



An Adaptive Island Evolutionary Algorithm for the berth scheduling problem

Maxim A. Dulebenets¹

Received: 19 July 2018 / Accepted: 30 July 2019 / Published online: 8 August 2019
© Springer-Verlag GmbH Germany, part of Springer Nature 2019

Abstract

Increasing volumes of the seaborne containerized trade put additional pressure on marine container terminal operators. Long congestion periods have been reported at certain marine container terminals due to inability of the infrastructure to serve the growing demand, increasing number of megaships, port disruptions, and other factors. In order to alleviate congestion and avoid potential cargo delivery delays to the end customers, marine container terminal operators have to enhance the efficiency of their operations. This study focuses on improving the seaside operations at marine container terminals. A new Adaptive Island Evolutionary Algorithm is proposed for the berth scheduling problem, aiming to minimize the total weighted service cost of vessels. The developed algorithm simultaneously executes separate Evolutionary Algorithms in parallel on its islands and exchanges individuals between the islands based on an adaptive mechanism, which allows more efficient exploration of the problem search space. A set of extensive computational experiments indicate that the optimality gaps of the Adaptive Island Evolutionary Algorithm do not exceed 1.93% for the considered small-size problem instances. Furthermore, the proposed solution algorithm was compared against the other state-of-the-art metaheuristic algorithms and exhibited statistically significant improvements in terms of the objective function values.

Keywords Maritime transportation · Berth scheduling problem · Optimization · Parallel evolutionary algorithms · Island model · Adaptive mechanism · Migration

1 Introduction

Maritime transportation plays a crucial role for the international trade and the global economy [1]. According to the recent review of maritime transport, conducted by the United Nations Conference on Trade and Development (UNCTAD) in 2017, the containerized seaborne cargo volumes increased by 34.4%, whereas dry cargo, major bulk cargo, and oil/gas volumes increased by 15.7%, 35.9%, and 10.2%, respectively, between 2010 and 2016 [1]. The overall international seaborne trade volumes reached 10.3 billion tons in 2016, which is a 22.3% increase as compared to 2010. A substantial portion of general consumption cargoes is transported by oceangoing vessels in a containerized form. The world fleet of container vessels increased from 244,339 vessels

in 2016 to 245,609 in 2017 [1]. Liner shipping companies continue increasing size of their vessels, as larger vessels (a.k.a., megaships) yield the route service cost savings due to their economies of scale. In 2017, Orient Overseas Container Line (OOCL) launched five vessels with the total capacity of 21,413 twenty-foot equivalent units (TEUs). On the other hand, China Ocean Shipping Company (COSCO) deployed a container vessel “Shipping Universe” in 2018, which is able to carry 21,237 TEUs. COSCO expects to launch another five vessels of the same type in 2019. According to the Journal of Commerce [2], the overall number of large-size vessels is forecasted to grow at least by 13.0% by 2020.

Marine container terminal (MCT) operators are responsible for handling containers that are delivered by vessels. A drastic increase in the vessel size imposes additional pressure on MCT operators. In order to serve the arriving megaships, MCT operators have to deepen the access channels to make sure that the megaships will be able to safely navigate, strengthen berthing positions, and purchase more advanced handling equipment. Many MCT operators were not able to successfully implement the latter upgrades, which resulted

✉ Maxim A. Dulebenets
mdulebenets@eng.famu.fsu.edu

¹ Department of Civil and Environmental Engineering, Florida A&M University-Florida State University, 2525 Pottsdamer Street, Building A, Suite A124, Tallahassee, FL 32310-6046, USA

in severe port congestion and disruptions in the MCT operations. Along with increasing number of megaships, there are several other factors, which may cause congestion at MCTs, including the following: (1) formation of liner shipping alliances; (2) lack of innovative technologies; (3) shortage of chassis; (4) adverse weather conditions; (5) industrial actions (e.g., strikes at the United States West Coast ports); (6) labor shortages; (7) unstable demand (i.e., frequent increases and decreases in demand); (8) customs shutdowns; and others.

In order to alleviate the existing congestion issues at MCTs and avoid potential delays in the transport of cargoes to the end customers, MCT operators should improve the efficiency of operations and develop innovative strategies and alternative approaches for scheduling their container handling resources. The MCT operations are generally classified into three groups [3, 4]: (1) seaside operations; (2) marshaling yard operations; and (3) landside operations. As for the seaside operations, containers are unloaded by on-shore quay cranes from vessels and are further placed on internal transport vehicles (which include straddle carriers, yard trucks, automated guided vehicles, automated lifting vehicles, etc.). Internal transport vehicles are responsible for transferring containers between the seaside and the marshaling yard. As for the marshaling yard operations, the delivered containers are handled by gantry cranes (a.k.a., yard cranes), which place those containers in the yard blocks and perform necessary reshuffling operations. As for the landside operations, the assigned containers are either delivered or picked up by drayage trucks, which enter the MCT via the dedicated gate and wait in a specific area.

All the MCT operations received a lot of attention from the researchers and practitioners. This study primarily focuses on the MCT seaside operations and addresses the berth scheduling problem (BSP). The BSP aims to assign the arriving vessels to the available MCT berths and determine the order of vessels that will be served at every berth. The berth scheduling affects the total turnaround time of vessels and the overall MCT performance. Inefficient berth schedules may lead to delays in vessel service, unbalanced workload among the handling equipment units, increasing MCT congestion, which will further result in product delivery delays to the end customers. To assist MCT operators with the design of efficient berth schedules, a novel Adaptive Island Evolutionary Algorithm is proposed in this study, which simultaneously executes separate Evolutionary Algorithms in parallel on its islands and exchanges individuals between the islands based on a certain adaptive mechanism. Specifically, the exchange of individuals occurs if no improvements in the objective function values over all the islands have been observed after a pre-determined number generations. A set of computational experiments are conducted to estimate the optimality gaps of the developed solution algorithm and compare the algorithm

against the other state-of-the-art metaheuristic algorithms, which have been widely used in the BSP literature.

The remaining sections of the manuscript are organized in the following manner. The second section of the manuscript presents a comprehensive overview of the previously conducted BSP studies with a primary focus on the developed solution algorithms. The third section provides a detailed description of the problem, considered in this study. The fourth section describes a mixed-integer linear mathematical model, which was formulated for the problem. The fifth section focuses on the key features of the developed solution algorithm, while the sixth section discusses the extensive computational experiments, which were performed to assess the efficiency of the proposed solution algorithm against the other state-of-the-art metaheuristic algorithms. The last section provides a summary of findings for this study and outlines potential avenues for future research.

2 Literature review

The BSP received a lot of attention from the research community over the last decade. The BSP can be reduced to the machine scheduling problem, and the latter class of problems has *NP*-hard complexity [3]. A number of heuristic and metaheuristic algorithms have been developed in the past to solve the BSP within an acceptable computational time. For a comprehensive review of the berth scheduling literature and seaside operations at MCTs this study refers to Bierwirth and Meisel [3]. The literature review in this study primarily focuses on features of the solution algorithms, proposed by the previously conducted BSP studies.

A significant amount of BSP studies relied on the Evolutionary Algorithms (EAs). Golias et al. [5] studied the BSP, where vessel arrival and handling times were not known with certainty. The authors presented a bi-level formulation, where the average total vessel service time and the total range of vessel service times were minimized at the upper and lower levels, respectively. An EA-based heuristic was applied to solve the problem. A set of numerical experiments demonstrated that the proposed berth scheduling policy was more efficient under the scenarios with high congestion as compared to the first-come-first-served policies. Frojan et al. [6] proposed a mathematical formulation for the BSP at the MCT with multiple quays. The objective of the model minimized the total service cost of vessels. The authors implemented an EA, which relied on a local search heuristic, to solve the problem. The extensive computational experiments indicated that the proposed solution approach showcased a competitive performance against the benchmark values, which were provided by the recent BSP studies.

Dulebenets [7, 8] and Dulebenets et al. [9] focused on evaluation of different parameter control strategies within

the EA-based algorithms, which were developed to solve various BSP mathematical formulations. More specifically, deterministic, adaptive, and self-adaptive parameter control strategies were considered. According to the deterministic parameter control strategy, the EA parameters should be updated based on a particular counter (e.g., target number of fitness function evaluations, target number of generations). On the other hand, the adaptive parameter control strategy updates the EA parameters using the feedback from the search (e.g., were there any changes in the fitness function values after a certain number of generations). As for the self-adaptive parameter control strategy, the EA parameters are encoded in the chromosomes and are being updated throughout the EA evolution. It was found that the EA-based algorithms with the aforementioned parameter control strategies were more promising as compared to the alternative EAs, which solely relied on the parameter tuning strategy [7–9].

Dulebenets et al. [10] proposed a set of Hybrid EAs for the green BSP, which accounted for the cost of carbon dioxide emissions, produced due to container handling. The Hybrid EAs deployed the additional problem-specific local search heuristics along with stochastic search operators. A set of the numerical experiments were conducted to evaluate the developed solution approach. It was found that deployment of the additional problem-specific local search heuristics was favorable for the search process. Furthermore, the berth schedules were significantly influenced with the unit cost of carbon dioxide emissions. Several BSP studies applied the EA-based algorithms for evaluation of different berth scheduling policies, where vessels could be diverted from one MCT to another [11, 12]. The considered berth scheduling policies were based on the shared seaside capacity of different MCTs. Such berth scheduling policies were found to be efficient during the peak hours at busy MCTs.

A number of the EA algorithms were developed for the multi-objective BSPs, which capture conflicting objectives of the MCT operator throughout berth scheduling. For example, Karafa et al. [13] presented an EA-based heuristic for a bi-objective BSP, considering stochastic nature of vessel handling times. The first objective function aimed to minimize the expected total service time of the arriving vessels. The second objective function aimed to minimize the service start and finish time risk for the arriving vessels. The numerical experiments demonstrated that the proposed methodology provided superior solutions as compared to the solutions that were based on the expected vessel handling times. Hu [14] proposed a bi-objective BSP mathematical model, where the first objective minimized the total delayed workload, while the second one minimized the total night workload. The problem was solved using the EA, which was based on the well-known Non-Dominated Sorting Genetic Algorithm II (NSGA-II). The computational experiments showcased how the developed methodology could be used for analysis of

tradeoffs between the total delayed workload and the total night workload. The importance of setting the appropriate algorithmic parameters was highlighted as well.

Several studies applied the Simulating Annealing (SA)-based algorithms to solve BSPs. Xu et al. [15] proposed a mathematical formulation for the BSP, capturing uncertainty in vessel arrival and handling times. The objective aimed to minimize the total late departures of vessels and maximized the buffer time length. The authors developed an algorithm, which relied on the features of a SA and a Branch-and-Bound algorithm, to solve the problem. The numerical experiments showed that the proposed solution approach would be effective in practice to develop robust berth schedules. Emde and Boysen [16] proposed a SA-based algorithm to solve the BSP, which aimed to minimize the total vessel waiting time and the total number of containers delayed. The considered MCT served two types of vessels, including feeder vessels and deep-sea vessels. The feeder vessels were assumed to have low capacity and were used primarily for short-distance shipping. On the other hand, the deep-sea vessels were assumed to have larger capacity and were used primarily for long-distance shipping. The results from computational experiments indicated that the developed solution approach was effective, especially for the realistic-size problem instances.

Some of the BSP studies relied on a Greedy Randomized Adaptive Search Procedure (GRASP). Lee et al. [17] presented two versions of a GRASP for solving the BSP, one of which was based on the “*first-come-first-pack*” rule. The objective function minimized the total weighted vessel turnaround time. The developed solution algorithm was compared against CPLEX and the Stochastic Beam Search (SBS) algorithm for the small-size and realistic-size problem instances. It was found that the CPLEX computational time was increasing exponentially with the problem size. Moreover, the first GRASP version was recommended for the realistic-size problem instances, while the second GRASP version was efficient for the small-size problem instances.

A few BSP papers implemented the Tabu Search (TS)-based algorithms. For example, Cordeau et al. [18] developed a TS heuristic to solve the BSP, aiming to minimize the total weighted service time of vessels. The numerical experiments demonstrated that only small-size problem instances could be solved to the global optimality using CPLEX. On the other hand, the TS heuristic was able to tackle the small-size and realistic-size problem instances. Lalla-Ruiz et al. [19] presented a solution algorithm for the BSP, which was based on a TS and a path relinking heuristic. The objective of the model minimized the total vessel service time. Similar to the findings from Cordeau et al. [18], CPLEX was able to solve the small-size problem instances only. As for the realistic-size problem instances, the proposed solution algorithm outperformed a typical TS algorithm in terms of the solution quality

and was found to be competitive in terms of the computational time. A variety of the other heuristic and metaheuristic algorithms have been presented in the BSP literature, including the following: Subgradient heuristic [20], Ant Colony Optimization [21], Variable Neighborhood Search heuristic [22], Adaptive Large Neighborhood Search heuristic [23], Particle Swarm Optimization [23], Clustering Search heuristic [23], and others.

A comprehensive review of the collected BSP studies revealed a large variety of solution algorithms. However, as highlighted in the review of the BSP literature performed by Bierwirth and Meisel [3], the majority of the BSP studies still apply canonical EAs and other stochastic search optimization algorithms that solely rely on the stochastic search without taking into consideration the feedback from the search and/or applying any local search. To address the latter shortcoming and improve the efficiency of berth schedules at MCTs, this study presents a new Adaptive Island EA, which simultaneously executes separate EAs in parallel on its islands and, based on the feedback from the search, exchanges individuals between those islands. Application of parallel Evolutionary Algorithms is expected to improve the quality of berth schedules and assist MCT operators with efficient decision making. The contributions of this study to the state-of-the-art can be summarized as follows:

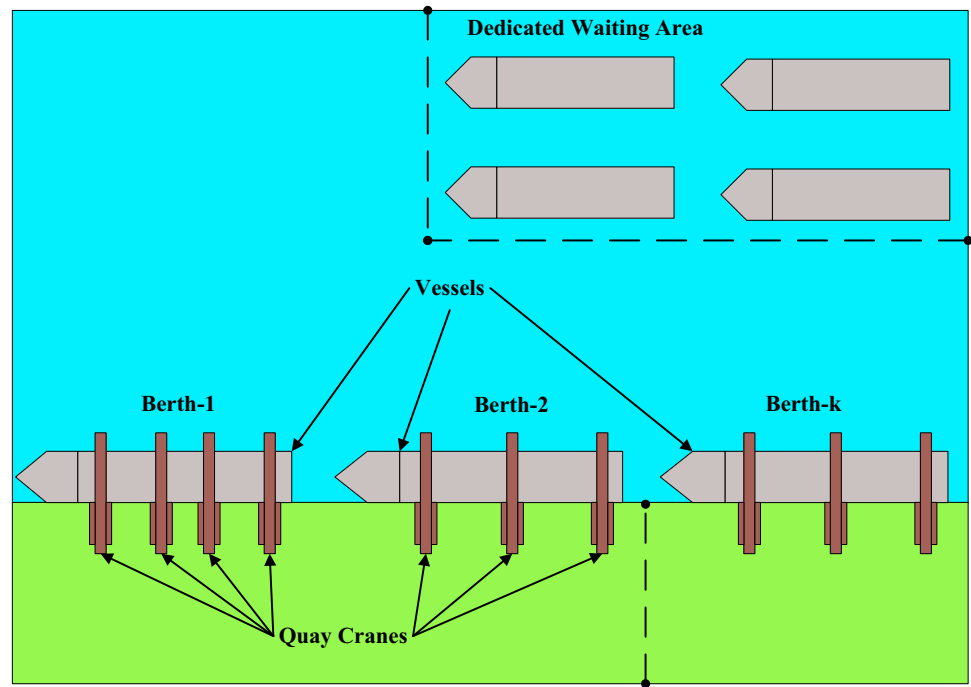
- (1) This study contributes to the existing BSP literature and proposes a new Adaptive Island EA, which simultaneously executes separate EAs in parallel on its islands and exchanges individuals between the islands. On the other hand, the published-to-date BSP studies rely on typical EAs that do not apply the concept of multiple islands;
- (2) This study contributes to the existing EA literature and proposes an adaptive mechanism within the developed Adaptive Island EA for the exchange of individuals between the islands. The exchange of individuals occurs if no improvements in the objective function values over all the islands have been observed after a pre-determined number of generations (i.e., the feedback from the search is explicitly considered). On the other hand, typical Island EAs, proposed in the EA literature, use deterministic migration criteria (e.g., number of fitness function evaluations, number of generations, computational time) and ignore the feedback from the search;
- (3) This study compares the proposed Adaptive Island EA against the other state-of-the-art metaheuristic algorithms, which have been widely used in the BSP literature. The candidate algorithms are evaluated in terms of the optimality gap values, objective function values at termination, computational time values, and convergence patterns.

3 Problem description

This study will focus on the BSP at a multi-user MCT. Vessels that are calling for service at the MCT belong to various liner shipping companies. The MCT is assumed to have a discrete berthing layout. According to the latter terminal configuration, only one vessel can be served by the MCT operator at each berth at a time. Note that the discrete berthing layout has been used in many BSP studies conducted to date [3]. Let $I = \{1, \dots, n\}$ be a set of vessels calling for service at the MCT, and $J = \{1, \dots, k\}$ be a set of the MCT berths. Liner shipping companies provide the information regarding the expected arrival times of vessels to the MCT operator (i.e., the dynamic vessel arrival scenario). Modeling uncertainty in the vessel arrival times due to different factors (e.g., severe weather conditions, changes in the vessel schedules, delays in vessel service at preceding ports, etc.) is not included in the scope of this study. Each vessel is towed by push boats to the assigned berth upon its arrival at the MCT. If the assigned berth is occupied by another vessel, the given vessel will be towed to the specific waiting area (see Fig. 1), which is allocated close to the MCT. The MCT operator is assumed to incur an additional vessel waiting cost c_i^{WT} , $i \in I$ (which is measured in USD per hour), as a substantial amount of waiting vessels may increase congestion at the seaside of the given MCT, which will further cause navigational difficulties for the vessels entering or leaving the MCT.

The liner shipping companies negotiate the container handling productivity hp_{ij} , $i \in I$, $j \in J$ (which is measured in TEUs per hour) with the MCT operator. Based on the expected vessel arrival times and the requested handling productivity, the MCT operator must develop a preliminary berth schedule, assign the available handling equipment accordingly (i.e., quay cranes for handling containers at the vessel, internal transport vehicles for the transfer of containers between the marshaling yard and the seaside of the MCT, gantry cranes for serving internal transport vehicles in the marshaling yard), and allocate the storage areas in the marshaling yard. Depending on the handling productivity requested, the liner shipping company is anticipated to pay a vessel handling cost c_i^{HT} , $i \in I$ (which is measured in USD per hour). The handling time of a vessel HT_{ij} , $i \in I$, $j \in J$ (which is measured in hours) is calculated based on the amount of containers to be handled NC_i , $i \in I$ (which is measured in TEUs) and the negotiated container handling productivity. Note that the container handling productivity will decrease if a vessel is diverted from the initially scheduled berth (i.e., “preferred berth”) to another MCT berth, which will further increase the handling time of the vessel. The decrease in the handling productivity can be justified by the fact that the travel distance from a given MCT berth to the assigned storage area of the marshaling yard is typically larger than the distance to be traveled from the “preferred

Fig. 1 Service of vessels at the MCT



berth” [7–10]. Uncertainty in the handling time of vessels due to various factors (e.g., handling equipment breakdowns, collision of internal transport vehicles, adverse weather, etc.) is not modeled in this study.

Liner shipping companies are responsible for developing their own vessel schedules, where ports of call must be visited with a certain frequency. Each liner shipping company designs its vessel schedules based on the expected vessel service completion time at ports of call. It is assumed that each liner shipping company negotiates the requested vessel departure time RD_i , $i \in I$ (which is measured in hours) with the MCT operator for each vessel. If the MCT operator violates the requested vessel departure time for a given vessel, a vessel late departure penalty c_i^{LD} , $i \in I$ (which is measured in USD per hour) will be imposed. Moreover, the MCT operator must account for the vessel priority in berth scheduling (e.g., the vessels, belonging to preferential liner shipping companies with higher volumes of containers to be handled over a given planning time horizon, must have higher priorities as opposed to the vessels, belonging to the other liner shipping companies with lower volumes of containers to be handled). This study adopts a methodology, where the vessel priority is captured by assigning a specific weight (w_i , $i \in I$) for each vessel [17, 18]. The MCT operator aims to design a cost-efficient berth schedule that will allow minimizing the total weighted service cost of vessels, which is composed of the following three components: (1) the total weighted handling cost of vessels; (2) the total weighted waiting cost of vessels; and (3) the total weighted late departure penalty of vessels.

4 Model formulation

This section presents the nomenclature, which will be further adopted in the manuscript, and a mixed-integer mathematical model for the discrete dynamic berth scheduling problem with vessel service priority.

4.1 Nomenclature

4.1.1 Sets

$I = \{1, \dots, n\}$ Set of vessels calling for service at the MCT
 $J = \{1, \dots, k\}$ Set of the MCT berths

4.1.2 Decision variables

x_{ij} , $i \in I$, $j \in J$ = 1 if vessel i is served at berth j (= 0 otherwise)
 y_{ii^*} , $i, i^* \in I$, $i \neq i^*$ = 1 if vessel i^* is served at the same berth immediately after vessel i (= 0 otherwise)
 f_i , $i \in I$ = 1 if vessel i is served as the first vessel at the assigned berth (= 0 otherwise)
 l_i , $i \in I$ = 1 if vessel i is served as the last vessel at the assigned berth (= 0 otherwise)

4.1.3 Auxiliary variables

$ST_i, i \in I$ Start time of service for vessel i (h)
 $WT_i, i \in I$ Waiting time of vessel i (h)
 $LD_i, i \in I$ Hours of late departure for vessel i (h)

4.1.4 Parameters

$w_i, i \in I$ Weight of vessel i
 $A_i, i \in I$ Expected arrival time of vessel i (h)
 $HT_{ij}, i \in I, j \in J$ Handling time of vessel i at berth j (h)
 $RD_i, i \in I$ Requested departure time of vessel i (h)
 $c_i^{HT}, i \in I$ Handling cost for vessel i (USD/h)
 $c_i^{WT}, i \in I$ Waiting cost for vessel i (USD/h)
 $c_i^{LD}, i \in I$ Penalty for late departure of vessel i (USD/h)
 M Large positive number

4.2 Mathematical model

A mixed-integer mathematical model for the discrete dynamic berth scheduling problem with vessel service priority (**BSP-VSP**), which is investigated in this study, can be formulated as follows.

BSP-VSP:

$$\min \left[\sum_{i \in I} \sum_{j \in J} (w_i HT_{ij} x_{ij} c_i^{HT}) + \sum_{i \in I} (w_i WT_i c_i^{WT}) + \sum_{i \in I} (w_i LD_i c_i^{LD}) \right] \quad (1)$$

Subject to:

$$\sum_{j \in J} x_{ij} = 1 \quad \forall i \in I \quad (2)$$

$$f_{i^*} + \sum_{i \in I: i \neq i^*} y_{ii^*} = 1 \quad \forall i^* \in I \quad (3)$$

$$l_i + \sum_{i^* \in I: i^* \neq i} y_{ii^*} = 1 \quad \forall i \in I \quad (4)$$

$$f_i + f_{i^*} \leq 3 - x_{ij} - x_{i^*j} \quad \forall i, i^* \in I, \quad i \neq i^*, \quad j \in J \quad (5)$$

$$l_i + l_{i^*} \leq 3 - x_{ij} - x_{i^*j} \quad \forall i, i^* \in I, \quad i \neq i^*, \quad j \in J \quad (6)$$

$$y_{ii^*} - 1 \leq x_{ij} - x_{i^*j} \leq 1 - y_{ii^*} \quad \forall i, i^* \in I, \quad i \neq i^*, \quad j \in J \quad (7)$$

$$ST_i \geq A_i \quad \forall i \in I \quad (8)$$

$$ST_{i^*} \geq ST_i + \sum_{j \in J} (HT_{ij} x_{ij}) - M(1 - y_{ii^*}) \quad \forall i, i^* \in I, \quad i \neq i^*, \quad j \in J \quad (9)$$

$$WT_i \geq ST_i - A_i \quad \forall i \in I \quad (10)$$

$$LD_i \geq ST_i + \sum_{j \in J} (HT_{ij} x_{ij}) - RD_i \quad \forall i \in I \quad (11)$$

$$x_{ij}, y_{ii^*}, f_i, l_i \in \{0, 1\} \quad \forall i, i^* \in I, \quad i \neq i^*, \quad j \in J \quad (12)$$

$$ST_i, WT_i, LD_i, w_i, A_i, HT_{ij}, RD_i, c_i^{HT}, c_i^{WT}, c_i^{LD}, M \in R^+ \quad \forall i \in I, \quad j \in J \quad (13)$$

In **BSP-VSP**, the objective function (1) is directed to minimize the total weighted vessel service cost, which is composed of 3 components: (1) the total weighted handling cost of vessels; (2) the total weighted waiting cost of vessels; and (3) the total weighted vessel late departure penalty. Constraint set (2) guarantees that the service should be provided at one of the berths for every vessel calling for service at the MCT. Constraint set (3) states that every vessel can be either served as the first vessel or after the other vessel, which is scheduled for service at the given MCT berth. Constraint set (4) indicates that every vessel can be either served as the last vessel or before the other vessel, which is scheduled for service at the given MCT berth. Constraint set (5) guarantees that only one vessel can be served first at every MCT berth. Constraint set (6) indicates that only one vessel can be served last at every MCT berth. Constraint set (7) ensures that a given vessel, requesting service at the MCT, can be served after the other vessel, if both vessels are to be served at the same MCT berth. Constraint set (8) guarantees that service of a given vessel can start only after arrival of that vessel at the MCT. Constraint set (9) calculates the start service time for every vessel calling for service at the MCT. Constraint set (10) calculates the waiting time for every vessel calling for service at the MCT. Constraint set (11) estimates the late departure hours for every vessel calling for service at the MCT. Constraint sets (12) and (13) show the ranges of parameters and variables in the **BSP-VSP** program.

5 Solution methodology

As it was mentioned in Sect. 2 of the manuscript, the BSPs are complex combinatorial optimization problems that can be further reduced to the machine scheduling problems. The problems of the latter class are known to have NP-hard complexity. The small-size problem instances of the **BSP-VSP** mathematical model can be solved to the global optimality using commercial mixed-integer programming solvers (e.g., CPLEX, MOSEK, GUROBI) within a reasonable computational time. However, development of heuristic or metaheuristic algorithms is required to solve the realistic-size problem instances (which include the problem

instances of medium and large sizes) within a reasonable computational time. This study proposes an Adaptive Island EA (AIEA) for solving the **BSP-VSP** mathematical model. Unlike canonical EAs, AIEA simultaneously executes separate EAs in parallel on its islands (one EA on each island), which independently explore different domains of the search space and exchange individuals between each other based on a certain rule (see Fig. 2). The latter allows preserving population diversity, enhancing the convergence patterns of the algorithm, and improving the value of the objective function at convergence [24]. The major steps of the proposed AIEA algorithm are shown in Algorithm 1.

that will participate in the AIEA operations (step 14); (2) function **AIEAoper**(*Parents_{qg}*, *MutRate*) creates the offspring chromosomes (step 15); (3) function **Repair**(*Offspring_{qg}*) repairs all the infeasible offspring produced as a result of the AIEA operations (step 16); (4) function **Evaluate**(*Offspring_{qg}*, *InputData*) computes the fitness function values for the offspring chromosomes (step 17); and (5) function **SurvSelect**(*Offspring_{qg}*, *Fit_{qg}*, Ω , Ψ) determines the offspring chromosomes that will survive in a given generation and will become potential parents in the consecutive generation (step 18). Next, if the migration criterion is met, AIEA exchanges the individuals between the islands (i.e., migration of individuals occurs in steps 21–23).

Algorithm 1: Adaptive Island Evolutionary Algorithm (AIEA)

```

AIEA(InputData, PopSize, MutRate,  $\Omega$ ,  $\Psi$ , Q, MigCriterion, MigSize, TermCriterion)
in: InputData - values of the BSP-VSP parameters; PopSize - population size; MutRate - mutation rate;  $\Omega$  -
tournament size;  $\Psi$  - amount of individuals selected at each tournament; Q = {1, ..., m} - set of islands;
MigCriterion - migration criterion; MigSize - individuals migrated; TermCriterion - termination criterion
out: BestChrom - the best berth schedule
1:  $g \leftarrow 1$                                  $\triangleleft$  Start the generation count
2:  $q \leftarrow 1$ 
3: for all  $q \in Q$  do
4:    $|Pop_{qg}| \leftarrow PopSize$ ;  $|Fit_{qg}| \leftarrow PopSize$ ;  $|Parents_{qg}| \leftarrow PopSize$ ;  $|Offspring_{qg}| \leftarrow PopSize$ 
5:   Chromq  $\leftarrow$  ChromInit(InputData)                                 $\triangleleft$  Initialize the chromosome
6:   Popqg  $\leftarrow$  PopInit(Chromq, PopSize)                                 $\triangleleft$  Initialize the population
7:   Fitqg  $\leftarrow$  Evaluate(Popqg, InputData)                                 $\triangleleft$  Compute fitness of the initial population
8:    $q \leftarrow q + 1$ 
9: end for
10: while TermCriterion  $\leftarrow$  FALSE do                                 $\triangleleft$  Update the generation count
11:    $g \leftarrow g + 1$ 
12:    $q \leftarrow 1$ 
13:   for all  $q \in Q$  do
14:     Parentsqg  $\leftarrow$  ParentSelect(Popqg)                                 $\triangleleft$  Determine the parent chromosomes
15:     Offspringqg  $\leftarrow$  AIEAoper(Parentsqg, MutRate)                                 $\triangleleft$  Create the offspring chromosomes
16:     Offspringqg  $\leftarrow$  Repair(Offspringqg)                                 $\triangleleft$  Repair the infeasible chromosomes
17:     Fitqg  $\leftarrow$  Evaluate(Offspringqg, InputData)                                 $\triangleleft$  Compute fitness of the offspring
18:     Popq(g+1)  $\leftarrow$  SurvSelect(Offspringqg, Fitqg,  $\Omega$ ,  $\Psi$ )                                 $\triangleleft$  Determine the survived chromosomes
19:      $q \leftarrow q + 1$ 
20:   end for
21:   if MigCriterion  $\leftarrow$  TRUE then
22:     Popg+1  $\leftarrow$  Exchange(Q, Popg+1, Fitg+1, MigSize)                                 $\triangleleft$  Exchange individuals between the islands
23:   end if
24: end while
25: BestChrom  $\leftarrow$  argmin(Fitg)
26: return BestChrom

```

In steps 1–9, the data structures for the AIEA variables, the initial chromosomes, and the initial population are constructed for each island. Furthermore, values of the fitness function for the chromosomes of the initial population are computed for each island. Next, AIEA enters the main loop (steps 10–24), where for each island the following steps are performed: (1) function **ParentSelect**(*Pop_{qg}*) selects the parent chromosomes

The procedure is terminated once the stopping criterion is satisfied. At termination, AIEA returns the best berth schedule (or the best chromosome—*BestChrom*) with the lowest total weighted service cost of vessels. The main steps of the proposed AIEA are presented in Sects. 5.1–5.9 of the manuscript.

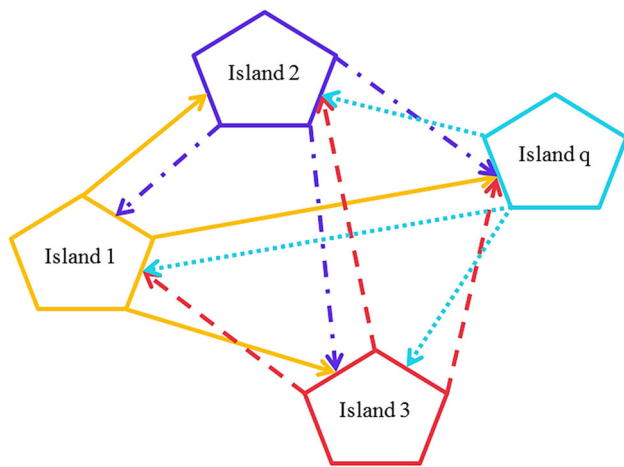


Fig. 2 Exchange of individuals between the islands

5.1 Representation of solutions

This study will be using an integer chromosome for representing a solution (i.e., assignment of vessels to the MCT berths) for each EA. Each chromosome consists of genes. The value of a gene (i.e., MCT berth or vessel ID) will be referred to as “*allele*”, while the location of a gene in the chromosome will be referred to as “*locus*” [24]. Note that throughout the manuscript such terms as chromosome, vessel assignment, and individual will be used interchangeably as they have the same meaning. An example chromosome is illustrated in Fig. 3, where the MCT has 2 berths, and 8 vessels are calling for service. According to the berth schedule,

vessels “2”, “6”, “7”, and “8” are assigned for service at berth “1” (in that particular order), while vessels “1”, “3”, “4”, and “5” are assigned for service at berth “2” (in that particular order). The chromosome length is assumed to be the same during the EA evolution on each island.

5.2 Initialization of chromosomes and populations

This study uses a local search heuristic for the chromosome initialization. The heuristic relies on the first-come-first-served (FCFS) policy, which is commonly used in practice by MCT operators for constructing berth schedules. Application of the local search heuristic for the chromosome initialization will increase the time complexity of the algorithm as compared to the case with randomly initialized chromosomes. Nevertheless, the quality of individuals, initialized using a local search heuristic, is typically better than the quality of randomly initialized individuals [24]. Furthermore, deployment of random operators at the chromosome initialization stage may cause formation of the infeasible individuals, which may further result in a genetic drift and affect the objective function value at convergence [24]. Denote S as a set of vessels calling for service at the MCT and sorted based on their times of arrival in the ascending order; BA_j as the time, when berth j is available for service of vessels for the first time in the considered planning horizon; and FT_i as the finish service time for vessel i (which is measured in hours). The remaining notations will be adopted from Sect. 4.1 of the manuscript. The key steps of the chromosome initialization process are shown in Algorithm 2.

Algorithm 2: Chromosome Initialization

ChromInit(I, J, A, HT)

in: $I = \{1, \dots, n\}$ - set of vessels; $J = \{1, \dots, k\}$ - set of berths; A - vessel arrival times; HT - vessel handling times

out: x - initial vessel to berth assignment

- 1: $|BA| \leftarrow k$; $|x| \leftarrow n \cdot k$; $|S| \leftarrow n$; $|ST| \leftarrow n$; $|FT| \leftarrow n$ ◁ Data structure initialization
 - 2: $S \leftarrow \text{Sort}(I, A)$ ◁ Sort vessels based on their times of arrival
 - 3: $i \leftarrow 1$
 - 4: **for all** $i \in S$ **do**
 - 5: $j \leftarrow \text{argmin}(BA)$ ◁ Identify the first available berth
 - 6: $x_{ij} \leftarrow 1$ ◁ Assign a vessel to that berth in the earliest service order
 - 7: $ST_i \leftarrow \max(A_i, BA_j)$ ◁ Compute the start service time of a vessel
 - 8: $FT_i \leftarrow ST_i + HT_{ij}$ ◁ Compute the finish service time of a vessel
 - 9: $BA_j \leftarrow FT_i$ ◁ Update the berth availability
 - 10: $i \leftarrow i + 1$
 - 11: **end for**
 - 12: **return** x
-

Fig. 3 Chromosome representation

MCT Berth	1	1	1	1	1	1	1	1	2	2	2	2	2	2	2	2	2
Vessel ID	2	6	7	8	0	0	0	0	1	3	4	5	0	0	0	0	0

Order of vessels at berth "1" Order of vessels at berth "2"

Fig. 4 An infeasible individual generated by the one-point crossover operator

Before One-Point Crossover																	
Parent 1																	
MCT Berth	1	1	1	1	1	1	1	1	2	2	2	2	2	2	2	2	2
Vessel ID	1	6	7	8	0	0	0	0	2	3	4	5	0	0	0	0	0
Parent 2																	
MCT Berth	1	1	1	1	1	1	1	1	2	2	2	2	2	2	2	2	2
Vessel ID	2	4	5	8	0	0	0	0	1	3	6	7	0	0	0	0	0
After One-Point Crossover																	
Offspring 1																	
MCT Berth	1	1	1	1	1	1	1	1	2	2	2	2	2	2	2	2	2
Vessel ID	1	6	7	8	0	0	0	0	2	3	6	7	0	0	0	0	0
Offspring 2																	
MCT Berth	1	1	1	1	1	1	1	1	2	2	2	2	2	2	2	2	2
Vessel ID	2	4	5	8	0	0	0	0	1	3	4	5	0	0	0	0	0

The data structures, required for the chromosome initialization process, are created in step 1. Next, the vessels are sorted based on their times of arrival at the MCT in the ascending order in step 2. After that, the procedure starts an iterative process, where the first available MCT berth is identified in step 5. Next, a vessel is scheduled for service at the first available berth (in the earliest order of service) in step 6. The start and finish service times of a vessel are computed in steps 7 and 8, respectively. After that, the availability of a berth is updated based on the vessel finish time in step 9. The chromosome initialization procedure terminates, when the initial vessel to berth assignment is generated for all the vessels calling for service at the MCT. This study constructs the initial population on each island using the identical chromosomes. Various initial population sizes (*PopSize*) will be considered throughout the AIEA parameter tuning analysis (details are further described in Sect. 6.2 of the manuscript). The number of individuals in the population does not alter from one generation to another on each island of the AIEA algorithm.

5.3 Parent selection

This study relies on the “*deterministic parent selection scheme*”, assuming that every individual survived in the given generation is able produce the offspring in the consecutive generation (i.e., all the survived individuals become parents). The latter parent selection mechanism has been commonly deployed in Evolutionary Programming [24].

5.4 AIEA operations

Canonical EAs typically rely on the crossover and mutation operators to generate and mutate the offspring chromosomes. For the adopted chromosome representation (see Fig. 3), the common crossover operators (e.g., uniform crossover, one-point crossover, half-uniform crossover, multi-point crossover, etc.) may produce the chromosomes with a complex infeasibility, which have the genes representing the same vessels. An example of an infeasible individual, produced by the one-point crossover operator, is presented in Fig. 4. We

observe that the one-point crossover operator produced two infeasible offspring from the feasible parents. The infeasibility of the first offspring consists in the fact that vessels “6” and “7” are scheduled for service twice, while vessels “4” and “5” are not scheduled at all. The infeasibility of the second offspring consists in the fact that vessels “4” and “5” are scheduled for service twice, while vessels “6” and “7” are not scheduled at all. The infeasible offspring can be repaired, but the computational efforts for repairing such infeasibility will be significant (with at least $O(|Q| \cdot PopSize^2)$ time complexity). To avoid an increasing time complexity of the AIEA algorithm, only the mutation operator will be deployed in this study throughout the AIEA operations to produce and mutate the offspring.

5.5 Repairing operations

As a result of the AIEA operation, the swap mutation may produce the infeasible chromosomes. An example of an infeasible chromosome is illustrated in Fig. 6, where the strings of genes with “zero” alleles are inserted between the genes with “non-zero” alleles. The existence of “zero” alleles, which are not associated with any vessels, may cause an erroneous estimation of the fitness function values for the offspring chromosomes. To address the latter issue, a repairing operator was developed within the proposed AIEA. The key steps, which are necessary for repairing the infeasible chromosomes on each island, are outlined in Algorithm 3.

Algorithm 3: Repairing Operation

```

Repair(Offspringqg)
in: Offspringqg - offspring chromosomes in generation g of island q that contain infeasible individuals
out: Offspringqg - repaired offspring chromosomes in generation g of island q
1:  $|\Phi| \leftarrow |\text{Offspring}_{qg}|$  ◁ Initialization
2:  $o \leftarrow 1$ 
3: while  $o \leq |\text{Offspring}_{qg}|$  do
4:    $Vessels_o \leftarrow \text{ExtractZeros}(\text{Offspring}_{qg_o})$  ◁ Extract “zero” alleles
5:    $\Phi_o \leftarrow Vessels_o$  ◁ Copy a vessel to berth assignment to a new data structure
6:    $o \leftarrow o + 1$ 
7: end while
8: Offspringqg  $\leftarrow \Phi$  ◁ Replace the infeasible offspring
9: return Offspringqg

```

Different types of the mutation operators have been widely deployed in the literature [24], including the following: invert, bit flipping, floating point, swap, scramble, insert, etc. The proposed AIEA will use the swap mutation operator to generate the offspring chromosomes, as it was found to be effective for the adopted chromosome representation [7, 8, 12]. Note that the swap mutation operator may also cause infeasibility, which can be repaired significantly faster as compared to the infeasibility caused by the crossover operators (details are described in Sect. 5.5 of the manuscript). An example of the AIEA operation is shown in Fig. 5, where vessel “4” is diverted for service at berth “1”, while vessels “2” and “8” are diverted for service at berth “2”. The number of genes swapped for each individual in the population on each island is determined based on the mutation rate (*MutRate*). The *MutRate* value will be selected based on the AIEA parameter tuning analysis (details are further described in Sect. 6.2 of the manuscript).

A new data structure (Φ) is generated in step 1 to store the repaired offspring individuals. After that, the repairing operator executes an iterative procedure, where for every offspring individual of the population on the given island function **ExtractZeros**(*Offspring*_{qgo}) removes “zero” alleles for every string of alleles that correspond to the vessels, which are assigned to a given MCT berth (step 4). Then, the “non-zero” string of alleles is stored in the new data structure (step 5). The repaired offspring individuals replace the infeasible offspring individuals in step 8. Figure 6 demonstrates an example of the offspring repairing operation.

5.6 Fitness function

The fitness function is generally associated with the objective function of the problem in canonical EAs [24]. Within the developed AIEA, the fitness function is assumed to be identical to the objective function of the **BSP-VSP** mathematical model.

Fig. 5 The AIEA operation example

Before AIEA Operations																	
MCT Berth	1	1	1	1	1	1	1	1	2	2	2	2	2	2	2	2	2
Vessel ID	2	6	7	8	0	0	0	0	1	3	4	5	0	0	0	0	0

After AIEA Operations																	
MCT Berth	1	1	1	1	1	1	1	1	2	2	2	2	2	2	2	2	2
Vessel ID	4	6	7	0	0	0	0	0	1	3	2	5	8	0	0	0	0

Fig. 6 Repairing operation example

An Infeasible Individual																	
MCT Berth	1	1	1	1	1	1	1	1	2	2	2	2	2	2	2	2	2
Vessel ID	2	0	0	0	6	7	8	0	1	3	0	0	4	5	0	0	0

A Repaired Individual																	
MCT Berth	1	1	1	1	1	1	1	1	2	2	2	2	2	2	2	2	2
Vessel ID	2	6	7	8	0	0	0	0	1	3	4	5	0	0	0	0	0

5.7 Survivor selection

A survivor selection identifies the individuals from the offspring, which will survive in the given generation of the AIEA algorithm and will serve as potential parents in the consecutive generation. The Tournament Selection will be deployed within the proposed AIEA algorithm [24]. The Tournament Selection mechanism launches multiple tournaments. A pre-determined quantity of individuals (generally referred to as the tournament size— Ω) are chosen at random from the offspring at each tournament. After that, only a specific number of individuals with the highest values of the fitness function (i.e., the “fittest” individuals) out of the chosen individuals (generally referred as the amount of individuals selected— Ψ) will be transferred to the consecutive generation. Various values of the Tournament Selection parameters (i.e., Ω and Ψ) will be considered throughout the AIEA parameter tuning analysis (details are further described in Sect. 6.2 of the manuscript). The Elitist Strategy will be employed in the developed AIEA to guarantee that the individual with the highest fitness will be present in the consecutive generation [24].

5.8 Migration

Unlike canonical EAs, AIEA has two additional parameters: (1) migration interval; and (2) migration size. The migration interval denotes the frequency of exchanging individuals between the islands, whereas the migration size determines

the number of individuals migrating from one island to every other island. This study adopts an adaptive mechanism for the migration interval (hence, the proposed Island EA becomes an adaptive algorithm). Specifically, the migration of individuals occurs if no improvements in the objective function values over all the islands have been observed after $MaxGen$ generations (i.e., migration is performed based on the feedback from the search). It is assumed that if the migration criterion is satisfied, a pre-determined number of individuals ($MigSize$) will migrate from one island to every other island (i.e., the total number of migrating individuals over all the islands will be $MigSize \cdot |Q - 1| \cdot |Q|$). Furthermore, only the “fittest” individuals are selected for migration. Those individuals will replace the “weakest” individuals on the island they are migrating to (i.e., individuals with the lowest fitness function values). Hence, the EA population size will remain the same after migration on each island of AIEA. The main steps of migration process are presented in Algorithm 4.

The data structures for storing the “fittest” and the “weakest” individuals are initialized in step 1. Next, the procedure enters the loop, where functions $SelectFittest(Q, Pop_g, Fit_g, MigSize)$ and $SelectWeakest(Q, Pop_g, Fit_g, MigSize)$ are used to collect a pre-determined number (equal to $MigSize$) of the “fittest” and the “weakest” individuals from each island in steps 4 and 5. Then, a set of the “weakest” individuals ($List_{qg}$) in a given population is replaced on each island with a set of the “fittest” individuals (\overline{List}_{qg}), collected from the rest of the islands, in steps 10 and 11. Decreasing

MaxGen values would increase the migration frequency of individuals among the islands of the AIEA algorithm. Furthermore, increasing *MigSize* values would increase the number of individuals that will migrate from one island to every other island. Decreasing *MaxGen* values and increasing *MigSize* values are expected to improve the quality of solutions, returned by the AIEA algorithm, but would incur an additional computational time (since the migration process would be more frequent and the number of migrating individuals would increase). Different values of *MaxGen* and *MigSize* parameters will be considered throughout the AIEA parameter tuning analysis (details are further described in Sect. 6.2 of the manuscript). Note that the scope of future research for this study includes the development of additional control strategies for *MaxGen* and *MigSize* parameters and evaluation of how these control strategies would affect the overall performance of the proposed AIEA algorithm.

6 Computational experiments

This section of the manuscript presents the computational experiments that were undertaken to assess the effectiveness of the proposed AIEA algorithm. The AIEA performance was evaluated based on a comparative analysis against the other state-of-the-art metaheuristic algorithms, which have been widely used in the BSP literature. More specifically, the AIEA algorithm was compared against the following solution algorithms: (1) a typical EA (i.e., an AIEA with only one island that does not perform any exchange of individuals); (2) TS; and (3) SA. For a detailed description of the TS algorithm, this study refers to Cordeau et al. [18]. On the other hand, a detailed description of the SA algorithm is provided by Emde and Boysen [16]. Furthermore, throughout the optimality gap analysis, performance of the considered solutions algorithms (i.e., the AIEA, EA, TS, and SA algorithms) was assessed against an exact optimization algorithm, which is able to provide the global optimal solution for the **BSP-VSP** mathematical model. The AIEA, EA,

Algorithm 4: Migration

Exchange($Q, Pop_g, Fit_g, MigSize$)
in: $Q = \{1, \dots, m\}$ - set of islands; Pop_g - population in generation g ; Fit_g - fitness values of individuals in generation g ; $MigSize$ - migration size
out: Pop_g - updated population in generation g

- 1: $|List| \leftarrow MigSize \cdot m$; $|List| \leftarrow MigSize \cdot m$ \triangleleft Initialization
- 2: $q \leftarrow 1$
- 3: **for all** $q \in Q$ **do**
- 4: $List_{qg} \leftarrow SelectFittest(Q, Pop_g, Fit_g, MigSize)$ \triangleleft Select the “fittest” individuals
- 5: $List_{qg} \leftarrow SelectWeakest(Q, Pop_g, Fit_g, MigSize)$ \triangleleft Select the “weakest” individuals
- 6: $q \leftarrow q + 1$
- 7: **end for**
- 8: $q \leftarrow 1$
- 9: **for all** $q \in Q$ **do**
- 10: $Pop_{qg} \leftarrow Pop_{qg} - \{List_{qg}\}$ \triangleleft Remove the “weakest” individuals from a given island
- 11: $Pop_{qg} \leftarrow Pop_{qg} \cup \{List_{qg}\}$ \triangleleft Append the “fittest” individuals collected from the other islands
- 12: $q \leftarrow q + 1$
- 13: **end for**
- 14: **return** Pop_g

5.9 Termination criterion

The maximum number of generations (*GenLimit*) is set as a termination criterion for the developed AIEA algorithm. The value of *GenLimit* parameter will be established throughout the AIEA parameter tuning analysis (details are further described in Sect. 6.2 of the manuscript).

TS, and SA algorithms were coded using MATLAB 2016a. The computational experiments were conducted on a CPU with Dell Intel(R) Core™ i7 Processor, 32 GB of RAM, and Windows 10 Operating System. The AIEA algorithm was designed using the Parallel Computing Toolbox of MATLAB 2016a to ensure that separate EAs would be executed in parallel on the AIEA islands. The main steps of the conducted numerical experiments are presented next, including the input data generation, parameter tuning, optimality gap

analysis, comprehensive comparative analysis of the algorithms, and evaluation of the proposed adaptive mechanism for the exchange of individuals between the AIEA islands.

6.1 Input data generation

The input data necessary for the computational experiments were generated using the available MCT and liner shipping operations literature [4, 5, 7–12, 20, 25–27] and are shown in Table 1. The weights of vessels were assigned as: $w_i = U[0.1; 1.0] \forall i \in I$, where U —is a notation used for the uniformly distributed pseudorandom numbers. The vessel arrival pattern was modeled using the exponential distribution [7–12, 20] with the average inter-arrival time of 2 h: $exp[2]$, where exp —is a notation used for the exponentially distributed pseudorandom numbers. A total of three MCT berthing capacity cases were considered during the computational experiments, including the following: (a) 2 berths; (b) 4 berths; and (c) 6 berths.

The amount of containers, handled by quay cranes for each vessel, was initialized as: $NC_i = U[500; 3,000] \forall i \in I$ (TEUs). The container handling productivity at the “preferred berth” was assumed to be equal to $hp_{ij_i^*} = 150 \forall i \in I, j_i^* \in J$ (TEUs/h), where j_i^* —is a notation for the “preferred berth” of vessel i . Note that FCFS policy was used to determine the “preferred berth” for each one of the vessels calling for service at the MCT. The handling time of vessel i at berth j was computed as: $HT_{ij} = \frac{NC_i}{hp_{ij_i^*}} \cdot (1 + 0.15 \cdot |j_i^* - j|) \forall i \in I, j, j_i^* \in J$ (h). Note that the second term of the HT_{ij} equation (i.e., $[1 + 0.15 \cdot |j_i^* - j|]$) captures the fact that the vessel handling time will increase proportionally to the distance between the assigned berth and “preferred berth” for that vessel. The requested departure time for each vessel was assigned as a function of its arrival and handling times as follows: $RD_i = A_i + HT_{ij_i^*} \cdot U[1.0; 1.2] \forall i \in I, j_i^* \in J$ (h). The handling cost of vessels was initialized as: $c_i^{HT} = U[60,000; 90,000] \forall i \in I$ (USD/h). The waiting cost of vessels was assigned as: $c_i^{WT} = U[500; 1,500] \forall i \in I$ (USD/h); while the late departure penalty was generated as: $c_i^{LD} = U[5,000; 8,000] \forall i \in I$ (USD/h).

Based on the generated values of the input data, a total of 60 problem instances were developed by changing the number of vessels, requesting service at the MCT, and the MCT berthing capacity. All the developed problem instances were categorized into the following two groups: (1) small-size problem instances (S1–S30), where the number of vessels was altered from 6 to 20 with an increment of 1 ÷ 2 vessels, while the MCT berthing capacity was altered from 2 to 6 with an increment of 2 berths; and (2) realistic-size problem instances (i.e., medium- and large-size problem instances: R1–R30), where the number of vessels was altered from

40 to 85 with an increment of 5 vessels, while the MCT berthing capacity was altered from 2 to 6 with an increment of 2 berths. The small-size problem instances were used to estimate the optimality gaps for the considered solution algorithms (details are further described in Sect. 6.3 of the manuscript), while the realistic-size problem instances were used for a detailed comparative analysis of the algorithms (details are further described in Sect. 6.4 of the manuscript) and evaluation of the proposed adaptive mechanism for the exchange of individuals between the AIEA islands (details are further described in Sect. 6.5 of the manuscript).

6.2 Parameter tuning analysis

The AIEA, EA, TS, and SA algorithms use a set of parameters. Values of those parameters must be set based on a parameter tuning analysis [24]. This study adopts a “factorial design” for the parameter tuning analysis [28–30], where each one of the algorithmic parameters (i.e., “factors”) is assumed to have a set of candidate values (i.e., “levels”). A total of three problem instances were chosen at random from the generated realistic-size problem instances, presented in Sect. 6.1 of the manuscript, to perform the parameter tuning for the considered solution algorithms. Each parameter was assumed to have three candidate values for each algorithm, i.e. a 3^k factorial design with 3 levels and k factors. The AIEA, EA, TS, and SA algorithms were launched five times for each parameter combination. Hence, a total of $3^k \cdot inst \cdot rep$ (where $inst$ —is the number of instances [$inst = 3$], and rep —is the number of replications [$rep = 5$]) experimental runs were conducted for the AIEA, EA, TS, and SA algorithms during the parameter tuning analysis. A summary of the parameter tuning analysis for the considered solutions algorithms is presented in Table 2, where the following information is provided: (1) the algorithm; (2) the parameter description; (3) the considered candidate values; and (4) the best parameter value (which was selected based on a tradeoff between the solution quality and the computational time).

The results from parameter tuning analysis indicate that the most favorable EA population size was $PopSize = 60$ individuals, while the most favorable AIEA population size was $PopSize = 20$ individuals per island. Since the number of AIEA islands was set to $|Q| = 3$ islands, AIEA will not have advantages over EA in terms of the population size throughout the numerical experiments. Based on the experimental runs, it was found that the objective function did not alter substantially after 2500 generations for the AIEA and EA algorithms. Furthermore, the objective function did not alter substantially after 2500 iterations for the TS and SA algorithms. Hence, the AIEA and EA algorithms will be terminated once $GenLimit = 2500$ generations is reached, while the TS and SA algorithms will be terminated after $IterLimit = 2500$ iterations.

Table 1 Numerical data

Parameter	Value
Vessel weight: $w_i, i \in I$	$U[0.1; 1.0]$
Vessel inter-arrival time (h)	$exp [2]$
MCT berthing capacity (number of berths)	$[2; 4; 6]$
Amount of containers assigned to each vessel: $NC_i, i \in I$ (TEUs)	$U[500; 3, 000]$
Handling productivity at the “preferred berth”: $hp_{ij_i^*}, i \in I, j_i^* \in J$ (TEUs/h)	150
Vessel handling time: $HT_{ij}, i \in I, j \in J$ (h)	$HT_{ij} = \frac{NC_i}{hp_{ij_i^*}} \cdot (1 + 0.15 \cdot j_i^* - j)$
Requested vessel departure: $RD_i, i \in I$ (h)	$RD_i = A_i + HT_{ij_i^*} \cdot U[1.0; 1.2]$
Vessel handling cost: $c_i^{HT}, i \in I$ (USD/h)	$U[60, 000; 90, 000]$
Vessel waiting cost: $c_i^{WT}, i \in I$ (USD/h)	$U[500; 1, 500]$
Late departure penalty: $c_i^{LD}, i \in I$ (USD/h)	$U[5, 000; 8, 000]$

6.3 Optimality gap analysis

The optimality gap analysis was conducted using the small-size problem instances (S1–S30) with the number of vessels varying from 6 to 20 vessels and the MCT berthing capacity varying from 2 to 6 berths (details regarding the input data generation for the small-size problem instances are presented in Sect. 6.1 of the manuscript). CPLEX was used to obtain the optimal objective function values for all the considered small-size problem instances. The CPLEX relative optimality gap was set equal to 1.00%, whereas the limit on the CPU time was set to 2 h. The AIEA, EA, TS, and SA algorithms were also launched for all the developed small-size problem instances. The results of the optimality gaps analysis are summarized in Table 3, where the following information is provided: (1) the instance number; (2) the number of vessels that request service at the MCT; (3) the MCT berthing capacity; (4) the CPLEX objective function value; (5) the CPLEX CPU time; (6) the average objective function values (over 5 replications) for the AIEA, EA, TS, and SA algorithms; (7) the optimality gap values for the AIEA, EA, TS, and SA algorithms; and (8) the average CPU time (over 5 replications) for the AIEA, EA, TS, and SA algorithms. The optimality gap was computed for each algorithm as follows: $\Delta_a = \frac{Z_a - Z_{CPLEX}}{Z_{CPLEX}}$, where Δ_a —is the optimality gap for algorithm a ($a = AIEA, EA, TS, SA$), Z_a —is the objective function value obtained by algorithm a , and Z_{CPLEX} —is the optimal objective function value obtained by CPLEX.

It can be noticed that due to NP-hard complexity of the **BSP-VSP** mathematical model, the CPLEX CPU time increases exponentially with increasing problem size. Furthermore, CPLEX was not able to provide a solution within 2 h for problem instances S10, S20, S29, and S30. The optimality gap values did not exceed 1.93%, 3.60%, 3.69%, and 3.77% for the AIEA, EA, TS, and SA algorithms, respectively, which showcases a high accuracy of the considered

solution algorithms. The CPU time averaged on 14.99 s, 17.32 s, 15.95 s, and 1.59 s over the generated small-size problem instances for the AIEA, EA, TS, and SA algorithms, respectively.

6.4 Comparison of the algorithms

A detailed comparison of the AIEA, EA, TS, and SA algorithms was performed using the realistic-size problem instances (R1–R30) with the number of vessels varying from 40 to 85 vessels and the MCT berthing capacity varying from 2 to 6 berths (details regarding the input data generation for the realistic-size problem instances are presented in Sect. 6.1 of the manuscript). All the considered solution algorithms were evaluated based on the obtained objective function values at termination and CPU time. Furthermore, convergence patterns of the AIEA, EA, TS, and SA algorithms were investigated as well.

6.4.1 Objective function values and CPU time

The AIEA, EA, TS, and SA algorithms were executed for each one of the realistic-size problem instances, and the results of the analysis are provided in Table 4, which presents the following data: (1) the instance number; (2) the number of vessels calling for service at the MCT; (3) the MCT berthing capacity; (4) the average objective function values (over 5 replications) for the AIEA, EA, TS, and SA algorithms; and (5) the average CPU time (over 5 replications) for the AIEA, EA, TS, and SA algorithms. Findings from the conducted analysis indicate that AIEA consistently outperforms EA, TS, and SA in terms of the objective function values for all the considered realistic-size problem instances. A simultaneous execution of separate EAs in parallel on multiple islands and the proposed adaptive mechanism for the exchange of individuals between the islands on average yielded 7.21%,

Table 2 Results of the parameter tuning analysis for the considered solution algorithms

Algorithm	Parameter	Candidate values	Best value	Algorithm	Parameter	Candidate values	Best value
AIEA	Number of islands— $ Q $	[2; 3; 4]	3	EA	Mutation rate— $MutRate$	[2; 4; 6]	2
AIEA	Migration size (% of $PopSize$)— $MigSize$	[5; 10; 15]	15	EA	Tournament size (% of $PopSize$)— Ω	[30; 40; 50]	50
AIEA	Migration criterion: number of generations with no improvements in the objective function values over all the islands— $MaxGen$	[50; 100; 150]	100	EA	Number of individuals selected at each tournament (% of $PopSize$)— ψ	[10; 15; 20]	10
AIEA	Population size (per island)— $PopSize$	[20; 30; 40]	20	TS	Number of solutions evaluated during the local search— $NumSol^{TS}$	[20; 40; 60]	60
AIEA	Mutation rate— $MutRate$	[2; 4; 6]	2	TS	Exchange rate ^a — $Rate^{TS}$	[2; 4; 6]	2
AIEA	Tournament size (% of $PopSize$)— Ω	[30; 40; 50]	50	TS	Tabu List ^b size (% of $NumSol^{TS}$)— $List^{TS}$	[10; 15; 20]	10
AIEA	Number of individuals selected at each tournament (% of $PopSize$)— ψ	[10; 15; 20]	10	SA	Initial temperature— t^0	[1250; 1500; 1750]	1500
EA	Population size— $PopSize$	[20; 40; 60]	60	SA	Temperature interval ^c — Δt	[0.50; 0.75; 1.00]	0.50
				SA	Exchange rate ^a — $Rate^{SA}$	[2; 4; 6]	2

^aThe exchange rate parameter denotes the number of vessel to berth assignments to be changed in a given solution in order to generate a new solution during the local search

^bTabu List contains the solutions that cannot be visited during the local search

^cNote that the temperature at iteration $iter$ was estimated within the SA algorithm based on the following relationship: $t_{iter} = t^0 - \Delta t \cdot iter$ (temperature units)

8.17%, and 10.11% reduction in the objective function values as compared to the EA, TS, and SA algorithms. Superiority of the AIEA and EA algorithms over the TS and SA algorithms in terms of the objective function values can be supported by the fact that both AIEA and EA are population-based metaheuristics and are able to discover solutions more efficiently, when compared to TS and SA that are single solution-based metaheuristics.

Throughout the numerical experiments, the CPU time averaged on 33.59 s, 44.42 s, 39.72 s, and 5.78 s over the generated realistic-size problem instances for the AIEA, EA, TS, and SA algorithms, respectively. Superiority of the AIEA algorithm over the EA algorithm in terms of the CPU time can be justified by a parallel execution of separate EAs on the AIEA islands. Therefore, a parallel execution of separate EAs on the AIEA islands allowed the AIEA algorithm outperforming the EA algorithm in terms of the CPU time on average by 32.23% over the generated realistic-size problem instances. An increase in the CPU time of the EA algorithm as compared to the TS and SA algorithms can be explained by the fact that EA is a population-based metaheuristic and performs algorithmic operations with a population of candidate solutions. The latter incurs larger CPU time as compared to the TS and SA algorithms that are single solution-based metaheuristics. A significantly lower CPU time of SA, when comparing to TS, can be supported by the fact that SA evaluates only two solutions at each iteration (i.e., the current solution and the neighbor solution), while TS evaluates multiple solutions at each iteration throughout the local search.

In order to determine whether the objective function values, recorded for the developed AIEA algorithm over 5 replications, were statistically different from the objective function values, recorded for the other considered solution algorithms (i.e., EA, TS, and SA) over 5 replications, a detailed statistical analysis was conducted for each realistic-size problem instance. A total of three types of the analysis of variance (ANOVA) tests were performed for each realistic-size problem instance, including the following: (1) ANOVA-1 test with the null hypothesis, assuming that the objective function values, recorded for AIEA over 5 replications, were equal to the objective function values, recorded for SA over 5 replications; (2) ANOVA-2 test with the null hypothesis, assuming that the objective function values, recorded for AIEA over 5 replications, were equal to the objective function values, recorded for TS over 5 replications; and (3) ANOVA-3 test with the null hypothesis, assuming that the objective function values, recorded for AIEA over 5 replications, were equal to the objective function values, recorded for EA over 5 replications. The ANOVA analysis results are provided in Table 5, which presents the following data for each realistic-size problem instance: (1) the instance number; (2) the number of vessels calling for service at the

Table 3 Results of the optimality gap analysis for the considered solution algorithms

Instance	#Vessels	#Berths	CPLEX	SA		TS		EA		AIEA						
				Z_{SA} , 10 ⁶ USD	Δ_{SA}	CPU, min	Z_{TS} , 10 ⁶ USD	Δ_{TS}	CPU, min	Z_{EA} , 10 ⁶ USD	Δ_{EA}	CPU, min	Z_{AIEA} , 10 ⁶ USD	Δ_{AIEA}	CPU, min	
S1	6	2	4.30	0.01	4.30	0.00%	0.02	4.30	0.00%	0.19	4.30	0.00%	0.20	4.30	0.00%	0.19
S2	7	2	5.12	0.01	5.12	0.00%	0.02	5.12	0.00%	0.20	5.12	0.00%	0.21	5.12	0.00%	0.19
S3	8	2	5.83	0.02	5.83	0.00%	0.02	5.83	0.00%	0.21	5.83	0.00%	0.21	5.83	0.00%	0.19
S4	9	2	6.19	0.04	6.19	0.00%	0.02	6.19	0.00%	0.21	6.19	0.00%	0.22	6.19	0.00%	0.20
S5	10	2	7.37	0.06	7.37	0.00%	0.02	7.37	0.00%	0.22	7.37	0.00%	0.23	7.37	0.00%	0.21
S6	12	2	8.90	0.19	8.90	0.00%	0.02	8.90	0.00%	0.23	8.90	0.00%	0.24	8.90	0.00%	0.22
S7	14	2	10.51	0.26	10.57	0.53%	0.02	10.57	0.53%	0.24	10.57	0.53%	0.26	10.51	0.00%	0.23
S8	16	2	12.35	0.75	12.49	1.10%	0.02	12.49	1.10%	0.25	12.49	1.10%	0.27	12.40	0.41%	0.24
S9	18	2	13.40	52.32	13.72	2.37%	0.02	13.72	2.37%	0.26	13.72	2.37%	0.29	13.57	1.30%	0.26
S10	20	2	N/A	>120	15.52	N/A	0.02	15.50	N/A	0.27	15.50	N/A	0.30	15.20	N/A	0.27
S11	6	4	4.25	0.01	4.25	0.00%	0.02	4.25	0.00%	0.22	4.25	0.00%	0.24	4.25	0.00%	0.22
S12	7	4	5.07	0.01	5.07	0.00%	0.02	5.07	0.00%	0.23	5.07	0.00%	0.25	5.07	0.00%	0.22
S13	8	4	5.74	0.05	5.74	0.00%	0.02	5.74	0.00%	0.24	5.74	0.00%	0.26	5.74	0.00%	0.23
S14	9	4	6.10	0.06	6.10	0.00%	0.02	6.10	0.00%	0.24	6.10	0.00%	0.26	6.10	0.00%	0.23
S15	10	4	7.21	0.21	7.21	0.00%	0.02	7.21	0.00%	0.25	7.21	0.00%	0.27	7.21	0.00%	0.24
S16	12	4	8.65	2.45	8.65	0.00%	0.02	8.65	0.00%	0.26	8.65	0.00%	0.29	8.65	0.00%	0.25
S17	14	4	10.15	10.97	10.25	0.96%	0.03	10.25	0.96%	0.28	10.25	0.96%	0.30	10.15	0.00%	0.26
S18	16	4	11.85	21.23	12.05	1.70%	0.03	12.05	1.70%	0.29	12.05	1.70%	0.32	11.91	0.55%	0.28
S19	18	4	12.70	69.37	13.15	3.51%	0.03	13.15	3.51%	0.31	13.10	3.12%	0.34	12.92	1.74%	0.29
S20	20	4	N/A	>120	14.81	N/A	0.04	14.78	N/A	0.32	14.71	N/A	0.36	14.39	N/A	0.31
S21	6	6	4.25	0.008	4.25	0.00%	0.02	4.25	0.00%	0.25	4.25	0.00%	0.28	4.25	0.00%	0.24
S22	7	6	5.07	0.014	5.07	0.00%	0.02	5.07	0.00%	0.26	5.07	0.00%	0.29	5.07	0.00%	0.24
S23	8	6	5.74	0.022	5.74	0.00%	0.03	5.74	0.00%	0.27	5.74	0.00%	0.29	5.74	0.00%	0.25
S24	9	6	6.10	0.071	6.10	0.00%	0.03	6.10	0.00%	0.27	6.10	0.00%	0.30	6.10	0.00%	0.26
S25	10	6	7.22	0.24	7.22	0.00%	0.03	7.22	0.00%	0.28	7.22	0.00%	0.31	7.22	0.00%	0.27
S26	12	6	8.67	3.217	8.75	0.93%	0.03	8.75	0.93%	0.31	8.75	0.93%	0.33	8.67	0.00%	0.28
S27	14	6	10.10	11.72	10.37	2.66%	0.04	10.37	2.66%	0.32	10.37	2.66%	0.35	10.18	0.84%	0.30
S28	16	6	11.75	80.12	12.19	3.77%	0.04	12.18	3.69%	0.34	12.17	3.60%	0.37	11.98	1.93%	0.31
S29	18	6	N/A	>120	13.40	N/A	0.05	13.40	N/A	0.36	13.35	N/A	0.39	13.00	N/A	0.32
S30	20	6	N/A	>120	15.04	N/A	0.05	15.01	N/A	0.38	14.92	N/A	0.41	14.59	N/A	0.33
Average:			7.87	9.75	8.85	0.67%	0.03	8.84	0.67%	0.27	8.83	0.65%	0.29	8.75	0.26%	0.25

Table 4 The objective function values and CPU time values for the considered solution algorithms

Instance	#Vessels	#Berths	SA		TS		EA		AIEA	
			Z_{SA} , 10^6 USD	CPU, s	Z_{TS} , 10^6 USD	CPU, s	Z_{EA} , 10^6 USD	CPU, s	Z_{AIEA} , 10^6 USD	CPU, s
R1	40	2	36.56	2.57	36.14	23.89	35.87	27.72	33.93	22.05
R2	45	2	42.97	2.89	42.63	26.10	41.84	30.20	39.55	23.49
R3	50	2	50.12	3.23	49.18	27.53	48.45	32.39	45.54	25.14
R4	55	2	57.60	3.52	56.33	29.59	55.61	35.22	52.26	27.25
R5	60	2	69.22	3.78	67.47	31.56	66.87	37.74	61.79	29.13
R6	65	2	79.58	4.13	77.97	33.69	77.37	39.87	71.45	30.62
R7	70	2	88.18	4.38	86.41	35.96	85.81	42.45	78.29	32.55
R8	75	2	95.31	4.67	93.40	37.56	92.62	44.95	84.26	34.46
R9	80	2	108.15	4.95	104.16	39.01	103.38	47.65	95.08	36.53
R10	85	2	117.20	5.21	113.39	41.79	111.69	50.04	102.36	38.33
R11	40	4	32.85	3.94	32.04	28.46	31.78	31.79	30.71	24.34
R12	45	4	37.81	4.41	37.15	31.53	36.87	34.55	34.95	26.44
R13	50	4	42.93	4.78	41.99	33.38	41.73	37.43	39.30	28.62
R14	55	4	48.70	5.20	47.84	35.72	47.46	40.37	44.55	30.84
R15	60	4	57.71	5.61	56.67	39.06	56.23	42.96	52.06	32.76
R16	65	4	64.85	6.00	63.88	40.12	63.40	45.95	58.76	34.96
R17	70	4	70.05	6.46	69.15	42.99	68.68	49.09	64.19	37.24
R18	75	4	74.47	6.82	73.53	45.39	73.06	51.34	67.36	38.92
R19	80	4	81.36	7.27	80.42	48.70	79.68	54.19	74.25	41.07
R20	85	4	86.89	7.74	85.30	51.68	84.03	57.12	79.09	43.17
R21	40	6	33.36	5.18	33.01	33.77	32.53	36.30	30.97	27.34
R22	45	6	38.38	5.73	37.95	36.66	37.51	39.36	35.41	29.62
R23	50	6	43.49	6.28	42.91	39.32	42.41	42.12	39.42	31.69
R24	55	6	49.50	6.80	48.26	42.21	47.91	45.79	45.06	34.20
R25	60	6	58.11	7.34	56.86	44.84	56.49	48.23	52.44	36.01
R26	65	6	65.20	7.85	64.04	47.73	63.72	51.30	59.19	37.84
R27	70	6	70.84	8.33	69.53	50.44	69.26	54.67	64.25	40.29
R28	75	6	73.90	8.97	73.37	53.97	73.05	57.26	67.57	42.03
R29	80	6	80.38	9.50	79.92	57.16	79.51	60.88	73.24	44.42
R30	85	6	85.17	10.01	84.27	61.68	83.78	63.69	78.41	46.44
Average:			64.69	5.78	63.51	39.72	62.95	44.42	58.52	33.59

MCT; (3) the MCT berthing capacity; (4) the F -statistic for each ANOVA test; and (5) the p value for each ANOVA test.

Note that F -statistic of each ANOVA test corresponds to the ratio of the mean squared errors that quantify the differences between the objective function values, which were recorded for the considered solution algorithms. On the other hand, p -value corresponds the probability of the test statistic having a value greater than the value of the calculated test statistic. Fairly low p -values were observed for all the conducted ANOVA tests, which indicate that the objective function values, recorded for the developed AIEA algorithm over 5 replications, were statistically different from (i.e., were statistically lower than) the objective function values,

recorded for the other considered solution algorithms over 5 replications. The maximum p -value did not exceed $2.91\text{E} - 06$, $3.79\text{E} - 04$, and $9.99\text{E} - 04$ over the generated realistic-size problem instances for the ANOVA-1, ANOVA-2, and ANOVA-3 tests, respectively. Therefore, the null hypothesis was rejected for each one of the conducted ANOVA tests.

6.4.2 Convergence patterns

A detailed analysis of convergence patterns for the developed algorithms was performed throughout the numerical experiments. The convergence patterns were recorded during execution of the AIEA, EA, TS, and SA algorithms, and the

Table 5 The ANOVA analysis results for the conducted statistical tests

Instance	#Vessels	#Berths	ANOVA-1		ANOVA-2		ANOVA-3	
			<i>F</i> -statistic	<i>p</i> -value	<i>F</i> -statistic	<i>p</i> -value	<i>F</i> -statistic	<i>p</i> -value
R1	40	2	913.10	1.56E – 09	266.87	1.99E – 07	63.83	4.41E – 05
R2	45	2	315.30	1.04E – 07	125.28	3.64E – 06	58.88	5.89E – 05
R3	50	2	798.86	2.65E – 09	675.50	5.16E – 09	168.77	1.17E – 06
R4	55	2	1436.31	2.58E – 10	501.76	1.67E – 08	203.85	5.65E – 07
R5	60	2	4574.10	2.54E – 12	1777.12	1.10E – 10	135.12	2.73E – 06
R6	65	2	351.03	6.80E – 08	221.94	4.06E – 07	213.71	4.70E – 07
R7	70	2	1149.03	6.27E – 10	572.11	9.94E – 09	439.75	2.81E – 08
R8	75	2	572.70	9.90E – 09	379.69	5.00E – 08	121.91	4.03E – 06
R9	80	2	752.85	3.36E – 09	484.74	1.91E – 08	234.76	3.27E – 07
R10	85	2	207.35	5.29E – 07	191.84	7.14E – 07	171.52	1.10E – 06
R11	40	4	132.89	2.91E – 06	34.33	3.79E – 04	25.42	9.99E – 04
R12	45	4	519.99	1.45E – 08	357.02	6.37E – 08	321.02	9.65E – 08
R13	50	4	249.28	2.59E – 07	145.08	2.08E – 06	95.19	1.02E – 05
R14	55	4	323.48	9.37E – 08	283.80	1.56E – 07	247.23	2.67E – 07
R15	60	4	2350.62	3.62E – 11	1740.58	1.20E – 10	401.94	4.00E – 08
R16	65	4	2039.31	6.39E – 11	550.65	1.16E – 08	88.14	1.36E – 05
R17	70	4	480.06	1.99E – 08	128.72	3.28E – 06	123.34	3.86E – 06
R18	75	4	1543.76	1.94E – 10	673.88	5.21E – 09	476.69	2.04E – 08
R19	80	4	276.05	1.74E – 07	117.78	4.59E – 06	96.82	9.57E – 06
R20	85	4	1320.68	3.60E – 10	116.50	4.79E – 06	112.91	5.38E – 06
R21	40	6	207.94	5.23E – 07	175.68	1.00E – 06	60.70	5.28E – 05
R22	45	6	276.37	1.73E – 07	240.06	3.00E – 07	93.42	1.09E – 05
R23	50	6	444.56	2.69E – 08	423.29	3.26E – 08	149.58	1.85E – 06
R24	55	6	307.65	1.14E – 07	120.09	4.27E – 06	30.91	5.35E – 04
R25	60	6	427.97	3.12E – 08	305.56	1.17E – 07	295.15	1.34E – 07
R26	65	6	403.00	3.96E – 08	364.92	5.84E – 08	176.22	9.90E – 07
R27	70	6	998.61	1.09E – 09	353.47	6.62E – 08	265.94	2.01E – 07
R28	75	6	5146.94	1.59E – 12	678.15	5.08E – 09	608.56	7.79E – 09
R29	80	6	4929.51	1.89E – 12	3165.87	1.10E – 11	849.57	2.08E – 09
R30	85	6	720.82	3.99E – 09	295.36	1.34E – 07	224.41	3.89E – 07
Average:			1139.00	1.70E – 07	515.59	1.35E – 05	218.51	5.86E – 05

results are shown in Fig. 7 for realistic-size problem instances R16–R30 (i.e., the realistic-size problem instances with the largest number of vessels calling for service at the MCT and the largest number of the available berthing positions). Note that the convergence patterns are presented only for the first replication of each problem instance, as they did not vary substantially from one replication to the other for the considered solution algorithms.

The results of the convergence pattern analysis demonstrate that the considered solution algorithms return the berth schedules with a lower total weighted service cost of vessels as compared to the initial berth schedules (which are based on the FCFS policy), especially for the problem instances with lower MCT berthing capacity. Moreover, execution of

separate EAs in parallel on multiple islands and the proposed adaptive mechanism for the exchange of individuals between the islands allow AIEA exploring the search space more efficiently and identifying the promising domains of the search space much faster as compared to the EA, TS, and SA algorithms. Furthermore, the findings indicate that explorative and exploitative capabilities of population-based metaheuristics (i.e., the AIEA and EA algorithms) are more efficient as compared to single solution-based metaheuristics (i.e., the TS and SA algorithms). The worst performance was recorded for SA, as it works with only two solutions at each iteration, which significantly limits the ability of the algorithm to discover promising domains of the search space and typically causes the premature convergence.

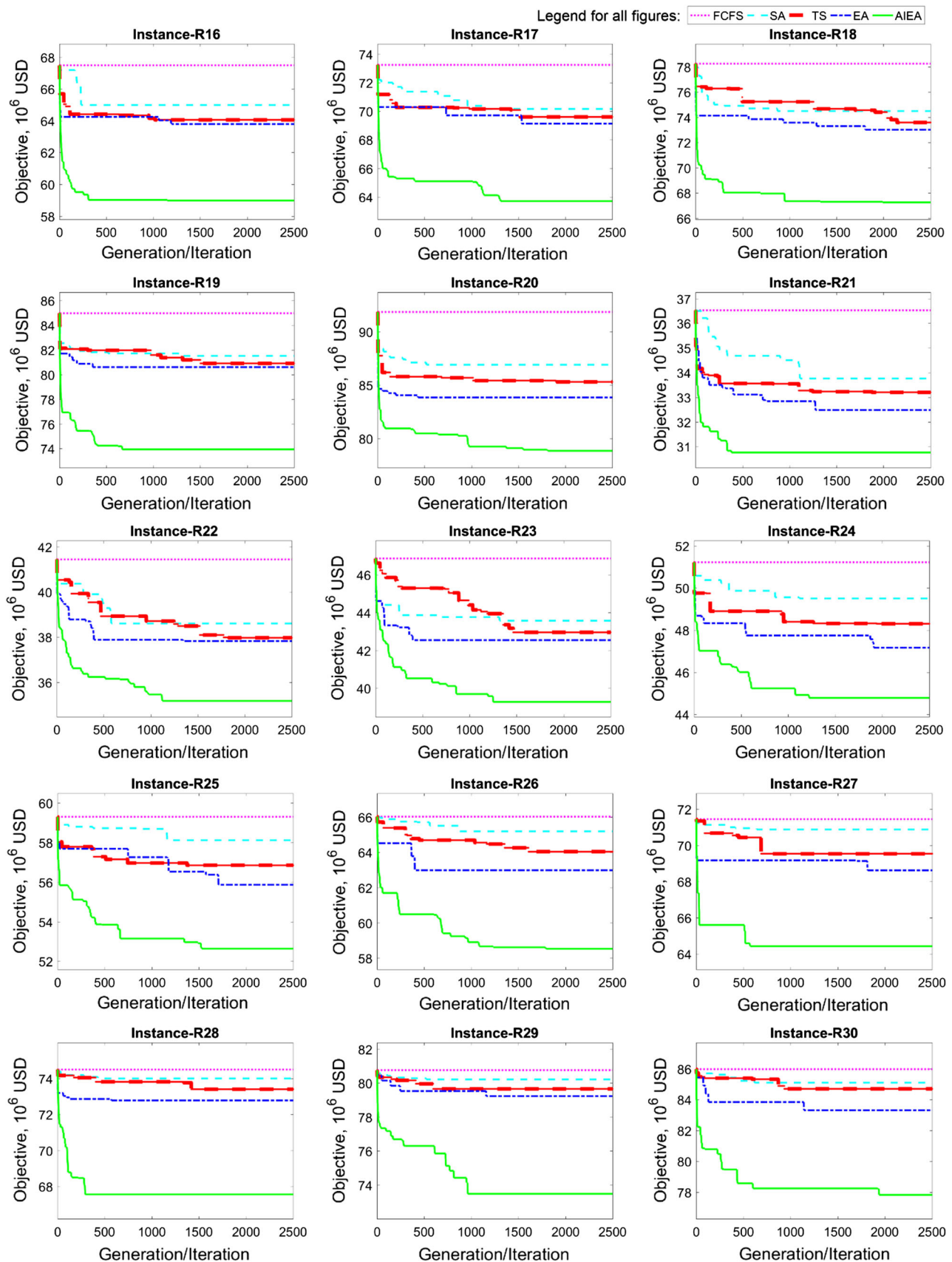
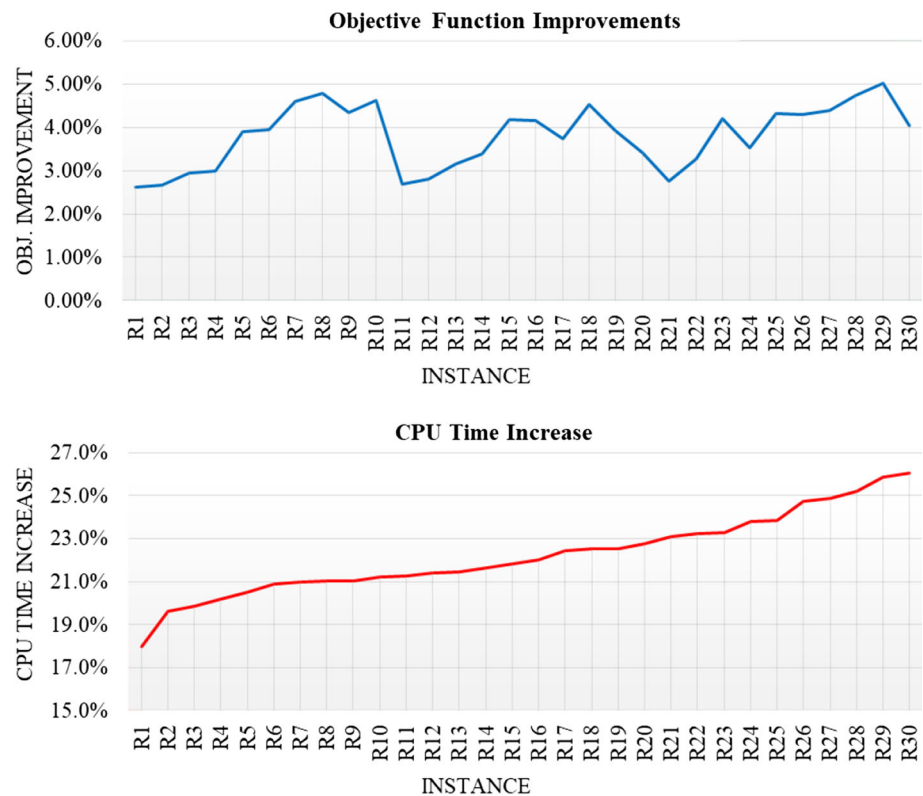


Fig. 7 Convergence patterns of the considered solution algorithms for instances R16–R30

Fig. 8 The objective function improvements and CPU time increase from deployment of the proposed adaptive exchange mechanism



6.5 Evaluation of the proposed adaptive exchange mechanism

The scope of computational experiments also included an evaluation of the proposed adaptive mechanism for the exchange of individuals between the AIEA islands. Two variations of the AIEA algorithm were considered, including the following: (1) AIEA-1 that simultaneously executes separate EAs in parallel on its islands and exchanges individuals between the islands based on the proposed adaptive mechanism; and (2) AIEA-2 that simultaneously executes separate EAs in parallel on its islands but does not perform the exchange of individuals between the islands based on the proposed adaptive mechanism. Both AIEA-1 and AIEA-2 were evaluated using the realistic-size problem instances (R1–R30). The results of the analysis are provided in Fig. 8, which presents the following data: (1) the average objective function improvements (over 5 replications) of AIEA-1 over AIEA-2; and (2) the average CPU time increase (over 5 replications) of AIEA-1 over AIEA-2. It can be observed that application of the proposed adaptive exchange mechanism allowed achieving up to 5.01% improvements in terms of the objective function values. Furthermore, the improvements in the objective function values increased with increasing number of vessels calling for service at the MCT and increasing number of the available berthing positions.

However, the additional exchange operations increase the time complexity of the AIEA-1 algorithm, when compared to the AIEA-2 algorithm that does not perform any exchange operations. An increase in the CPU time from application of the proposed adaptive exchange mechanism varied between 17.97 and 26.04% for the considered realistic-size problem instances (see Fig. 8). Nevertheless, as indicated in Sect. 6.4.1, the maximum CPU time of the AIEA-1 algorithm did not exceed 46.44 s, which can be considered as acceptable from the practical point of view (i.e., the MCT operator will be able to generate the berth schedules using the developed solution algorithm within a short span of time). Therefore, the conducted analysis demonstrates efficiency of the proposed adaptive exchange mechanism, as it allows achieving significant improvements in terms of the objective function values within a reasonable computational time.

7 Conclusions and future research directions

A significant increase in the seaborne containerized trade volumes has been observed over the last years. Long congestion periods have been reported at certain marine container terminals due to inability of the terminal infrastructure to handle the growing container volumes, increasing number of large-size vessels, port disruptions, and other factors. To alleviate potential congestion issues and avoid the cargo

delivery delays to the end customers, marine container terminal operators must improve the effectiveness of the terminal operations. The work, conducted as a part of this study, primarily focused on enhancing the seaside terminal operations and proposed a new Adaptive Island Evolutionary Algorithm to solve the berth scheduling problem, aiming to minimize the total weighted service cost of vessels. The developed algorithm simultaneously executed separate Evolutionary Algorithms in parallel on its islands and exchanged individuals between the islands based on a certain adaptive mechanism.

Performance of the proposed Adaptive Island Evolutionary Algorithm was evaluated based on a comprehensive comparative analysis against a typical Evolutionary Algorithm, Tabu Search, and Simulated Annealing. The latter metaheuristics have been widely used in the berth scheduling literature. The computational experiments demonstrated that the optimality gaps did not exceed 1.93% for the developed Adaptive Island Evolutionary Algorithm, which highlights its accuracy. Furthermore, the proposed solution algorithm outperformed the typical Evolutionary Algorithm, Tabu Search, and Simulated Annealing by 7.21%, 8.17%, and 10.11% in terms of the objective function values, respectively, for the realistic-size problem instances. A parallel execution of separate Evolutionary Algorithms on multiple islands allowed the Adaptive Island Evolutionary Algorithm outperforming the typical Evolutionary Algorithm in terms of the computational time on average by 32.23%. A detailed evaluation of the algorithmic convergence patterns demonstrated that a simultaneous execution of separate Evolutionary Algorithms in parallel on multiple islands and the exchange of individuals based on the adaptive mechanism allowed the Adaptive Island Evolutionary Algorithm exploring the search space more efficiently and identifying the promising domains of the search space much faster as compared to the other considered metaheuristics. Hence, the developed solution algorithm can serve as an efficient practical tool for marine container terminal operators and may assist with the design of cost-effective berth schedules.

The scope of future research includes, but is not limited to: (1) develop an alternative mutation operator (e.g., adaptive or self-adaptive); (2) design additional heuristics to improve performance of parallel Evolutionary Algorithms; (3) evaluate alternative convergence criteria for the algorithm; (4) develop an alternative heuristic for the chromosome initialization; (5) develop additional control strategies for the migration interval and the migration size parameters and evaluate how these control strategies would affect the overall performance of the proposed Adaptive Island Evolutionary Algorithm; (6) evaluate alternative rules for the exchange of individuals between islands of the algorithm; (7) consider additional spatial requirements in berth scheduling (e.g., vessel draft); (8) evaluate the overall performance of

the proposed Adaptive Island Evolutionary Algorithm for multi-objective mathematical models; and (9) assesses performance of the developed solution algorithms against the other algorithms, reported in the berth scheduling literature (e.g., Ant Colony Optimization, Variable Neighborhood Search, Bee Colony Optimization, Stochastic Beam Search).

References

1. UNCTAD (2017) Review of maritime transport 2017. United Nations Conference on Trade and Development, New York
2. Journal of Commerce (2015) Largest container ships on order to rise 13 percent by 2020. www.joc.com. Accessed 06 Jan 2018
3. Bierwirth C, Meisel F (2015) A follow-up survey of berth allocation and quay crane scheduling problems in container terminals. *Eur J Oper Res* 244(3):675–689
4. Dadashi A, Dulebenets MA, Golias MM, Sheikholeslami A (2017) A novel continuous berth scheduling model at multiple marine container terminals with tidal considerations. *Marit Bus Rev* 2(2):142–157
5. Golias M, Portal MI, Konur D, Kaisar E, Kolomvos G (2014) Robust berth scheduling at marine container terminals via hierarchical optimization. *Comput Oper Res* 41:412–422
6. Frojan P, Correcher J, Alvarez-Valdes R, Koulouris G, Tamarit J (2015) The continuous Berth Allocation Problem in a container terminal with multiple quays. *Expert Syst Appl* 42(21):7356–7366
7. Dulebenets MA (2017) A novel memetic algorithm with a deterministic parameter control for efficient berth scheduling at marine container terminals. *Marit Bus Rev* 2(4):302–330
8. Dulebenets MA (2018) Application of evolutionary computation for berth scheduling at marine container terminals: parameter tuning versus parameter control. *IEEE Trans Intell Transp Syst* 19(1):25–37
9. Dulebenets MA, Kavousi M, Abioye OF, Pasha J (2018) A self-adaptive evolutionary algorithm for the berth scheduling problem: towards efficient parameter control. *Algorithms* 11(7):1–35
10. Dulebenets MA, Moses R, Ozguven EE, Vanli A (2017) Minimizing carbon dioxide emissions due to container handling at marine container terminals via hybrid evolutionary algorithms. *IEEE Access* 5:8131–8147
11. Imai A, Nishimura E, Paradimitriou S (2008) Berthing ships at a multi-user container terminal with a limited quay capacity. *Transp Res Part E* 44(1):136–151
12. Dulebenets MA, Golias M, Mishra S (2018) A collaborative agreement for berth allocation under excessive demand. *Eng Appl Artif Intell* 69:76–92
13. Karafa J, Golias M, Ivey S, Saharidis G, Leonardos N (2013) The berth allocation problem with stochastic vessel handling times. *Int J Adv Manuf Technol* 65(1–4):473–484
14. Hu Z (2015) Multi-objective genetic algorithm for berth allocation problem considering daytime preference. *Comput Ind Eng* 89:2–14
15. Xu Y, Chen Q, Quan X (2011) Robust berth scheduling with uncertain vessel delay and handling time. *Ann Oper Res* 192(1):123–140
16. Emde S, Boysen N (2016) Berth allocation in container terminals that service feeder ships and deep-sea vessels. *J Oper Res Soc* 67(4):551–563
17. Lee D, Chen J, Cao J (2010) The continuous Berth Allocation Problem: a greedy randomized adaptive search solution. *Transp Res Part E* 46(6):1017–1029
18. Cordeau J, Laporte G, Legato P, Moccia L (2005) Models and Tabu Search heuristics for the berth allocation problem. *Transp Sci* 39(4):526–538

19. Lalla-Ruiz E, Melian-Batista B, Moreno-Vega J (2012) Artificial intelligence hybrid heuristic based on Tabu search for the dynamic berth allocation problem. *Eng Appl Artif Intell* 25(6):1132–1141
20. Imai A, Yamakawa Y, Huang K (2014) The strategic berth template problem. *Transp Res Part E* 72:77–100
21. Cheong C, Tan K (2008) A multi-objective multi-colony ant algorithm for solving the berth allocation problem. *Stud Comput Intell* 116(333–350):16
22. Hansen P, Oguz C, Mladenovic N (2008) Variable neighborhood search for minimum cost berth allocation. *Eur J Oper Res* 191(3):636–649
23. Mauri G, Ribeiro G, Lorena L, Laporte G (2016) An adaptive large neighborhood search for the discrete and continuous berth allocation problem. *Comput Oper Res* 70:140–154
24. Eiben AE, Smith JE (2015) *Introduction to evolutionary computing*. Springer, Berlin
25. Dulebenets MA (2018) Green vessel scheduling in liner shipping: modeling carbon dioxide emission costs in sea and at ports of call. *Int J Transp Sci Technol* 7(1):26–44
26. Dulebenets MA (2018) A comprehensive multi-objective optimization model for the vessel scheduling problem in liner shipping. *Int J Prod Econ* 196:293–318
27. Dulebenets MA (2019) Minimizing the total liner shipping route service costs via application of an efficient collaborative agreement. *IEEE Trans Intell Transp Syst* 20(1):123–136
28. Jakob W (2010) A general cost-benefit-based adaptation framework for multimeme algorithms. *Memet Comput* 2(3):201–218
29. Muthuswamy S, Lam S (2011) Discrete particle swarm optimization for the team orienteering problem. *Memet Comput* 3(4):287–303
30. LaTorre A, Muelas S, Peña J (2013) Evaluating the multiple offspring sampling framework on complex continuous optimization functions. *Memet Comput* 5(4):295–309

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.