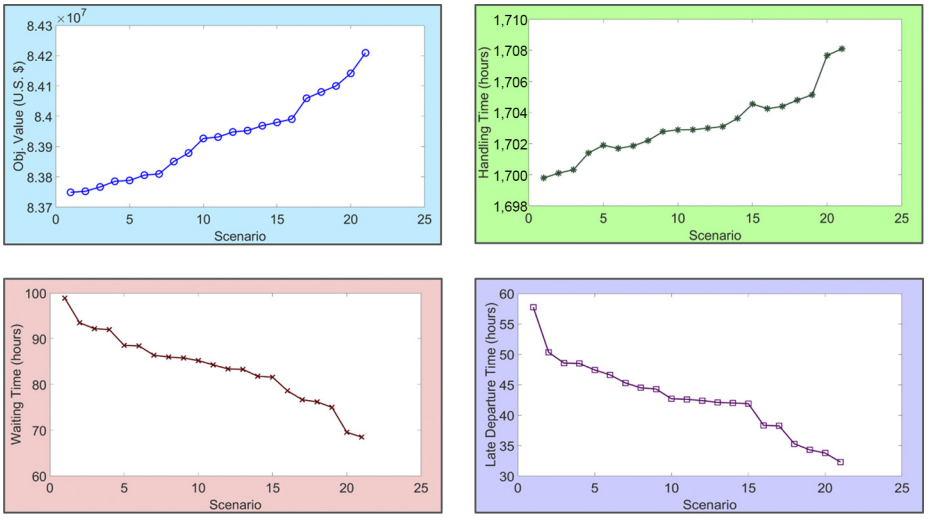

Figure 13. Sensitivity of berth schedules to the unit waiting cost

further decreased the total vessel late departure time. However, the assigned berthing positions were not preferred berthing positions for certain vessels, which caused an increase in the total vessel handling time.

Second, a total of 21 scenarios were developed by increasing the unit handling cost ranges from US$50,000 and US$70,000 to US$70,000 and US$90,000 with an increment of US$1000/period (i.e. the unit handling cost for scenario "1" was generated as $\varphi_v^{ht1} = U[50000; 70000] \ \forall v \in V$, while the unit handling cost for scenario "21" was generated as $\varphi_v^{ht21} = U[70000; 90000] \ \forall v \in V$). The values of the remaining parameters of the SCBSP mathematical model (Table II) were assumed to be unchanged for all the generated unit handling cost scenarios. The results of the conducted unit handling cost sensitivity analysis are presented in Figure 14. As the unit waiting cost and the unit late departure cost components were constant, and the unit handling cost increased from scenario "1" to "21", the total vessel service cost increased as well (the total vessel service cost increased by ≈41.86 per cent from scenario "1" to "21"). The numerical experiments show that the total vessel handling time generally decreased after increasing the unit handling cost from one scenario to another by assigning the arriving vessels to their preferred berthing positions. However, certain vessels had to wait in the designated waiting area of the MCT until their preferred berthing positions became available for service, which further increased the total vessel waiting time and the total vessel late departure time.

Third, a total of 11 scenarios were developed by increasing the unit late departure cost ranges from US$5,000 and US$10,000 to US$10,000 and US$15,000 with an increment of US $500/period (i.e. the unit late departure cost for scenario "1" was generated as $\varphi_v^{lt1} = U[5000; 10000] \ \forall v \in V$, while the unit late departure cost for scenario "11" was generated as $\varphi_v^{lt11} = U[10000; 15000] \ \forall v \in V$). The values of the remaining parameters of the SCBSP mathematical model (Table II) were assumed to be unchanged for all the generated unit late departure cost scenarios. The results of the conducted unit late departure cost sensitivity analysis are presented in Figure 15. As the unit waiting cost and the unit handling cost components were constant, and the unit late departure cost increased from scenario "1" to "11", the total vessel service cost increased as well (the total vessel service

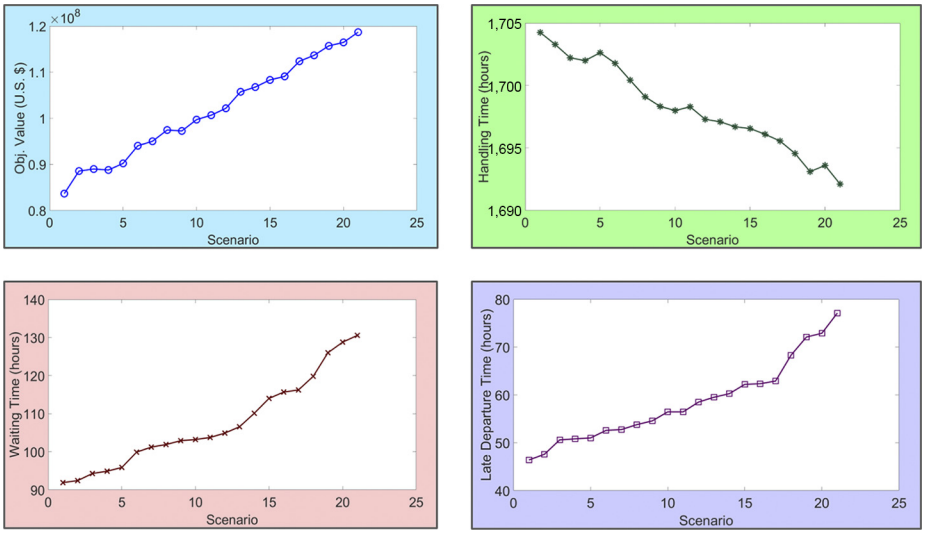

Figure 14.
Sensitivity of berth schedules to the unit handling cost

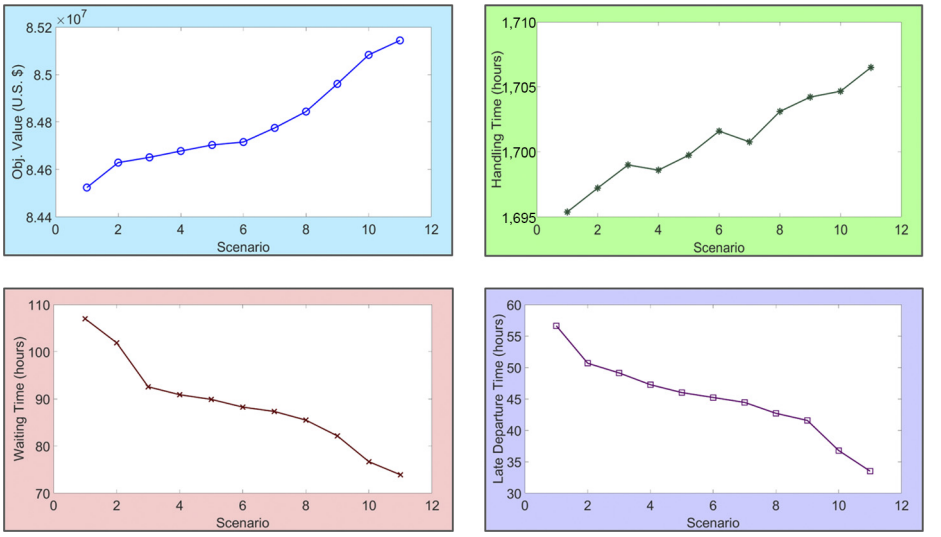

Figure 15.
Sensitivity of berth schedules to the unit late departure cost

cost increased by ≈0.73 per cent from scenario "1" to "11"). The numerical experiments show that the total vessel late departure time generally decreased after increasing the unit late departure cost from one scenario to another by assigning the arriving vessels to the first available berthing positions. Since the arriving vessels were mostly assigned to the first available berthing positions, the total vessel waiting time decreased from one scenario to another as well. However, the assigned berthing positions were not preferred berthing positions for certain vessels, which caused an increase in the total vessel handling time.

7. Conclusions and future research extensions

The MCT operators are looking for effective decision support tools that can assist with berth scheduling to serve the increasing demand for containerized trade and avoid potential vessel service delays. Island-based metaheuristic algorithms have been found to be efficient for various challenging decision problems, but have not been used in the berth scheduling literature. A UIMA was proposed in this study, aiming to solve the spatially constrained BSP. The UIMA population was divided into four sub-populations (i.e. islands). Four different population-based metaheuristics were adopted to search the islands, including the following:

(1) EA;
(2) PSO;
(3) EDA; and
(4) DE.

The adopted population-based metaheuristic algorithms rely on different operators, which facilitate the search process for superior solutions on the UIMA islands.

The conducted numerical experiments demonstrated that the developed UIMA algorithm returned near-optimal solutions for the small-size problem instances. As for the large-size problem instances, UIMA was found to be superior to the EA, PSO, EDA, and DE algorithms, which were executed in isolation, in terms of the obtained objective function values at termination. Furthermore, the developed UIMA algorithm outperformed various single-solution-based metaheuristic algorithms (including Variable Neighborhood Search, Tabu Search, and Simulated Annealing) in terms of the solution quality. The analysis of convergence patterns demonstrated that the developed UIMA algorithm was able to move more effectively along the search space and discover superior solutions as compared to the alternative metaheuristic algorithms. As for the computational time, the maximum UIMA computational time did not exceed 306 sec over the generated large-size problem instances, which can be considered as acceptable from the practical point of view. Therefore, the developed UIMA algorithm can be used by the MCT operators as an efficient decision support tool and assist with a cost-effective design of berth schedules within an acceptable computational time.

This study has a number of potential future research extensions, including:

- development of new approaches for migration;
- consideration of new strategies for immigrant selection and removing individuals on the destination islands;
- development of new strategies for the population initialization and assignment of sub-populations to the islands;
- design of a new solution representation, which is compatible with all the adopted algorithms;
- development of new strategies for handling the infeasible solutions throughout the search process;
- consideration of uncertainties in the MCT operations (e.g., uncertainty in vessel arrival times, uncertainty in vessel handling times, potential internal transport vehicle breakdowns);
- modeling different berthing layout types (i.e., continuous, hybrid, channel); and
- evaluation of the developed UIMA algorithm for the integrated MCT decision problems at the seaside (e.g. berth scheduling and quay crane assignment).

References

Alba, E. and Tomassini, M. (2002), "Parallelism and evolutionary algorithms", *IEEE Transactions on Evolutionary Computation*, Vol. 6 No. 5, pp. 443-462.

Al-Betar, M.A., Awadallah, M.A., Khader, A.T. and Abdalkareem, Z.A. (2015), "Island-based harmony search for optimization problems", *Expert Systems with Applications*, Vol. 42 No. 4, pp. 2026-2035.

Ammi, M. and Chikhi, S. (2015), "A generalized island model based on parallel and cooperating metaheuristics for effective large capacitated vehicle routing problem solving", *Journal of Computing and Information Technology*, Vol. 23 No. 2, pp. 141-155.

Arabani, A.B., Ghomi, S.F. and Zandieh, M. (2011), "Meta-heuristics implementation for scheduling of trucks in a cross-docking system with temporary storage", *Expert Systems with Applications*, Vol. 38 No. 3, pp. 1964-1979.

Bierwirth, C. and Meisel, F. (2015), "A follow-up survey of berth allocation and quay crane scheduling problems in container terminals", *European Journal of Operational Research*, Vol. 244 No. 3, pp. 675-689.

Brester, C., Ryzhikov, I. and Semenkin, E. (2017), "Multi-objective optimization algorithms with the island metaheuristic for effective project management problem solving", *Organizacija*, Vol. 50 No. 4, pp. 364-373.

Carlo, H.J., Vis, I.F. and Roodbergen, K.J. (2015), "Seaside operations in container terminals: literature overview, trends, and research directions", *Flexible Services and Manufacturing Journal*, Vol. 27 Nos 2/3, pp. 224-262.

Cohoon, J.P., Hegde, S.U., Martin, W.N. and Richards, D. (1987), "Punctuated equilibria: a parallel genetic algorithm", in *Genetic Algorithms and their Applications: Proceedings of the Second International Conference on Genetic Algorithms*, MA Institute of Technology, Cambridge, MA.

Cordeau, J., Laporte, G., Legato, P. and Moccia, L. (2005), "Models and tabu search heuristics for the berth allocation problem", *Transportation Science*, Vol. 39 No. 4, pp. 526-538.

Da Silveira, G.J. (2014), "An empirical analysis of manufacturing competitive factors and offshoring", *International Journal of Production Economics*, Vol. 150, pp. 163-173.

Dulebenets, M.A. (2017), "A novel memetic algorithm with a deterministic parameter control for efficient berth scheduling at marine container terminals", *Maritime Business Review*, Vol. 2 No. 4, pp. 302-330.

Dulebenets, M.A. (2019), "A delayed start parallel evolutionary algorithm for just-in-time truck scheduling at a cross-docking facility", *International Journal of Production Economics*, Vol. 212, pp. 236-258.

Dulebenets, M.A., Kavoosi, M., Abioye, O.F. and Pasha, J. (2018), "A self-adaptive evolutionary algorithm for the berth scheduling problem: towards efficient parameter control", *Algorithms*, Vol. 11 No. 7, pp. 1-35.

Dulebenets, M.A., Moses, R., Ozguven, E.E. and Vanli, A. (2017), "Minimizing carbon dioxide emissions due to container handling at marine container terminals via hybrid evolutionary algorithms", *IEEE Access*, Vol. 5, pp. 8131-8147.

Dulebenets, M.A., Pasha, J., Abioye, O.F., Kavoosi, M., Ozguven, E.E., Moses, R., Boot, W.R. and Sando, T. (2019), "Exact and heuristic solution algorithms for efficient emergency evacuation in areas with vulnerable populations", *International Journal of Disaster Risk Reduction*, Vol. 39, pp. 1-18.

Eiben, A.E. and Smith, J.E. (2003), *Introduction to Evolutionary Computing*, Springer-Verlag Berlin Heidelberg.

Emde, S. and Boysen, N. (2016), "Berth allocation in container terminals that service feeder ships and deep-sea vessels", *Journal of the Operational Research Society*, Vol. 67 No. 4, pp. 551-563.

Eniram (2019), "Evolution of shipping", available at: www.eniram.fi/evolution-of-shipping/ (accessed 6 March 2019).

Gong, Y. and Fukunaga, A. (2011), "Distributed island-model genetic algorithms using heterogeneous parameter settings", in *2011 IEEE Congress of Evolutionary Computation (CEC)*, pp. 820-827.

Hansen, P., Oguz, C. and Mladenovic, N. (2008), "Variable neighborhood search for minimum cost berth allocation", *European Journal of Operational Research*, Vol. 191 No. 3, pp. 636-649.

Hsu, H.P. (2016), "A HPSO for solving dynamic and discrete berth allocation problem and dynamic quay crane assignment problem simultaneously", *Swarm and Evolutionary Computation*, Vol. 27, pp. 156-168.

Imai, A., Nishimura, E. and Papadimitriou, S. (2008), "Berthing ships at a multi-user container terminal with a limited quay capacity", *Transportation Research Part E: Logistics and Transportation Review*, Vol. 44 No. 1, pp. 136-151.

Izquierdo, C.E., Melián-Batista, B. and Moreno-Vega, J.M. (2012), "An estimation of distribution algorithm for solving the quay crane scheduling problem with availability constraints", *Advances in Knowledge-Based and Intelligent Information and Engineering Systems*, Vol. 243, pp. 10-19.

Jiao, X., Zheng, F., Liu, M. and Xu, Y. (2018), "Integrated berth allocation and time-variant quay crane scheduling with tidal impact in approach channel", *Discrete Dynamics in Nature and Society*, Vol. 2018, pp. 1-20.

Kavoosi, M., Dulebenets, M.A., Abioye, O.F., Pasha, J., Wang, H. and Chi, H. (2019), "An augmented self-adaptive parameter control in evolutionary computation: a case study for the berth scheduling problem", *Advanced Engineering Informatics*, Vol. 42, pp. 1-25.

Kennedy, J. and Eberhart, R.C. (1997), "A discrete binary version of the particle swarm algorithm", in *1997 IEEE International Conference on Systems, Man, and Cybernetics. Computational Cybernetics and Simulation*, 5, pp. 4104-4108.

Kurdi, M. (2015), "A new hybrid island model genetic algorithm for job shop scheduling problem", *Computers and Industrial Engineering*, Vol. 88, pp. 273-283.

Lardeux, F. and Goëffon, A. (2010), "A dynamic island-based genetic algorithms framework", in *Asia-Pacific Conference on Simulated Evolution and Learning*, Springer, Berlin, Heidelberg.

Larrañaga, P. and Lozano, J.A. (Eds) (2001), *Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation*, Springer Science and Business Media.

Li, M.W., Hong, W.C., Geng, J. and Wang, J. (2017), "Berth and quay crane coordinated scheduling using multi-objective chaos cloud particle swarm optimization algorithm", *Neural Computing and Applications*, Vol. 28 No. 11, pp. 3163-3182.

Lichtblau, D. (2002), "Discrete optimization using mathematica", in *World Multi-Conference on Systemics, Cybernetics and Informatics (SCI 2002)*. International Institute of Informatics and Systemics, 16, pp. 169-174.

Liu, C., Xiang, X., Zhang, C. and Zheng, L. (2016), "A decision model for berth allocation under uncertainty considering service level using an adaptive differential evolution algorithm", *Asia-Pacific Journal of Operational Research*, Vol. 33 No. 6, pp. 1-28.

Magalhaes, T.T., Krempser, E. and Barbosa, H.J. (2015), "Migration policies to improve exploration in parallel island models for optimization via metaheuristics", in *Proceedings of the XXXVII Ibero-Latin American Congress on Computational Methods in Engineering*.

Mauri, G.R., Ribeiro, G.M., Lorena, L.A.N. and Laporte, G. (2016), "An adaptive large neighborhood search for the discrete and continuous berth allocation problem", *Computers and Operations Research*, Vol. 70, pp. 140-154.

Osaba, E., Onieva, E., Carballedo, R., Diaz, F., Perallos, A. and Zhang, X. (2013), "A multi-crossover and adaptive island based population algorithm for solving routing problems", *Journal of Zhejiang University SCIENCE C*, Vol. 14 No. 11, pp. 815-821.

Şahin, C. and Kuvvetli, Y. (2016), "Differential evolution based meta-heuristic algorithm for dynamic continuous berth allocation problem", *Applied Mathematical Modelling*, Vol. 40 Nos 23/24, pp. 10679-10688.

Storn, R. and Price, K. (1997), "Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces", *Journal of Global Optimization*, Vol. 11 No. 4, pp. 341-359.

Ting, C.J., Wu, K.C. and Chou, H. (2014), "Particle swarm optimization algorithm for the berth allocation problem", *Expert Systems with Applications*, Vol. 41 No. 4, pp. 1543-1550.

Tsai, A.H., Lee, C.N., Wu, J.S. and Chang, F.S. (2016), "A novel wharf-based genetic algorithm for berth allocation planning", *Soft Computing*, Vol. 21 No. 11, pp. 2897-2910.

Venturini, G., Iris, Ç., Kontovas, C.A. and Larsen, A. (2017), "The multi-port berth allocation problem with speed optimization and emission considerations", *Transportation Research Part D: Transport and Environment*, Vol. 54, pp. 142-159.

Wang, S., Zheng, J., Zheng, K., Guo, J. and Liu, X. (2012), "Multi resource scheduling problem based on an improved discrete particle swarm optimization", *Physics Procedia*, Vol. 25, pp. 576-582.

Zhen, L., Liang, Z., Zhuge, D., Lee, L.H. and Chew, E.P. (2017), "Daily berth planning in a tidal port with channel flow control", *Transportation Research Part B: Methodological*, Vol. 106, pp. 193-217.

Further reading

Dulebenets, M.A. (2018a), "A diploid evolutionary algorithm for sustainable truck scheduling at a cross-docking facility", *Sustainability*, Vol. 10 No. 5, pp. 1-23.

Dulebenets, M.A. (2018b), "A comprehensive multi-objective optimization model for the vessel scheduling problem in liner shipping", *International Journal of Production Economics*, Vol. 196, pp. 293-318.

Dulebenets, M.A. (2018c), "Green vessel scheduling in liner shipping: modeling carbon dioxide emission costs in sea and at ports of call", *International Journal of Transportation Science and Technology*, Vol. 7 No. 1, pp. 26-44.

Dulebenets, M.A. (2018d), "A comprehensive evaluation of weak and strong mutation mechanisms in evolutionary algorithms for truck scheduling at cross-docking terminals", *IEEE Access*, Vol. 6, pp. 65635-65650.

Dulebenets, M.A. (2018e), "The green vessel scheduling problem with transit time requirements in a liner shipping route with emission control areas", *Alexandria Engineering Journal*, Vol. 57 No. 1, pp. 331-342.

Dulebenets, M.A. (2018f), "The vessel scheduling problem in a liner shipping route with heterogeneous fleet", *International Journal of Civil Engineering*, Vol. 16 No. 1, pp. 19-32.

Corresponding author

Maxim A. Dulebenets can be contacted at: mdulebenets@eng.famu.fsu.edu

Appendix

Table AI. The Objective values and computational times by CPLEX, UIMA, and the candidate population-based algorithms for the small-size problem instances

Problem instance	No. of vessels	No. of berths	CPLEX		UIMA		EA		PSO		EDA		DE	
			Objective value, 10^7 (US$)	CPU time, (sec)	Objective value, 10^7 (US$)	CPU time, (sec)	Objective value, 10^7 (US$)	CPU time, (sec)	Objective value, 10^7 (US$)	CPU time, (sec)	Objective value, 10^7 (US$)	CPU time, (sec)	Objective value, 10^7 (US$)	CPU time, (sec)
P1	5	2	0.49	0.56	0.49	27.52	0.49	8.38	0.49	9.73	0.49	11.61	0.49	22.28
P2	5	3	0.46	0.90	0.46	27.45	0.46	8.36	0.46	9.70	0.46	11.58	0.46	22.38
P3	5	4	0.44	1.59	0.44	27.51	0.44	8.39	0.44	9.79	0.44	11.69	0.44	22.35
P4	7	2	0.80	3.31	0.80	31.12	0.80	9.51	0.80	11.07	0.80	13.02	0.80	23.68
P5	7	3	0.73	5.21	0.73	31.05	0.73	10.22	0.73	11.05	0.73	13.11	0.73	23.81
P6	7	4	0.70	5.69	0.70	31.61	0.70	9.98	0.70	11.08	0.70	13.18	0.70	23.83
P7	9	2	1.02	25.36	1.02	34.79	1.02	10.84	1.02	12.45	1.02	14.57	1.02	25.20
P8	9	3	0.93	50.74	0.93	34.95	0.93	11.04	0.93	12.44	0.93	14.80	0.93	25.22
P9	9	4	0.87	20.03	0.87	35.38	0.87	10.87	0.87	12.50	0.87	15.00	0.87	25.39
P10	11	2	1.22	79.46	1.22	38.10	1.22	12.02	1.24	13.73	1.24	16.18	1.24	26.75
P11	11	3	1.15	88.67	1.15	38.19	1.15	12.29	1.16	13.80	1.18	16.56	1.18	26.78
P12	11	4	1.06	59.65	1.06	38.66	1.08	12.65	1.09	13.83	1.09	16.99	1.07	26.77
P13	13	2	1.42	391.33	1.42	42.18	1.47	13.38	1.49	15.06	1.47	17.82	1.47	28.06
P14	13	3	1.36	679.36	1.39	42.09	1.43	13.34	1.44	15.10	1.44	17.82	1.42	28.25
P15	13	4	1.28	704.26	1.29	42.18	1.31	13.30	1.31	15.28	1.32	18.13	1.30	28.43
P16	15	2	1.56	326.48	1.59	45.37	1.62	14.43	1.64	16.49	1.63	19.15	1.62	29.63
P17	15	3	1.50	1393.49	1.53	45.47	1.56	14.56	1.56	16.47	1.58	19.52	1.58	30.34
P18	15	4	1.47	2348.76	1.50	45.78	1.53	14.57	1.54	16.68	1.55	19.85	1.55	30.25
P19	17	2	1.77	1461.42	1.81	49.20	1.85	15.92	1.87	17.87	1.88	20.83	1.87	31.52
P20	17	3	1.85	7237.00	1.76	49.41	1.82	15.86	1.84	17.91	1.89	21.20	1.88	31.31
P21	17	4	1.78	7225.15	1.71	49.58	1.73	15.91	1.75	18.12	1.76	21.68	1.75	31.36
P22	20	3	2.31	7277.19	2.25	54.62	2.29	17.91	2.31	19.93	2.35	24.53	2.36	34.26
P23	20	4	2.26	7303.72	2.05	54.85	2.07	17.62	2.09	20.10	2.07	24.66	2.06	33.85
P24	20	5	1.97	7328.10	1.88	55.14	1.91	17.76	1.93	20.10	1.92	25.81	1.92	33.92
Average			*1.27*	*1834.06*	*1.25*	*40.51*	*1.27*	*12.88*	*1.28*	*14.59*	*1.28*	*17.47*	*1.28*	*27.73*

Table AII.
The objective values and computational times by CPLEX, UIMA, and the candidate single-solution-based algorithms for the small-size problem instances

			CPLEX		UIMA		VNS		TS		SA	
Problem instance	No. of vessels	No. of berths	Objective value, 10^7 (US$)	CPU time, (sec)	Objective value, 10^7 (US$)	CPU time, (sec)	Objective value, 10^7 (US$)	CPU time, (sec)	Objective value, 10^7 (US$)	CPU time, (sec)	Objective value, 10^7 (US$)	CPU time, (sec)
P1	5	2	0.49	0.56	0.49	27.52	0.49	1.87	0.49	2.76	0.49	1.83
P2	5	3	0.46	0.90	0.46	27.45	0.46	1.80	0.46	2.70	0.46	1.82
P3	5	4	0.44	1.59	0.44	27.51	0.44	1.83	0.44	2.74	0.44	1.84
P4	7	2	0.80	3.31	0.80	31.12	0.80	2.26	0.80	2.92	0.80	2.01
P5	7	3	0.73	5.21	0.73	31.05	0.73	2.26	0.73	2.92	0.73	2.04
P6	7	4	0.70	5.69	0.70	31.61	0.70	2.10	0.70	2.98	0.70	2.06
P7	9	2	1.02	25.36	1.02	34.79	1.02	2.41	1.02	3.19	1.02	2.30
P8	9	3	0.93	50.74	0.93	34.95	0.95	2.42	0.94	3.20	0.96	2.73
P9	9	4	0.87	20.03	0.87	35.38	0.88	2.30	0.89	3.23	0.88	2.74
P10	11	2	1.22	79.46	1.22	38.10	1.25	2.64	1.26	3.40	1.26	2.52
P11	11	3	1.15	88.67	1.15	38.19	1.18	2.64	1.19	3.42	1.18	2.49
P12	11	4	1.06	59.65	1.06	38.66	1.09	2.54	1.09	3.47	1.09	2.52
P13	13	2	1.42	391.33	1.42	42.18	1.49	2.71	1.48	3.62	1.49	2.72
P14	13	3	1.36	679.36	1.39	42.09	1.43	2.72	1.43	3.65	1.43	2.74
P15	13	4	1.28	704.26	1.29	42.18	1.31	2.76	1.33	3.69	1.33	2.75
P16	15	2	1.56	326.48	1.59	45.37	1.63	2.95	1.64	3.89	1.64	2.94
P17	15	3	1.50	1393.49	1.53	45.47	1.59	2.96	1.59	3.89	1.59	2.98
P18	15	4	1.47	2348.76	1.50	45.78	1.55	3.00	1.55	3.94	1.55	2.99
P19	17	2	1.77	1461.42	1.81	49.20	1.88	3.26	1.87	4.11	1.87	3.17
P20	17	3	1.85	7237.00	1.76	49.41	1.93	3.20	1.91	4.10	1.93	3.25
P21	17	4	1.78	7225.15	1.71	49.58	1.77	3.23	1.78	4.14	1.78	3.21
P22	20	3	2.31	7277.19	2.25	54.62	2.38	3.60	2.37	4.45	2.39	3.51
P23	20	4	2.26	7303.72	2.05	54.85	2.12	3.62	2.11	4.75	2.13	3.55
P24	20	5	1.97	7328.10	1.88	55.14	1.95	3.61	1.96	4.52	1.94	3.57
Average			*1.27*	*1834.06*	*1.25*	*40.51*	*1.29*	*2.69*	*1.29*	*3.57*	*1.30*	*2.68*

Table AIII.
The objective values and computational times by UIMA and the candidate population-based algorithms for the large-size problem instances

Problem instance	No. of Vessels	No. of Berths	UIMA		EA		PSO		EDA		DE	
			Objective value, 10^7 (US$)	CPU Time, (sec.)	Objective value, 10^7 (US$)	CPU time, (sec.)	Objective value, 10^7 (US$)	CPU time, (sec.)	Objective value, 10^7 (US$)	CPU time, (sec.)	Objective value, 10^7 (US$)	CPU time, (sec.)
P25	65	4	7.39	151.68	8.33	123.17	8.39	55.98	8.41	79.00	8.32	71.64
P26	65	6	5.57	160.04	6.22	123.38	6.27	55.46	6.32	88.11	6.24	71.81
P27	65	8	4.87	163.99	5.42	123.31	5.43	55.22	5.50	97.92	5.46	72.07
P28	70	4	8.05	164.22	8.98	131.78	9.14	59.07	9.16	85.82	9.14	77.58
P29	70	6	5.91	170.04	6.57	131.54	6.66	59.51	6.75	97.22	6.67	75.60
P30	70	8	5.22	177.28	5.77	132.24	5.86	59.10	5.87	107.28	5.85	74.55
P31	75	4	8.77	175.59	9.81	139.84	9.91	63.23	10.03	92.52	9.95	78.12
P32	75	6	6.36	181.77	7.06	139.90	7.19	62.64	7.23	105.06	7.17	78.22
P33	75	8	5.56	189.88	6.16	139.59	6.18	62.41	6.27	117.87	6.21	78.20
P34	80	4	9.55	187.26	10.76	148.21	10.86	66.80	10.94	100.15	10.83	82.51
P35	80	6	6.77	194.70	7.55	148.32	7.64	66.59	7.68	114.29	7.64	82.37
P36	80	8	6.02	205.32	6.71	148.53	6.75	66.96	6.79	128.39	6.73	82.78
P37	85	4	10.76	199.02	11.90	156.61	12.02	70.28	12.13	107.48	12.01	86.15
P38	85	6	7.51	206.94	8.29	156.10	8.38	70.27	8.41	123.45	8.34	86.30
P39	85	8	6.53	216.30	7.25	157.00	7.30	70.42	7.30	139.56	7.28	86.63
P40	90	6	8.10	220.08	8.93	164.71	9.00	74.17	9.14	134.16	8.97	90.62
P41	90	8	6.91	232.18	7.63	165.25	7.70	75.44	7.76	151.47	7.68	90.55
P42	90	10	6.77	242.47	7.48	164.83	7.55	76.22	7.60	170.08	7.53	90.74
P43	100	6	9.15	246.24	10.14	181.36	10.23	83.00	10.28	154.07	10.19	98.63
P44	100	8	7.84	260.20	8.68	182.13	8.71	82.63	8.81	176.20	8.66	98.84
P45	100	10	7.69	271.99	8.47	181.30	8.54	83.13	8.61	198.57	8.49	98.76
P46	110	6	10.37	271.68	11.46	198.21	11.50	89.29	11.64	175.78	11.53	106.36
P47	110	8	8.53	289.33	9.43	198.92	9.53	89.37	9.59	201.80	9.45	106.51
P48	110	10	8.37	305.14	9.25	198.22	9.33	89.86	9.40	228.01	9.24	106.59
Average			*7.44*	*211.81*	*8.26*	*155.60*	*8.34*	*70.29*	*8.40*	*132.26*	*8.32*	*86.34*

Table AIV.
The objective values and computational times by UIMA and the candidate single-solution-based algorithms for the large-size problem instances

Problem instance	No. of vessels	No. of berths	UIMA		VNS		TS		SA	
			Objective value, 10^7 (US$)	CPU time, (sec.)	Objective value, 10^7 (US$)	CPU time, (sec.)	Objective value, 10^7 (US$)	CPU time, (sec.)	Objective value, 10^7 (US$)	CPU time, (sec.)
P25	65	4	7.39	151.68	8.96	9.18	9.00	10.28	8.92	9.46
P26	65	6	5.57	160.04	6.64	9.31	6.73	10.25	6.68	9.40
P27	65	8	4.87	163.99	5.78	9.23	5.87	10.16	5.79	9.42
P28	70	4	8.05	164.22	9.77	9.77	9.84	10.79	9.81	9.97
P29	70	6	5.91	170.04	7.05	9.85	7.15	10.79	7.06	9.94
P30	70	8	5.22	177.28	6.22	9.86	6.29	10.85	6.20	10.20
P31	75	4	8.77	175.59	10.62	10.37	10.73	11.39	10.62	10.72
P32	75	6	6.36	181.77	7.61	10.38	7.68	11.38	7.62	10.66
P33	75	8	5.56	189.88	6.59	10.44	6.66	11.46	6.58	10.82
P34	80	4	9.55	187.26	11.58	11.04	11.68	12.08	11.60	11.30
P35	80	6	6.77	194.70	8.12	11.04	8.23	12.09	8.13	11.39
P36	80	8	6.02	205.32	7.13	11.07	7.26	12.13	7.16	11.34
P37	85	4	10.76	199.02	12.70	11.74	12.86	13.37	12.78	11.92
P38	85	6	7.51	206.94	8.88	11.76	9.00	13.35	8.86	11.93
P39	85	8	6.53	216.30	7.74	11.80	7.77	13.42	7.74	11.94
P40	90	6	8.10	220.08	9.61	12.28	9.72	13.33	9.60	12.48
P41	90	8	6.91	232.18	8.18	12.36	8.22	13.43	8.15	12.48
P42	90	10	6.77	242.47	7.96	12.35	8.11	13.37	7.96	12.47
P43	100	6	9.15	246.24	10.84	13.52	10.90	14.75	10.89	13.79
P44	100	8	7.84	260.20	9.29	13.58	9.32	14.72	9.23	13.83
P45	100	10	7.69	271.99	9.03	13.56	9.12	14.63	9.09	13.81
P46	110	6	10.37	271.68	12.25	14.71	12.39	15.87	12.24	15.02
P47	110	8	8.53	289.33	10.10	14.81	10.19	15.96	10.15	15.04
P48	110	10	8.37	305.14	9.88	14.79	9.97	15.97	9.87	15.02
Average			*7.44*	*211.81*	*8.86*	*11.62*	*8.94*	*12.74*	*8.86*	*11.85*