FLORIDA STATE UNIVERSITY

FAMU-FSU COLLEGE OF ENGINEERING

INNOVATIVE METAHEURISTIC ALGORITHMS FOR EFFICIENT BERTH SCHEDULING AT MARINE CONTAINER TERMINALS

By

MASOUD KAVOOSI

A Dissertation submitted to the
Department of Civil and Environmental Engineering
in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

2019

Masoud Kavoosi defended this dissertation on [February 14, 2015].

The members of the supervisory committee were:

Maxim A. Dulebenets
Professor Directing Dissertation

O. Arda Vanli
University Representative

Ren Moses
Committee Member

Eren Ozguven
Committee Member

Hui Wang
Committee Member

The Graduate School has verified and approved the above-named committee members, and certifies that the dissertation has been approved in accordance with university requirements.

To my family, *Farahbakhsh*, *Azam*, *Atefe*, *Reza, and Bahar*.

ACKNOWLEDGMENTS

I would like to express my deepest appreciation to my advisor Dr. Maxim Dulebenets. I do believe that without the support and nurturing of Dr. Dulebenets this dissertation could have not been completed. Moreover, his invaluable support and patience made me to get adjusted much more comfortable to the new environment.

I would like to extend my sincere thanks to my committee members: Dr. Ren Moses, Dr. Eren Ozguven, Dr. Hui Wang, and Dr. Arda Vanli for their helpful comments, advices, and contributions during the completion of this dissertation. I would like to extend my sincere thanks to my dear friend Dr. Mohamad Aslani, who helped me a lot in different aspects during my Ph.D. program. I also had great pleasure of working with my colleagues, graduate research assistants Olumide Abioye and Junayed Pasha for their great collaborations.

I would also like to extend my deepest gratitude to my family, who always relentlessly support me to reach my goals.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

Maritime transportation has been continuously playing an undeniable role for the global trade and economy of many countries. Based on the fast growth of the maritime trade, the marine container terminal (MCT) operators should focus on improving the operations planning at the MCTs. Seaside operations have substantial impacts on the general throughput of the MCTs. The daily berth planning as a seaside operation is the point of focus herein. The daily berth planning is modeled as a berth scheduling problem (BSP) in this dissertation. The BSP (as a decision problem) aims to assign the arriving vessels to the available berthing positions and can be reduced to the unrelated machine scheduling problem, which has NP-hard complexity. The large-size instances of decision problems with NP-hard complexity cannot be solved using exact optimization algorithms, while metaheuristic algorithms can effectively solve large-size problem instances and return good-quality solutions.

Evolutionary Algorithms (EAs) are among the most popular metaheuristic algorithms deployed to solve the real-size BSPs. There are some algorithmic parameters in EAs (e.g., crossover probability, mutation probability, population size, etc.), which should be assigned the appropriate values to have the best possible performance of the algorithm for a given BSP. The process of determination of algorithmic parameters values is called the parameter selection. Several methodologies have been introduced in the EA literature for parameter selection, which can be classified as follows: (1) parameter tuning; and (2) parameter control. In parameter tuning, the algorithmic parameter values remain constant throughout the algorithmic evolution, while the parameter control strategy updates the algorithmic parameters considering different approaches.

In this dissertation, an EA with a self-adaptive parameter control strategy is proposed to solve the developed BSP. Based on a self-adaptive parameter control strategy, the crossover and mutation probabilities are encoded in the solutions and evolve with the EA. The problem is formulated as a mixed-integer linear programming model, minimizing the total weighted vessel turnaround time and the total weighted vessel late departures. Comprehensive numerical experiments are conducted to assess performance of the proposed self-adaptive EA against the alternative EAs, which rely on the different parameter selection strategies. Results demonstrate that all the considered solution algorithms show a promising performance in terms of the objective

function values at termination. However, application of the self-adaptive parameter control strategy substantially enhances the objective function values at convergence without a significant impact on the computational time.

Furthermore, an EA with an augmented self-adaptive parameter control strategy is presented in this dissertation as another solution algorithm for the BSPs. Based on an augmented self-adaptive parameter control strategy, not only the crossover and mutation probabilities are encoded in the solutions but they are also updated based on the feedback from the search. A mixed-integer linear programming mathematical model is developed for the BSP, aiming to minimize the total costs for serving vessels at the MCT. The designed algorithm is evaluated against nine alternative state-of-the-art metaheuristic algorithms, which have been widely utilized in the BSP literature. The results show that all the developed algorithms have a high level of stability and return high-quality solutions at termination. The computational experiments also prove the superiority of the designed augmented self-adaptive EA over the alternative algorithms considering different performance indicators.

Another innovative solution methodology is developed in this dissertation, which relies on the island-based concept. Specifically, a universal island-based metaheuristic algorithm is designed for the BSP, where four different population-based metaheuristics are executed simultaneously in order to effectively search for solutions. A mixed-integer linear mathematical model is developed for the BSP, minimizing the total cost to serve the arriving vessels at the MCT. Comprehensive numerical experiments are conducted to evaluate performance of the island-based algorithm against seven commonly used metaheuristics in the BSP literature. The stability and the capability of the adopted algorithms in providing high-quality solutions at convergence are proven. The results demonstrate that the island-based algorithm outperforms other adopted algorithms considering different performance indicators.

To summarize, this dissertation proposes three different solution methodologies for various BSP mathematical formulations. The algorithms have been evaluated based on extensive numerical experiments against the alternative algorithms, which have been widely used in the MCT and freight terminal operations literature. Findings confirm effectiveness of the proposed

solution methodologies. Therefore, the developed solution methodologies can serve as promising decision support tools and assist MCT operators with the development of berth schedules. The latter will also assist with serving the growing demand for containerized trade and ensure that the vessel service will be completed in a timely manner.

CHAPTER 1

INTRODUCTION

The statistical data, collected for the international trade, indicates that waterborne transport has been a predominant transportation mode for many years. Approximately 80 percent of international trade by volume and 70 percent of that by value are the contributions of waterborne transport (UNCTAD, 2017). Based on the data released by United Nations Conference on Trade and Development (UNCTAD), an increase of 3 percent per year has been recorded for the waterborne trade volumes during the last four decades. Projections for 2017 to 2022 show that the waterborne trade will be increasing by 3.2 percent growth annually. The total volume of 10.6 billion tons of goods was transported by oceangoing vessels in 2017, which corresponds to an increase of 2.8 percent as compared to 2016. Figure 1 illustrates the general changes in four important indices including: 1 - world merchandise trade, 2 - world seaborne trade, 3 - world gross domestic product, and 4 - Organization for Economic Cooperation and Development (OECD) index of industrial production, which reflect the international trade tendencies from 1975 to 2016. According to the graph, throughout the considered time horizon all the four indices have grown with almost the same behavior while their progress speeds (i.e., the slopes of the diagram) are different. After 1990, the OECD index of industrial production has grown slowly and even in the last ten years have had some fluctuations, while the world seaborne trade and the world merchandise trade have shown a growing behavior. More information regarding the waterborne trade tendencies is provided next.

The goods, transported by vessels, can be categorized in two major groups: (a) bulk cargo and (b) containerized cargo. The bulk cargos are commonly classified into three subgroups: 1 - oil and gas, 2 - five major bulks (which include iron ore, grain, coal, bauxite, alumina and phosphate rock), and 3 - other dry cargo. According to Figure 2, the waterborne trade tendencies for container and five major bulks are almost the same, where they show a constant increase during the time period considered (i.e., from 1980 to 2016). In contrast, oil and gas and other dry cargo have fluctuated significantly more than the other two cargo types, and the growth pattern has been substantially slower for them as well (UNCTAD, 2017).

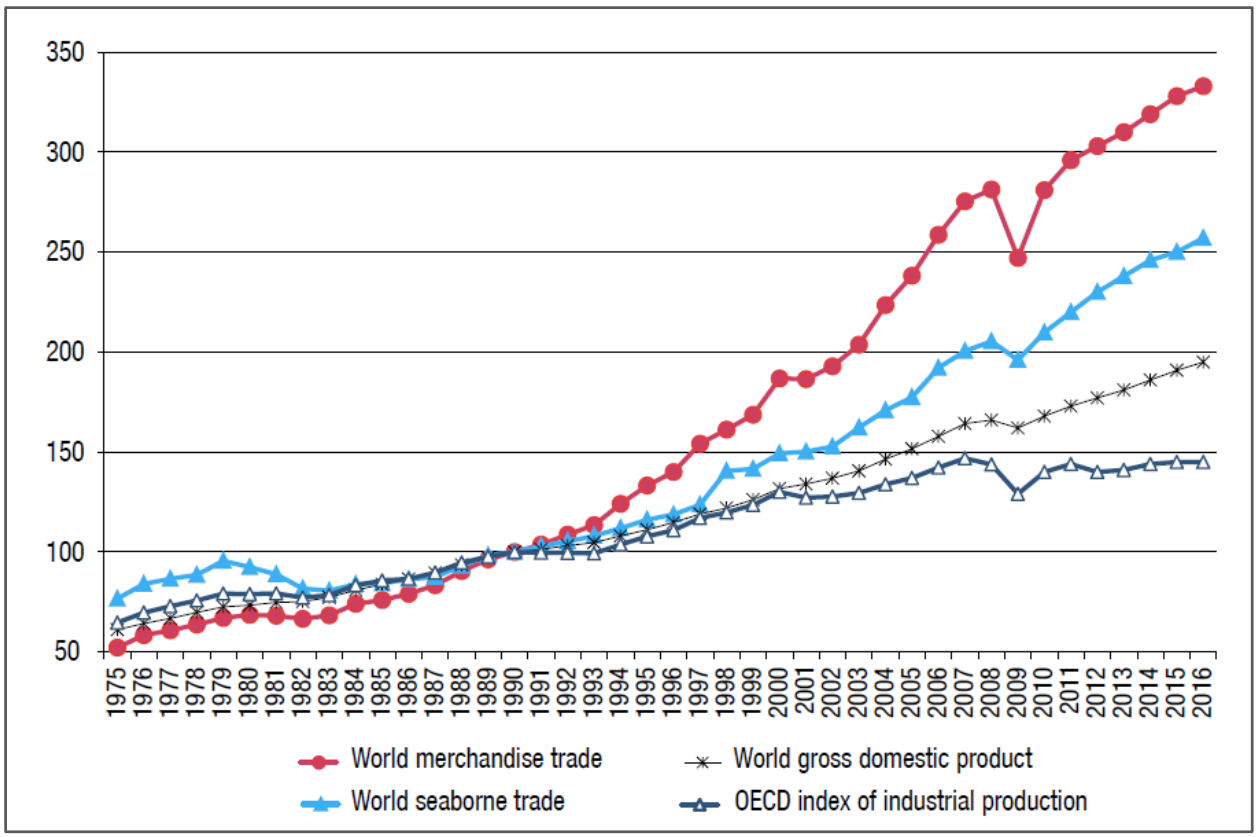

Figure 1 Values of the Organization for Economic Cooperation and Development (OECD) index of industrial production in comparison with world gross domestic product, world merchandise trade and world seaborne trade between 1975 and 2016.
Source: UNCTAD, 2017

Figure 3 shows the share of each cargo group for the waterborne transportation from 1980 to 2016. It has been published by UNCTAD and indicates that the total container and oil and gas shipment volumes have increased by 3.55% and 4.20% respectively from 2015 to 2016. On the other side, five major bulks and other dry cargo have experienced a total growth of 1.63% and 1.26%, respectively, throughout the same time period (UNCTAD, 2017). Based on the Figure 3, gas and oil have had the largest market share for about 30 years, while the marker share of five major bulks has grown from one year to another. In 2016, the marker share of five major bulks was even greater as compared to the marker share of oil and gas. The growth rate of containerized trade was found to be higher than other cargo types, while the amount of other dry cargo transported has almost remained constant after 2006 (UNCTAD, 2017).

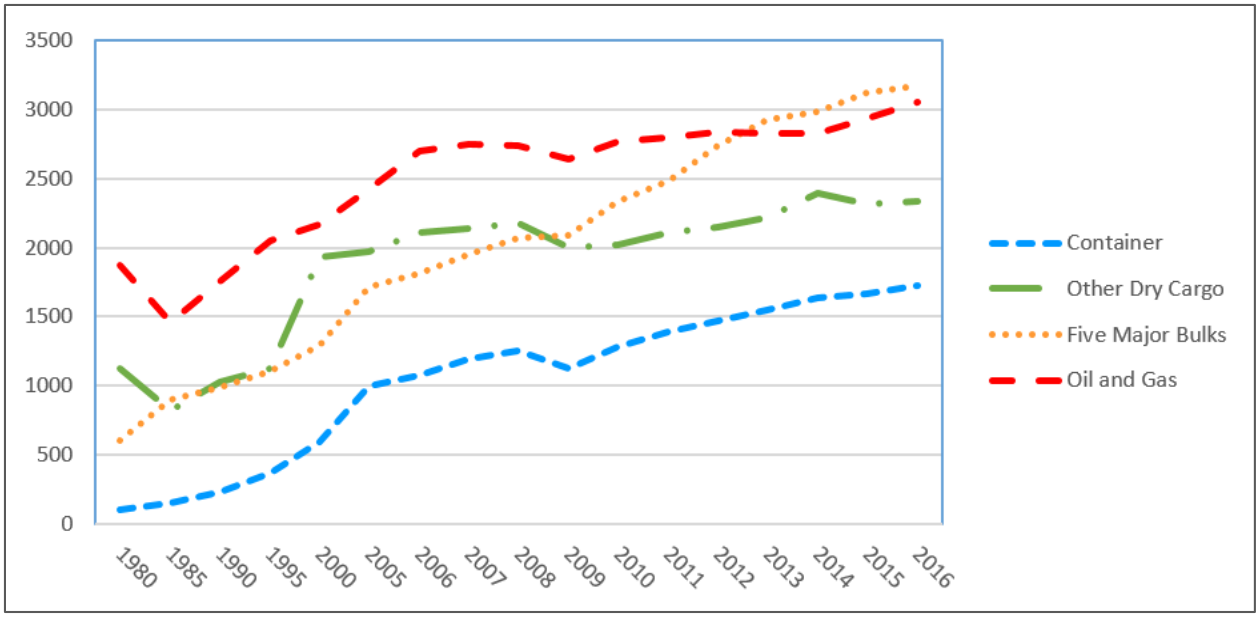

Figure 2 The general growth pattern for four different groups of cargo carried by vessel fleet (1980-2016).
Source: UNCTAD, 2017

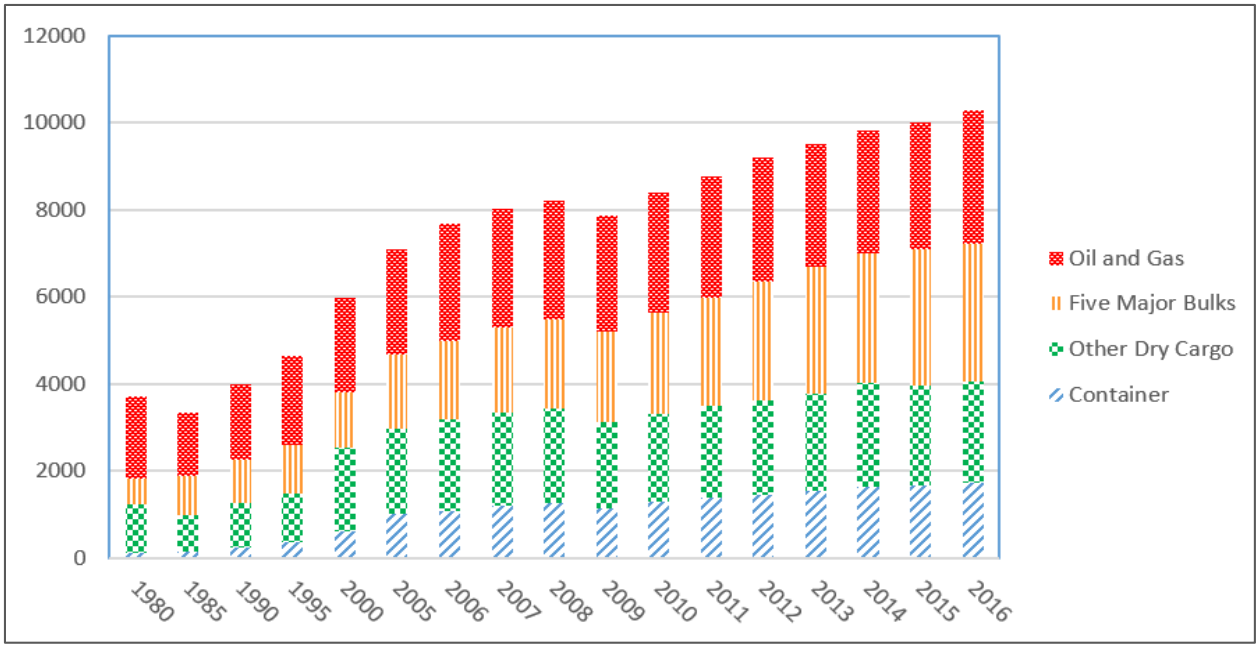

Figure 3 International waterborne trade (millions of tons loaded).
Source: UNCTAD, 2017

A significant increase in container volumes, which are handled at ports, has been observed over the last year. Table 1 shows the top 10 busiest container ports around the world, which are

ranked based on the number of handled Twenty-foot Equivalent Units (TEUs) in a year. UNCTAD reported the Port of Shanghai (China) to be the busiest container port in the world in 2016 with 37.14 million TEUs and 1.6% increase in comparison with 2015. Although the average container volumes have expanded in 2016 as compared with 2015, four of the busiest ports have experienced a reduction in their business. The Port of Dubai faced a 5.3% decrease (the largest reduction among the other top 10 ports), while the Port of Guangzhou experienced the highest increase among the top 10 ports by 8.0%. Based on the analysis of the collected data, all the top 10 ports are located in Asia. The Port of Rotterdam, located in the Netherlands, was ranked as the first port in Europe with 12.39 million TEUs (and the 12th port in the world), whereas the Port of Los Angeles was ranked as the first port in Americas (and the 18th port in the world). Furthermore, the Port of Piraeus (Greece) with 14.1% growth in 2016 (ranked 36th in the world) had the largest increase among the top 40 ports in the world. On the other hand, the Port of Tanjung Pelepas (Malaysia) experienced the largest reduction with -8.8% among the top 40 ports in the world and is ranked as the 19th among the top 40 container ports.

Seaports, as critical nodes in supply chain networks, have to cope with the growing demand for the international containerized trade. Considering the limitations that seaports may face for expanding the cargo throughput (e.g., lack of land, expensive equipment, capacity constraints, etc.), efficient management and exploitation of the existing resources are critical for the marine container terminal (MCT) operators. The number of shipping containers is forecasted to increase with 2-3 million TEUs in some cases (e.g., the Chinese special economic zone - Shenzhen). The larger container vessels have been designed in the recent years, which may lead to new challenges for the ports (Mourão, Pato, & Paixão, 2002). The Panama Canal will not be able to serve the larger vessel fleet (18,000 TEUs), which are deployed by Maersk at the Asia-Europe trade routes. Furthermore, the second Panama Canal expansion (which may be considered as necessary in order to serve larger vessels) is practically infeasible (Ashar, 2013). In order to meet the growing demand for the waterborne trade, along with construction of new ports (McCalla, 1999), the port operations efficiency can be increased through: 1 - new technology deployment for enhancing the operational efficiency of the conventional equipment (Ballis, Golias, & Abarkoumkin, 1997), 2 - deployment of innovative information systems (Henesey, 2004), and 3 - work organization improvement (Paixao & Bernard Marlow, 2003).

Table 1 The top 10 world seaports.

Ranking	Port	Country	Volume (million TEUs)		2015–2016 (Percentage change)
			2016	2015	
1	Shanghai	China	37.14	36.54	**1.6**
2	Singapore	Singapore	30.93	30.96	**-0.1**
3	Shenzhen	China	23.98	24.20	**-0.9**
4	Ningbo	China	21.57	20.59	**4.7**
5	Hong Kong	Hong Kong (China)	19.58	20.11	**-2.7**
6	Busan	Republic of Korea	19.38	19.30	**0.4**
7	Guangzhou	China	18.86	17.46	**8.0**
8	Qingdao	China	18.05	17.47	**3.3**
9	Dubai	United Arab Emirates	14.77	15.59	**-5.3**
10	Tianjin	China	14.52	14.11	**2.9**

Source: UNCTAD, 2017

The remainder of this dissertation is structured as follows. The routine operations at the MCTs are comprehensively described in Chapter 2. The berth scheduling problem is discussed in detail in Chapter 3. Chapter 4 provides an extensive review on the BSP literature, where some statistical analysis is conducted over the investigated studies, and also contributions of this dissertation as compared to the published studies are highlighted in this chapter. The first developed solution methodology, which is called an Evolutionary Algorithm with self-adaptive parameter control, is provided and evaluated in Chapter 5. The second solution algorithm, referred to as an Evolutionary Algorithm with augmented self-adaptive parameter control, is proposed and evaluated in Chapter 6. Chapter 7 is allocated to present a Universal Island EA-PSO-EDA-DE Metaheuristic for the berth scheduling problem, as the last solution algorithm developed in this dissertation. Chapter 8 provides a general conclusion on the whole dissertation, and also the future research directions are highlighted in this chapter as well.

CHAPTER 2

OVERVIEW OF THE MCT OPERATIONS

MCTs are critical nodes in supply chain networks, where containers are delivered by vessels of different liner shipping companies and then being handled by the appropriate types of equipment, owned or leased by the MCT operators (Norstad, Fagerholt, & Laporte, 2011; Wang, Alharbi, & Davy, 2014). The ongoing operations, which take place at MCTs, can be divided into the following three major categories:

1- seaside operations,
2- marshaling yard operations,
3- landside operations.

Seaside operations include mooring of the arriving vessels at the preferred berthing positions, allocation of quay cranes (QCs) among the available berthing positions, and (un)loading of the containers by QCs. On the other hand, marshaling yard operations focus on storage of the inbound containers (i.e., the containers that are delivered by the vessels and then stored in the marshaling yard) and outbound containers (i.e., the containers that have been already stored and are going to be transported to the other ports by vessels) in the yard blocks using the yard cranes (YCs), which are also referred to as gantry cranes (GCs). There are different types of vehicles, so-called internal transport vehicles (ITVs), which are used for the transfer of containers between the seaside and the marshaling yard of the MCT. Yard trucks, straddle carriers, automated guided vehicles, and automated lifting vehicles are the most common ITVs used at MCTs. Landside operations incorporate the receipt of containers and/or delivery by the outbound trucks (OTs), which can enter the MCT via the assigned gates. In general, the container flows at MCTs can be conventionally divided into three groups: 1 - from the storage yard to the vessel (export), 2 – from the vessel to the storage yard (import), and 3 – from one vessel to another vessel (transshipment).

All the general container flows can be briefly explained as follows: when an arriving vessel is moored at the assigned berthing position, the QCs start (un)loading (import/export) containers

from/to the vessels. In the meantime, the ITVs are called to pick up/drop off the containers from/to the seaside and deliver them to/from the storage area (marshaling yard), where the GCs are in charge of arranging the containers in the yard blocks parallel or perpendicular to the wharf. Moreover, OTs, which carry the export containers, are assigned to pre-determined gates to deliver the containers to the GCs in the specific yard blocks. In case of transshipment, a series of the containers should be picked up from a vessel (so-called the mother vessel) and be dropped off on the other vessel (so-called the feeder vessel) with or without temporary storage in the marshaling yard. Figure 4 illustrates the flowchart for the major MCT operations.

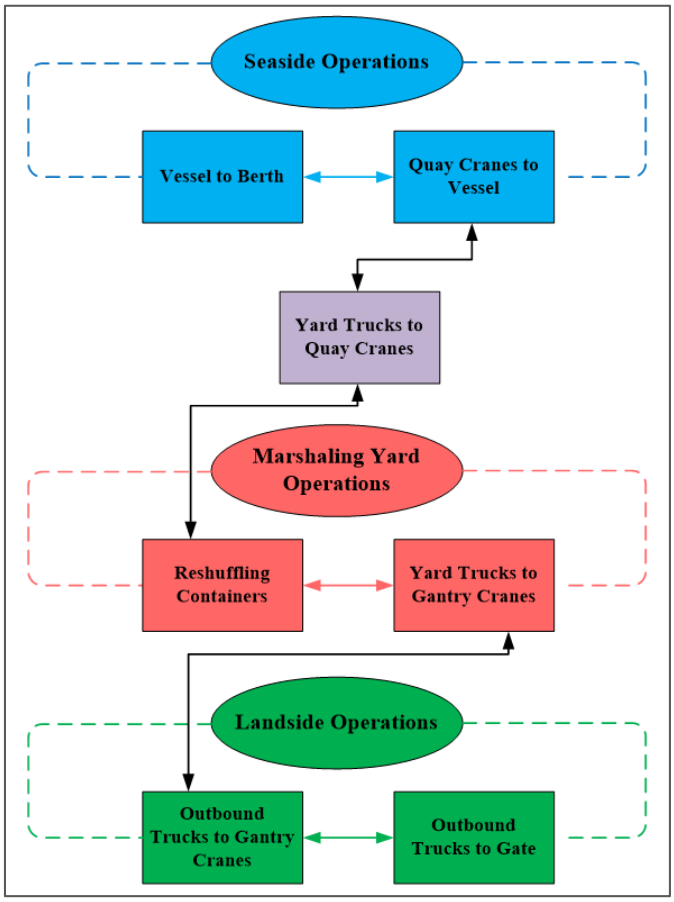

Figure 4 A flowchart of major MCT operations.

In order to manage the MCTs efficiently, the MCT operators have to solve many decision problems on a daily basis. Allocating an appropriate berthing position to a vessel considering spatial requirements and the number of containers that should be transferred is a challenging decision problem for the MCT operators. Assignment of quay cranes for a vessel is another

decision problem that should be addressed by the MCT operators. Assigning a sufficient number of ITVs (such as yard trucks, straddle cranes, automated guided vehicles) should be done based on the available resources, assigning an arrival time window for a given vessel, and many other factors throughout vessel service at the seaside have to be tackled by the MCT operators as well. Along with all the aforementioned issues, identification of an appropriate location in the yard blocks for storing containers is one of the most important decisions that can remarkably affect the process of serving a vessel.

Besides, there are some other operational aspects that should be considered by the MCT operators throughout the operations planning, for example: 1 – relocation of gantry cranes between different yard blocks, 2 - taking into account both the capacity of the yard block and gantry crane and safety issues, 3 - assigning a suitable gate to an outbound truck considering the storage area in the yard block, where the containers should be unloaded. All of these decision problems have attracted a lot of attention from the scientists for several decades. One of the most critical decision problems at MCTs is assigning berthing positions to the arrival vessels, as it directly affects the vessel service time (or vessel turnaround dime, which is a summation of the vessel waiting time and the vessel handling time). The latter decision problem (commonly referred to as the berth scheduling problem) will be the major focus of this dissertation.

CHAPTER 3

BERTH SCHEDULING PROBLEM

Once a vessel arrives at the MCT, the first decision problem is to assign that vessel to one of the available berthing positions. The berth scheduling problem (BSP) aims to specify which arriving vessel should be moored at which berthing position considering all the existing physical and operational constraints. BSPs can be classified based on different aspects as follows (Bierwirth & Meisel, 2010, 2015):

- Classification based on the spatial attributes,
- Classification based on the vessel arrivals,
- Classification based on the vessel handling times, and
- Classification based on the objective functions considered.

Based on the first classification, the BSPs can be divided into discrete, continuous, hybrid, and draft consideration. In a discrete BSP, the wharf is divided into several berthing positions, where only one vessel can be served at each berthing position at the time (Figure 5-a&b). In a continuous BSP, the wharf length is limited but is not divided into a specific number of berthing positions, and several vessels can be served along the length of the wharf considering the safety space requirements between them (Figure 5-c). A hybrid BSP is a combination of discrete and continuous BSPs, where the wharf is divided into several berthing positions, but a large vessel can occupy more than one berthing position and several small vessels can be served next to each other in a berthing position (Figure 5-d). An indented berthing layout is another type of berthing layout considered in the literature, mostly used for serving megaships and is classified as a hybrid berthing layout (Figure 5-e) (Carlo, Vis, & Roodbergen, 2015). Along with the aforementioned berthing layouts, some studies also considered the draft of the arriving vessels as a constraint. Specifically, the vessels, which have a draft larger than the depth of a given berthing position, should be moored at another berthing position to make sure that the depth will be sufficient to navigate and be moored (Figure 5-f).

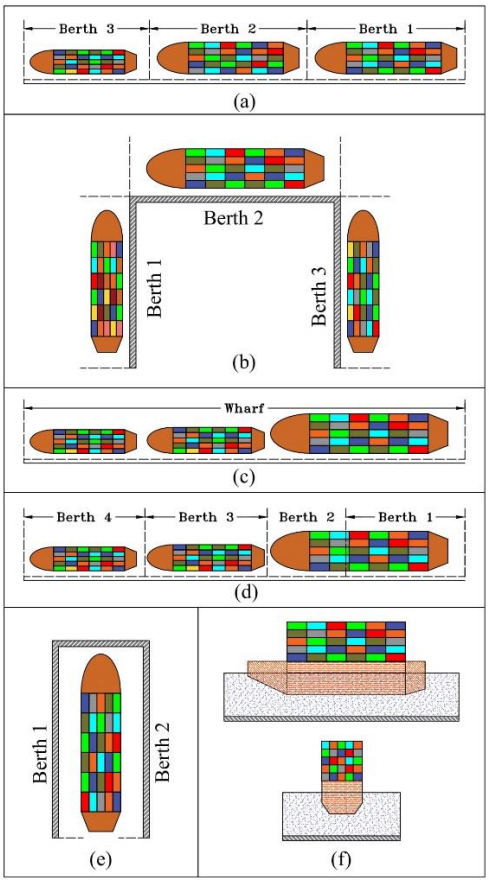

Figure 5 Spatial attribute of the BSP.

As for the vessel arrivals, the BSP can be classified as static and dynamic. A static vessel arrival means that all the vessels have already arrived at the MCT, whereas the dynamic vessel arrival means that vessels are expected to arrive at the terminal at a certain time period in the considered planning horizon (generally negotiated between the MCT operator and the liner shipping company). Furthermore, BSPs can be classified based on the handling time of the vessels. If the number of the assigned QCs to a vessel and their corresponding handling productivity do not change throughout the vessel service, the handling time is constant. However, the latter condition would rarely happen. Variable handling time could be a function of the berthing position (mooring at the "preferred berthing position" would lead to the maximum QC productivity, and the "preferred berthing position" is typically allocated closer to the dedicated storage space, where the containers will be moved from the vessel, which will reduce the total ITV travel time and increase the number of container moves over the considered time period), handling volumes, the number of the assigned QCs to each vessel. Moreover, the vessel handling time can be treated as

a stochastic variable in order to accommodate uncertainties throughout the container handling process (Bierwirth & Meisel, 2010, 2015). BSPs can be also classified based on the objective functions considered (e.g., minimize the total vessel turnaround time, minimize the total vessel handling time, minimize the total vessel waiting time, minimize the total vessel late departure time, minimize the total vessel service cost, etc.).

This dissertation specifically focuses on the discrete BSPs (DBSPs). Figure 6 illustrates a distribution of the DBSP papers, which have been identified from the literature search. A total of 54 papers were collected for the time period between 1997 to 2018. After 2010 and especially in the last three years, DBSPs have received a lot of attention from the scientific community. The following chapter of the dissertation provides a detailed review of the collected DBSP studies.

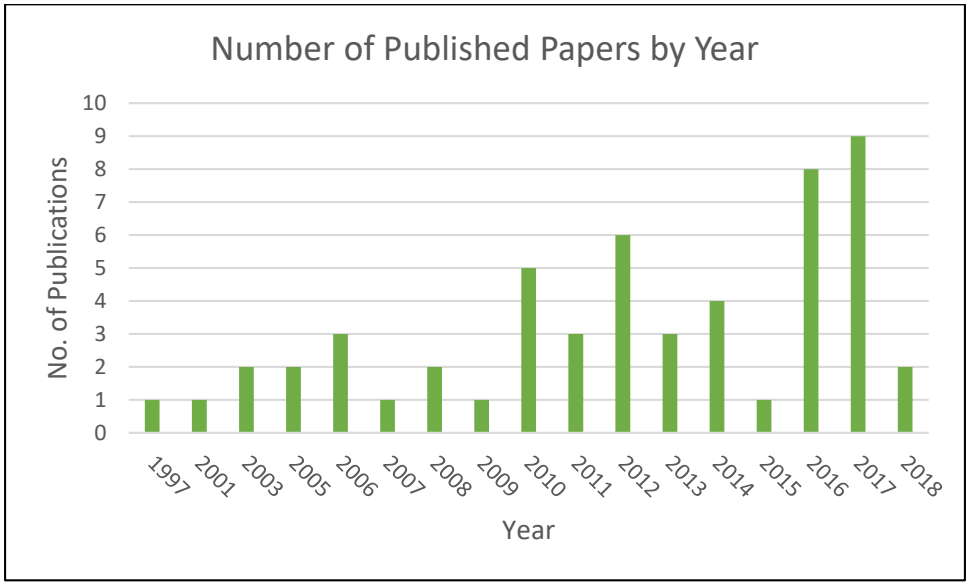

Figure 6 Distribution of the DBSP papers by year of publication.

CHAPTER 4

DBSP LITERATURE REVIEW

4.1. Overview of the Collected Studies

Brown et al. (1997) developed two models for optimizing nuclear submarine berthing in a port. The first model produced the berthing plan for the arriving submarines, while the second one managed the changes in case of the first plan approval. The paper aimed to maximize the total benefits earned from lower penalization due to berthing position shifts and failing to provide the requested service. The developed mathematical model was solved using CPLEX. The conducted numerical experiments, which relied on the data collected from Naval Submarine Base in San Diego, demonstrated efficiency of the proposed methodology.

Imai et al. (2001) investigated a dynamic berth allocation planning problem in the public berth system. Minimization of the total waiting time and handling time was considered as an objective function in that study. Along with adopting a heuristic method for solving the problem, a Lagrangian relaxation method was applied to the original problem to decrease the computational cost. A large number of numerical experiments showed efficiency of the proposed methodology for the real-world BSP problems.

Hansen et al. (2003) presented the models for both static and dynamic BSPs. The total handling and waiting times for all calling vessels were minimized in that study. A formulation was designed for the static BSP and then was modified to capture the dynamic BSP features. CPLEX was adopted to solve realistic problems with up to 10 berths and 50 vessels calling for service at the MCT. CPLEX could return high-quality solutions for the static BSP, while it did not perform well in case of a dynamic BSP.

Imai et al. (2003) proposed a BSP mathematical model considering the service priority of vessels at a Multi-User Container Terminal (MUT). A Genetic Algorithm (GA)-based heuristic was developed to minimize the summation of waiting and handling time of all vessels. A series of numerical experiments were implemented to assess performance of the proposed solution

approach. It was concluded that service priority could provide the MUT operators with more alternatives for scheduling vessels and be more flexible, which further yielded more benefits to the MCT operators throughout the intense competition with other ports.

Cordeau et al. (2005) developed mathematical models for the discrete and continuous BSPs. The total vessel turnaround time was minimized. A Tabu Search (TS) heuristic and a truncated Branch-and-Bound method were adopted as solution approaches. For the small-size problem instances both approaches returned high-quality solutions, while for the large-size problem instances the heuristic method outperformed the exact Branch-and-Bound method.

Li et al. (2005) formulated a short-term BSP, aiming to minimize the total tardiness for the arriving vessels. Special industrial characteristics were considered in modeling the problem. A Lagrangian relaxation and a series of the appropriate branching rules were adopted to define a lower bound to be used along with a Branch-and-Bound method. The efficiency of the algorithm was proved using the collected realistic operational data from the Baoshan Iron and Steel Complex (China). It was concluded the provided method could solve the problem in a reasonable computational time.

Boile et al. (2006) formulated both discrete and continuous BSPs, which aimed to minimize the total service time of the arriving vessels. Moreover, a service priority approach was considered to reflect the existing policies at the ports. The proposed non-linear formulation was transformed to a linear formulation in that study. The small instances were solved using GAMS, and the new formulation returned the same solution in comparison with the non-linear one. For larger instances, a heuristic was proposed.

Golias et al. (2006) presented a formulation for a BSP to evaluate the contractual agreement, where the vessels could be served and request departure during a certain time window. A dynamic discrete BSP was developed to minimize the vessel late departure cost along with maximizing the benefits from the vessel early and timely departure. The proposed contractual agreement was compared against an alternative agreement, where the vessel departure time was

treated as a point of time. Numerical experiments demonstrated superiority of the proposed agreement.

Zhou et al. (2006) proposed a model for minimizing the total turnaround time of the arriving vessels at the MCT. Dynamic arrival and discrete layout were considered for the MCT in that study; however, the first come first served approach was not applied. A GA was developed to solve the problem. The basic search space was reduced by considering the approximate optimal solution properties. The conducted numerical experiments proved the efficiency of the model for the real-world scenarios.

Imai et al. (2007) developed a multi-objective model for a BSP, aiming to: 1- minimize the delay in vessel departure time to increase the service quality, and 2- improve berth utilization by minimizing the total turnaround time. A GA and a Sub-Gradient Optimization procedure were designed to solve the problem. Comparing the provided solutions by the two aforementioned optimization approaches, it was found that GA outperformed the Sub-Gradient Optimization procedure.

Hansen et al. (2008) proposed a mathematical model, minimizing the total service cost of all vessels at the MCT, considering both earliness and tardiness of the service completion. A static arrival time was assumed in that study. Since the exact optimization methods required a prohibitively large computational time to solve the problem, a Variable Neighborhood Search (VNS) was adopted. Besides, Multi-Start (MS) heuristic, GA, and Memetic Algorithm (MA) were developed to check the quality of the solutions, which were provided by the VNS algorithm. The results showed that VNS outperformed the MS, GA, and MA algorithms.

Imai et al. (2008) presented a model to enhance the multi-user terminal (MUT) management. At the MUTs, the vessels with the service time that exceeds the expected waiting time are assigned to an external terminal. In that study, the total turnaround time of the vessels served at the external terminal was minimized. A GA-based heuristic was designed to solve the problem. The quality of the provided solutions was examined by implementing a wide range of

computational experiments. It was concluded that the developed model could be adopted by the MUT operators for efficient management of the vessel handling processes during the peak hours.

Golias et al. (2009) aimed to minimize the total turnaround service time (including waiting time and handling time) of the vessels served at the MCT considering priority agreements. A GA-based heuristic was developed to solve the multi-objective problem. The GA-based heuristic returned a set of solutions, which could enable the MCT operators assessing and choosing the best one according to the operational efficiency and the customer satisfaction.

Golias et al. (2010a) accounted for environmental sustainability of vessel handling processes at the MCT. The authors proposed a model for BSP, which minimized the total emissions and fuel consumption by vessels, when sailing at the next port of call, along with minimizing the total turnaround time of vessels served at the port. A GA-based optimization model was designed to solve the problem. The computational experiments proved efficiency of the proposed berth scheduling policy.

Golias et al. (2010b) proposed a unified mathematical formulation for a BSP at the MCT, considering the majority of the MCT operator's objectives. The objective function of the proposed model minimized the total vessel service cost, including the costs due to vessel waiting and handling times, late vessel departures, and deviation from the agreed vessel productivity. Also, the model accounted for premiums due to early and timely vessel departures (i.e., negative cost components). The study provided a detailed discussion regarding complexity of the proposed mathematical formulation.

Golias et al. (2010c) proposed a Lamda-Optimal based heuristic to solve a complex discrete BSP. In order to reduce the computational performance for the medium- and large-scale problem instances, a second internal GA-based heuristic was developed. The computational experiments showed that the solutions returned by the developed heuristic were near-optimal and were also obtained in a reasonable computational time.

Golias and Haralambides (2010) proposed a mathematical model for the BSP, which minimized the total cost of the vessel late departure and waiting time and maximized the benefits from vessel early departures. It was assumed that each vessel had its own special contractual agreement (i.e., different cost function), although the vessels could belong to the same liner shipping company. The applicability and superiority of the nonlinear cost functions were assessed against the linear ones, which are commonly used in the BSP literature. In order to solve the developed mathematical model, a GA-based heuristic was proposed. The conducted computational experiments not only evaluated the presented berth scheduling policy, but also showed how the adopted cost functions could affect the distribution of the total cost among all the vessels, calling for service at the MCT.

Saharidis et al. (2010) developed a hierarchical optimization framework for optimizing a discrete dynamic BSP. The framework adopted two levels of hierarchy that covered two conflicting objectives throughout the vessel service at the MCT. The k-th best algorithm was developed to solve the presented bi-level BSP mathematical model with two conflicting objective functions at each level. The results from the numerical experiments showed the ability of the developed algorithm to produce high-quality solutions as compared to the solutions generated by its single-level formulation counterpart.

Arango et al. (2011) utilized the Arena software to optimize and simulate a BSP emerged from the Port of Seville, which is the only inland port in Spain. The non-linear problem was solved by designing a heuristic procedure, which was based on a GA. The problem aimed to minimize the total turnaround time for each vessel, considering the first-come-first-served approach. The computational experiments, conducted based on the realistic MCT operational data, proved efficiency of the proposed model in improving operations at MCTs.

Buhrkal et al. (2011) described several major models, presented in the BSP literature, including: 1- Imai et al. (2001), minimizing the total waiting and handling times of the vessels, 2- a Heterogeneous Vehicle Routing Problem with Time Windows formulation (HVRPTW), minimizing the weighted sum of vessel service times, 3- an improved HVRPTW formulation, minimizing the weighted sum of vessel service times, and 4- a generalized set partitioning

formulation, minimizing the total vessel service time. All the models were solved using CPLEX. In that study, a set-partitioning model was adopted and its efficiency was assessed through the extensive computational experiments.

Tong and Zhao (2011) developed a model with a time-space diagram to optimize the MCT operations at the seaside. A dynamic vessel arrival pattern and a discrete berthing layout were adopted in that paper. It was aimed to minimize the total vessel service time, including the handling time and the waiting time. A GA was designed to solve the proposed mathematical model. The numerical experiments proved efficiency of the designed algorithm and its applicability for the container terminal management.

Lalla-Ruiz et al. (2012) combined a TS and a Path Relinking to develop a hybrid metaheuristic for solving a dynamic discrete BSP. The presented mathematical model aimed to minimize the total turnaround time of the served vessels. The results showed that in case of the small-size instances, the algorithm performed almost the same as some of the others in the literature. However, in case of the large-size instances, the algorithm, developed in that study, outperformed the other candidate algorithms, which were reported in the BSP literature.

Oliviera et al. (2012) adopted a Clustering Search method and a Simulated Annealing (SA) to solve a BSP. A dynamic vessel arrival and a discrete berthing layout were considered in that study. The objective function minimized the total service cost plus the total penalty due to violation of the pre-determined vessel schedules. The numerical experiments showed effectiveness of the proposed model in case of both computational time and solution quality as compared to other considered algorithms, which were previously presented in the BSP literature.

Song et al. (2012) developed a bi-level model to solve a combination of BSP and the Quay Crane Scheduling Process (QCSP). The upper level was used to solve the BSP, while the lower one was used to capture the QCSP. In order to minimize the total turnaround time for each vessel, a GA and a Branch-and-Bound method were utilized to solve the upper and lower level problems respectively. The developed model was applied to PSA Singapore Pasir Panjang container terminal and demonstrated efficiency of the developed solution approach.

Sun (2012) studied different types of decision problems in his dissertation regarding the MCT operations, which included: 1- Multiple BSP (MBSP), 2- integrated Simultaneous Berth Allocation and Quay Crane Scheduling Problem (BAQCSP), and 3- Multiple BAQCSP (MBAQCSP). Different types of berthing layouts (discrete, continuous, and semi-continuous (or hybrid)) and vessel arrivals (static and dynamic) were examined in that dissertation. A Branch and Price (B&P) algorithm, GA-based heuristic, and TS were developed for the aforementioned decision problems to minimize the total turnaround time of vessels and penalties due to late departures. The numerical experiments, conducted based on the randomly generated problem instances, showed effectiveness of developed methods and solution strategies.

Xu et al. (2012) developed a BSP mathematical model, considering both spatial requirements (water depth and berthing position length) and the tidal patterns. A parallel-machine scheduling approach was utilized to model the problem, where the constraints were considered inclusively. The time horizon was divided into two periods. Both dynamic and static vessel arrivals were analyzed using the developed heuristics. Extensive computational experiments were performed to test efficiency of the designed heuristics and also to assess the effects of considering the tidal patterns in berth scheduling.

Zhen and Chang (2012) focused on modeling uncertainties associated with the MCT seaside operations. In fact, the authors aimed to account for the uncertainly in vessel arrival and service times using a bi-objective optimization model (minimizing the cost and maximizing the robustness of the berth schedule). The results, provided by the developed heuristic, showed the efficiency and effectiveness of the proposed methodology.

Cubillos et al. (2013) developed a multi-agent based strategy for a dynamic discrete BSP. The model architecture could be described with the following attributes: 1- the interface layer, including Ship Agent and Berth Agent, and 2- the planning layers, including Berth Request Agent, Dock Agent, Berth Planner Agent, and Central Agent. It aimed to maximize the vessel throughput and the berth utilization. The java environment was adopted to create the multi-agent architecture, where new vessels joining to an existing berth sequence were counted using the insertion algorithm.

Karafa et al. (2013) proposed a bi-objective model for a BSP with stochastic vessel handling times. Minimizing the total turnaround time and the corresponding risk were determined as two objectives in that study. An Evolutionary Algorithm-based heuristic and a simulation-based Pareto front pruning algorithm were developed to solve the presented problem. According to the numerical experiments, it was concluded that the proposed strategy outperformed the alternative approach which utilized the expected value of the vessel handling times.

Sheikholeslami et al. (2013) did a simulation analysis to determine suitable access channel depths and increase the number of berthing positions, which can be considered as two practical approaches for decreasing the vessel waiting time. The Enterprise Dynamic software was adopted to simulate the MCT operations. A case study was performed in order to evaluate the proposed simulation model. The results showed a significant reduction in the vessel waiting time, and it was concluded that the proposed method could be adopted by the MCT operators to improve the terminal productivity.

Golias et al. (2014) focused on modeling uncertainty in vessel arrival times and vessel handling times throughout the berth scheduling. A mathematical model was presented to minimize the average and the range of the total turnaround time for all the vessels, which were served at the MCT. A heuristic algorithm was adopted to solve the presented problem. Furthermore, the results were simulated and were compared to the three common vessel service policies used in practice by the MCT operators.

Imai et al. (2014) considered the case, where the MCT operator faced with excessive vessel service requests, while some new service contracts were under consideration. A strategic berth template problem (BTPS) was designed, where a limited number of vessels were selected to be served and also their berth-windows were scheduled in a specific period of time. A sub-gradient optimization strategy was adopted to solve the problem. A wide range of numerical experiments were implemented, which proved effectiveness of the proposed model for the considered cases.

Legato et al. (2014) incorporated a mathematical programming model at the tactical level and a simulation model at the operational level for a BSP in a Simulation-Optimization framework.

A Beam Search heuristic was adopted to solve the problem at the tactical level, while SA was utilized to assist with the decisions at the operational level. The numerical experiments showed how adjustments at the operational level could affect the results at the tactical level.

Ting et al. (2014) formulated a mixed-integer programming model for a discrete dynamic BSP. The objective function of the presented model minimized the total vessel turnaround time (represented as a summation of the vessel handling and waiting times). A Particle Swarm Optimization (PSO) algorithm was adopted to solve the problem. The numerical experiments were conducted to evaluate the proposed algorithm against other candidate algorithms, which were presented in the BSP literature. The results proved superiority of the PSO algorithm in terms of both solutions quality and computational time.

Hu (2015) considered a daytime preference for a BSP, aiming to reduce the total number of night workloads. A bi-objective model, extending the BSP formulations presented in the literature, was proposed to minimize both delayed workloads and night workloads. A multi-objective GA (moGA), inspired by the features of the NSGA-II algorithm, was developed to solve the problem. A set of important managerial insights were demonstrated using the proposed solution algorithm throughout the numerical experiments.

Cao et al. (2016) studied a combined discrete dynamic BSP and a quay crane assignment problem considering the cost balance. The authors developed a model, aiming to minimize the total operational cost. A GA was adopted to solve the proposed model. The numerical experiments showed effectiveness of the developed GA-based method in solving the developed problem.

Emde and Boysen (2016) developed a model for a BSP at a container terminal, which served both feeder vessels and mother vessels. The study focused on container exchange between the mother vessels and the feeder vessels with the objective, aiming to serve each vessel at the earliest possible time after its arrival. An SA-based local search was adopted to solve the problem. The results demonstrated efficiency of the algorithm in providing high-quality solutions.

Hsu (2016) proposed a mathematical formulation for a dynamic discrete BSP and quay crane assignment problem. A Hybrid PSO (HPSO), which relied on an improved PSO and an event-based heuristic, was developed to solve the problem. The objective of the model aimed to minimize the total cost. The results, returned by the HPSO algorithm, were compared with the results, provided by a GA with time-invariant QC assignment and a hybrid GA (HGA) with variable-in-time QC assignment. The results indicated that the HPSO algorithm outperformed the two other algorithms for the generated problem instances.

Kordic et al. (2016) solved the Discrete BSP (DBSP) and the Hybrid BSP (HBSP) using an exact combinatorial algorithm (which was called as a Sedimentation Algorithm). The objective of the proposed mathematical models aimed to minimize the total cost, associated with the vessel service. The results showed that the proposed algorithm outperformed the other exact optimization approaches in terms of the solution quality for relatively large problem instances with up to 65 vessels calling for service at the MCT.

Lalla-Ruiz et al. (2016) presented a mathematical model, which accounted for both spatial requirements and regional tidal cycles. In that model, there was no limitation on the number of the low and high tides occurrence during the process of serving a given vessel. The objective of the proposed model minimized the total vessel turnaround time. A Generalized Set-Partitioning based algorithm was utilized to solve the problem. The computational experiments indicated that the developed mathematical model outperformed the other models presented in the literature in terms of the following aspects: 1- it guaranteed the feasibility and optimality of the solutions regardless of the given problem, 2- lower computational time, and 3- solving new large problems in a reasonable computational time.

Mauri et al. (2016) modeled a BSP considering both the spatial and temporal constraints. A dynamic vessel arrival time was assumed in that study. Two berthing layouts were considered, including the discrete and continuous berthing layouts. An Adaptive Large Neighborhood Search (ALNS) heuristic was adopted in that study to minimize the total turnaround time. It was concluded that the presented algorithm returned significantly better solutions for both discrete and continuous BSPs as compared with other well-known BSP algorithms.

Ribeiro et al. (2016) considered the effects of maintenance activities (that had to be done at the berthing positions) throughout berth scheduling. The authors formulated a mixed-integer linear programming model for the BSP, aiming to minimize the total vessel turnaround time. The developed model was solved using the ALNS algorithm. A set of computational experiments were conducted in order to evaluate performance of the proposed solution approach for the presented BSP formulation. The results proved that ALNS could return high-quality solutions within a reasonable computational time.

Ursavas and Zhu (2016) took into consideration uncertain nature of the vessel arrival and handling times throughout berth scheduling. The authors presented a framework based on the stochastic dynamic programming approach to cover all the operational characteristics of the problem. The presented optimal control policy depended on the number of vessels at a given berthing position. It was found that the proposed methodology was promising and could assist the MCT operators with improving the seaside operations.

Dulebenets et al. (2017a) proposed an innovative mixed-integer mathematical model for a BSP considering the environmental aspects. The objective function minimized the total service cost of vessels, which covered the total carbon dioxide (as a known greenhouse gas) emission cost imposed by container handling. The authors deployed a Hybrid Evolutionary Algorithm, including a set of local search heuristics, to solve the developed mathematical model. The results indicated that application of the local heuristics led to more promising solutions. Moreover, based on the results provided by the computational experiments, application of the carbon dioxide emission cost in the mathematical model changed the berth schedules remarkably.

Dulebenets (2017) developed a mixed-integer nonlinear mathematical model for a BSP, aiming to minimize the total vessel service cost. An MA with a deterministic parameter control was designed to solve the defined problem. To improve the quality of the initial solutions and increase the convergence speed, a local search heuristic that was based on a first-come-first-serve approach was adopted for the population initialization. The numerical experiments showed that the proposed solution approach returned superior objective function values as compared to the other state-of-the-art algorithms, which have been presented in the BSP literature.

Lalla-Ruiz et al. (2017) developed a Partial Optimization Metaheuristic under Special Intensification Conditions (POPMUSIC) heuristic for a BSP at the MCT. The objective of the presented mathematical formulation aimed to minimize the total handling time of all the vessels, calling for service at the MCT. POPMUSIC can be classified as a metaheuristic, which used the exact methods for solving sub-problems. The numerical experiments showed that the developed approach could provide high-quality berth schedules.

Leon et al. (2017) proposed a mathematical model for a BSP at a bulk marine terminal (Bulk-BSP). The model aimed to minimize the total turnaround time of the arriving vessels, considering both berthing and yard activities. The appropriate algorithm for each instance was selected based on the BSP literature, No Free Lunch Theorem, and a Machine-Learning based system. The numerical experiments indicated that selection of the appropriate algorithm for each instance was found to be more efficient as compared to selection of the best algorithm over all the generated problem instances.

Li et al. (2017) developed a multi-objective optimization model, considering both berth and quay crane scheduling at the MCT. The model aimed to minimize the total turnaround time and the total travel distance of yard trucks. An improved PSO was adopted to solve the problem. A total of eight problem instances were used throughout the numerical experiments in order to assess efficiency of the developed solution algorithm.

Pan et al. (2017) studied a combined berth and quay crane scheduling problem at the MCT with 1-lookahead ability. The study aimed to minimize the total turnaround time of the arriving vessels. Three different types of competitive algorithms were utilized for the problem instances with different number of quay cranes. A set of computational experiments demonstrated that the presented solution algorithms were able to obtain the solutions of a good quality.

Pratap et al. (2017) focused on a combined berth allocation and material handling problem at the MCT, unlike the previously conducted studies. In order to solve the latter decision problem, two different strategies were considered: 1- solving the problem sequentially by decomposing the problem into the two sub-problems including the berth allocation and the dynamic allocation of

vessel unloaders at different berths, and 2- solving the problem by integrating the berth allocation and dynamic allocation problems. The objective of the proposed mathematical model minimized the summation of the total waiting time, operating time, and vessel priority deviation. A GA-based heuristic was developed to solve the problem. The results indicated that integration of the berth and vessel unloader allocation yielded significant cost savings by reducing the total vessel turnaround time.

Umang et al. (2017) studied BSP at the MCT, taking into account uncertainty in vessel arrival times and handling times. The objective aimed to minimize the total vessel service cost. An Optimization-Based Recovery algorithm, which relied on the features of a Set Partitioning and a Smart Greedy algorithm, was adopted to solve the problem and tackle possible disruptions in the vessel arrival and handling times. The numerical experiments were conducted using real-life MCT operational data. It was found that the proposed methodology could remarkably reduce a potential increase of the total vessel service cost due to uncertainty in vessel arrival times and handling times.

Zhen et al. (2017) focused on a daily basis berth and quay crane scheduling problem, considering the tidal and channel flow constraints. The column generation and column enumeration algorithms were adopted to solve the problem, aiming to minimize the total waiting time and the late departure for all the vessels. A total of 30 real-world problem instances with up to 80 vessels, 40 berths, and 120 quay cranes were generated in order to conduct the computational experiments. The proposed methodology was found to be efficient, especially for the MCTs that experience the tidal effects and have channel access.

Dulebenets et al. (2018a) proposed a berth scheduling policy, where the demand could be diverted from a multi-user maritime container terminal to an external maritime container terminal at an additional cost. The objective of the proposed mathematical model aimed to minimize the total vessel service cost. Additional market-based rules were considered for vessel diversion decision making. An MA was developed to solve the defined problem. An extensive set of numerical experiments were conducted to assess the applicability of the proposed model and the

developed algorithm. The results confirmed that the model could yield remarkable cost savings during the peak demand periods.

Dulebenets (2018c) developed an innovative EA applying a parameter control strategy for a BSP at the MCT. In the parameter control strategy, the mutation rate was changed based on the feedback from the search. The total weighted vessel service cost was minimized in the developed mathematical model. A series of numerical experiments indicated that the deployed parameter control strategy could present more promising solutions in comparison with the results from the typical Evolutionary Algorithms.

Table 2 Adopted acronyms for the major BSP attributes.

Attribute	**Description**
1- Spatial	
- D	discrete
- C	continuous
- Dr	vessels draft consideration
2- Vessel arrivals	
- S	static
- D	dynamic
3- Handling times	
- C	constant
- V	variable
4- Performance measures	
Compl	service completion time of all vessels
Wait	waiting time of vessels
Hand	handling time of vessels
Late	late departures of vessels
Dev	deviation between actual and desired berthing positions
Fail	failing to provide a service request
Order	deviation between arrived vessels order and their service order
Fuel	fuel consumption of vessels
Other	different from ones, listed above
w	weighted coefficient

4.2. Literature Summary

This section provides a detailed summary of the reviewed BSP studies. The acronyms, adopted for the major BSP attributes, are provided in Table 2. The acronyms, adopted for the algorithms that were proposed in the BSP literature, are presented in Table 3. Table 4 provides the key information regarding each one of the collected BSP studies, including the following information: (1) author(s); (2) year; (3) spatial attribute; (4) vessel arrivals; (5) vessel handling times; (6) objective(s); and (7) solution approach.

Table 3 Adopted acronyms for the BSP solution algorithms (continued).

Solution Approach	Notation
Adaptive Large Neighborhood Search	ALNS
Branch-and-Bound Algorithm	B&B
Branch-and-Price Algorithm	B&P
Chemical Reaction Optimization	CRO
Cluster Search	CS
Column Enumeration Algorithm	CEA
Column Generation Algorithm	CGA
Competitive Algorithm	CA
Evolutionary Algorithm-Based Heuristic	EABH
GA-based Heuristic	GABH
Generalized Set-Partitioning Problem	GSPP
Genetic Algorithm	GA
Greedy Heuristic	GH
Greedy Randomized Algorithm	GRA
Heuristic Algorithm	HA
Heuristic First-Come First-Served	HFCFS
Hybrid Genetic Algorithm	HGA
Hybrid Particle Swarm Optimization	HPSO

Table 3 Adopted acronyms for the BSP solution algorithms (continued).

Solution Approach	Notation
Insertion Algorithm	IA
k-th Best Algorithm	k-th BA
Lamda-Optimal-Based Heuristic	LOBH
Large Neighborhood Search	LNS
Memetic Algorithm	MA
Partial Optimization Metaheuristic Under Special Intensification Conditions	POPMUSIC
Particle Swarm Optimization	PSO
Path Relinking	PR
Recovery Algorithm	RA
Second Internal Genetic Algorithm Based Heuristic	SIGABH
Sedimentation Algorithm	SEDA
Simulated Annealing	SA
Simulated annealing based local search	SALS
Simulation	S
Simulation-Based Pareto Front Pruning Algorithm	SPFPA
Simulation-Optimization	SO
Smart Greedy Algorithm	SGA
Stochastic Beam Search	SBS
Stochastic Dynamic Approach	SDA
Sub-gradient Optimization Procedure	SOP
Tabu Search	TS
Truncated Branch-and-Bound Method	TB&B
Variable Neighborhood Search	VNS

Table 4 Overview of the DBSP formulations.

#	Author(s)	Year	Spatial Attribute	Vessel Arrivals	Handling Times	Objective(s)	Solution Approach
1	Brown et al.	1997	D	D	C	Σ(Fail+Dev)	CPLEX
2	Imai et al.	2001	D	S&D	V	Σ(Wait+Hand)	HA
3	Hansen & Oguz	2003	D	S&D	V	Σ(Wait+Hand)	CPLEX
4	Imai et al.	2003	D	D	V	Σw(Wait+Hand)	GA
5	Cordeau et al.	2005	D	D	V	Σw(Wait+Hand)	TS, TB&B
6	Li et al.	2005	D&Dr	S	V	Σ(Late)	B&B
7	Boile et al.	2006	D	D	V	Σw(Wait+Hand)	GAMS & HA
8	Zhou et al.	2006	D&Dr	D	V	Σw(Wait)	GA
9	Golias et al.	2007	D	D	V	$\Sigma[w_1(\text{Late})+w_2(\text{Other})]$	GAMS
10	Imai et al.	2007	D	D	V	Σw(Late)&Σ(Wait+Hand)	GA & SOP
11	Hansen et al.	2008	D	D	V	$\Sigma[w_1(\text{Wait})+w_2(\text{Hand})+w_3(\text{Late})]$	VNS
12	Imai et al.	2008	D	S&D	V	Σ(Wait+Hand)	GABH
13	Golias et al.	2009	D	D	V	Σ(Wait+Hand)	GABH
14	Golias et al.	2010a	D	D	V	Σ(Wait+Hand+Late+Fuel)	GABH
15	Golias et al.	2010b	D	D	V	$\Sigma[w_1(\text{Wait})+w_2(\text{Hand})+w_3(\text{Late}) + +w_4(\text{Other})]$	N/A
16	Golias et al.	2010c	D	D	V	Σw(Wait+Hand)	LOBH & SIGABH
17	Golias & Haralambides	2010	D	D	V	$\Sigma[w_1(\text{Wait})+w_2(\text{Late})+w_3(\text{Other})]$	GABH
18	Saharidis et al.	2010	D	D	V	Σ(Wait+Hand)	k-th BA
19	Arango et al.	2011	D	D	V	Σ(Wait+Hand)	GABH
20	Buhrkal et al.	2011	D	D	V	Σ(Wait+Hand)	CPLEX, GH

#	Author(s)	Year	Spatial Attribute	Vessel Arrivals	Handling Times	Objective(s)	Solution Approach
21	Oliviera et al.	2011	D	D	V	Σw(Wait+Hand)	CS & SA
22	Shan and Jun	2011	D	D	V	Σ (w1(Wait) + w2(Delay) + w3(Hand))	GA
23	Lalla-Ruiz et al.	2012	D	D	V	Σ(Wait+Hand)	TS & PR
24	Song et al.	2012	D	S	V	Σ(Wait+Hand)	GA & B&B
25	Sun	2012	D&Dr	S&D	V	Σ(Wait+Hand+Late)	GABH, TS, B&P
26	Xu et al.	2012	D&Dr	S&D	V	Σw(Wait+Hand)	HA
27	Zhen and Chang	2012	D	D	V	$\Sigma[w_1(Late)+w_2(Dev)]$+Other	HA
28	Cubillos et al.	2013	D	D	V	Other	IA
29	Karafa et al.	2013	D	D	V	Σ(Wait+Hand)+Other	EABH & SPFPA
30	Sheikholeslami et al.	2013	D	S	C	N/A	SIM
31	Golias et al.	2014	D	D	V	Σ(Wait+Hand)+Other	HA
32	Imai et al.	2014	D	D	V	Σ(Wait+Hand)+Other (cost)	SOP
33	Legato et al.	2014	D	D	V	$\Sigma[w_1(Dev)+w_2(Order)]$	SO
34	Ting et al.	2014	D	D	V	Σ(Wait+Hand)	PSO
35	Hu	2015	D	D	V	Σ (w1(Late))	GA
36	Cao et al.	2016	D	D	V	Σ(Wait+Hand) + Other (Cost)	GA
37	Emde and Boysen	2016	D	S	V	Σ(Hand)	SALS
38	Hsu	2016	D	D	V	Σ (w1(Wait) + w2(Delay) + w3(Hand)) (Cost)	HPSO & GA & HGA
39	Kordic et al.	2016	D&H	D	C	Σ (w1 (Wait) + w2 (Speed) + w3 (Late) + w4 (Hand))	SEDA
40	Lalla-Ruiz et al.	2016	D	D	C	Σ (w1(Hand))	CPLEX & GSPP

#	Author(s)	Year	Spatial Attribute	Vessel Arrivals	Handling Times	Objective(s)	Solution Approach
41	Mauri et al.	2016	D&C	D	V	Σ[Wait + Hand + w1(Late) + w2(Order)]	ALNS
42	Ribeiro et al.	2016	D	D	C	Σ[Hand + Wait + Late + Other] (Cost)	CPLEX & ALNS
43	Ursavas and Zhu	2016	D	D	V	Σ (w1(Wait) + w2(Hand)) (Cost)	SDA
44	Dulebenets et al.	2017	D	D	V	Σ[Wait + Hand + Late + Other]	GA
45	Dulebenets	2017	D&C	D	V	Σ[Wait + Hand + w1(Late) + w2(Order)]	MA
46	Lalla-Ruiz et al.	2017	D	D	V	Σ (w1(Hand))	POPMUSIC & CPLEX
47	Leon et al.	2017	D	D	V	Σ (Hand+Wait)	GRA, HFCFS, LNS
48	Li et al.	2017	D	D	V	Σ (Hand), Σ (displacement)	PSO
49	Pan et al.	2017	D&C	D	V	Σ (Hand)	CA
50	Pratap et al.	2017	D	S	V	Σ[Wait + Hand + Late]	GA & CRO
51	Umang et al.	2017	D&C	D	V	Σ[w1(Hand) + w2(Delay)]	RA & SGA
52	Zhen et al.	2017	D	D	C	Σ[w1(Wait) + w2(Late)]	CGA, CEA & CPLEX
53	Dulebenets et al.	2018	D	D	V	Σ[Hand + Late]+Other (Cost)	MA
54	Dulebenets	2018	D	V	V	Σ (w1 (Wait) + w2 (Hand) + w3 (Late)) (Cost)	GA

A distribution of the collected studies by the solution approaches used for BSP is presented in Figure 7. As it is illustrated in the Figure 7, GA is the most deployed algorithm in the BSP literature. CPLEX, HA, GABH, and TS are the other types of solution approaches which have been utilized moderately in the BSP literature. Other solution approaches were deployed twice or once by the authors in the reviewed studies. GA, as a metaheuristic algorithm, was utilized in 11 studies. However, CPLEX was used in 7 studies as an exact solution approach, which mostly has been deployed to assess the accuracy of the results, obtained by stochastic algorithms for the small-size problem instances.

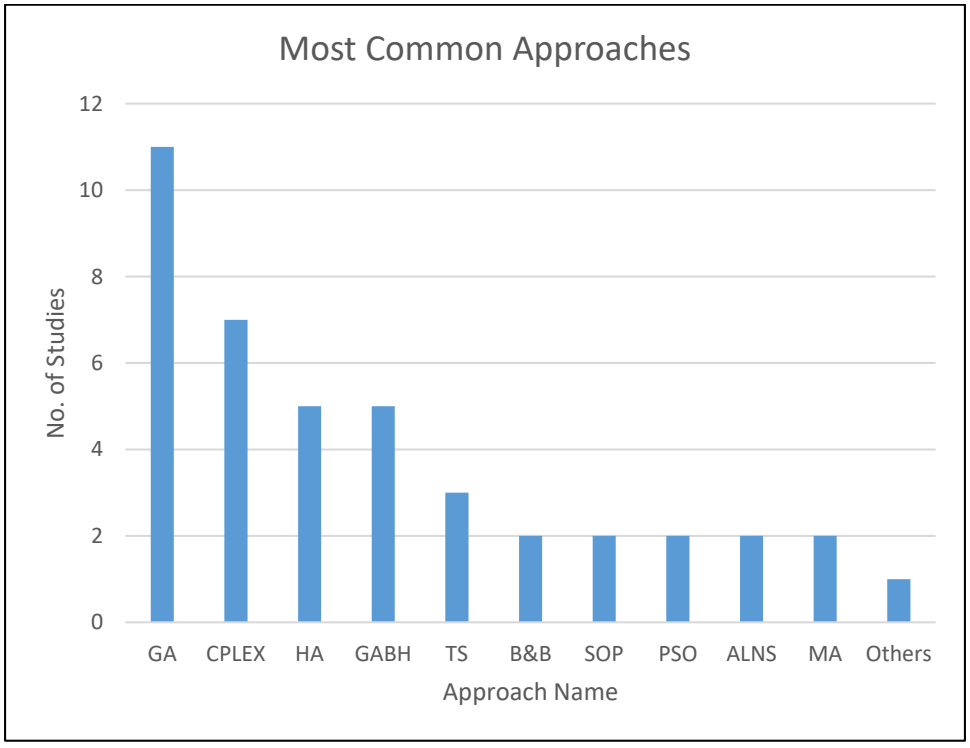

Figure 7 A distribution of the collected BSP studies by the solution approaches.

The distribution of the collected studies by the considered vessel arrivals is depicted in Figure 8. It is obvious that the dynamic vessel arrival is the predominant vessel arrival strategy as compared with the static one. The dynamic vessel arrival is defined as vessels would arrive at the terminal at a certain time period in the considered planning horizon, while a static vessel arrival means that all the vessels have already arrived at the MCT. Regarding these definitions, the dynamic vessel arrival is much more compatible with what happens in real world; thus, it received

much more attention in the BSP literature. However, some of the studies considered both approaches to make a comparison between them.

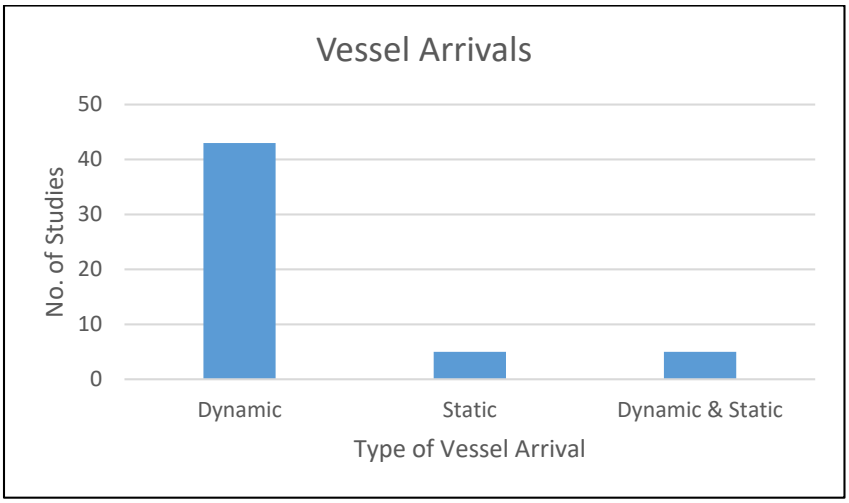

Figure 8 A distribution of the collected BSP studies by the vessel arrivals.

Figure 9 shows the distribution of the reviewed studies by the spatial requirements. This dissertation focuses on the discrete MCTs, thus all the reviewed studies considered a discrete berthing positions layout for the MCT. In some cases, the studies deployed other types of spatial requirements (continuous, draft, and hybrid) to make a comparison among the obtained results.

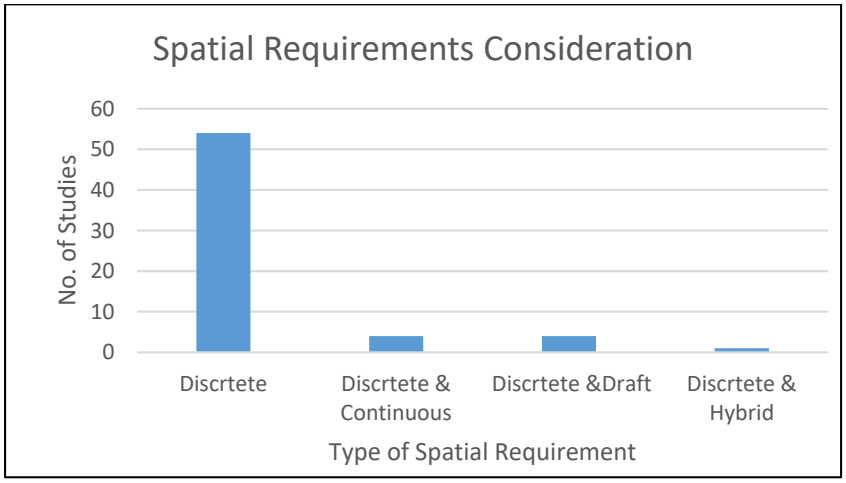

Figure 9 A distribution of the collected BSP studies by the spatial requirements.

The distribution of the reviewed studies by the handling time is illustrated in Figure 10. If the number of the assigned QCs and also their handling productivity remain constant during the service time of a vessel, the handling time is constant, otherwise it is variable. According to the aforementioned definitions, a case in which the handling time could be classified as a constant one would rarely happen. Hence, to model the real-world conditions, in most of the studies a variable handling time has been deployed.

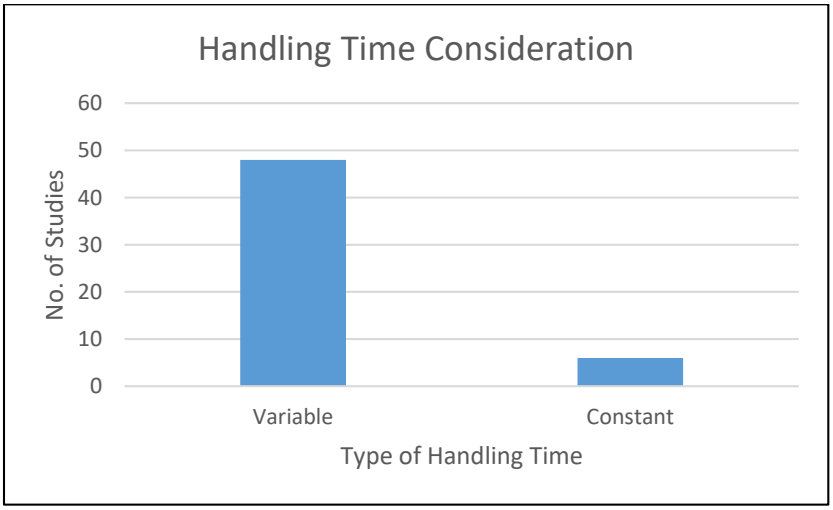

Figure 10 A distribution of the collected BSP studies by the handling time.

The distribution of the components, which were considered in the defined objective functions in the reviewed studies, is shown in the Figure 11. A presented mathematical model should be compatible with practice to the best extent possible. Thus, the defined objective functions in the BSP literature include some components that are considered as critical in the practical applications. Based on the process of serving the vessels at MCTs, all the vessels spend some time at berthing positions to be handled; hence, the handling time is the most common component captured by the mathematical models that were presented in the BSP literature. On the other hand, the waiting time is another component, which is more likely to happen for the arriving vessels compared with the other ones. Late departure is another component, which was considered more often in the BSP mathematical models as compared to deviation from the desired berthing position, order, service failure, and fuel.

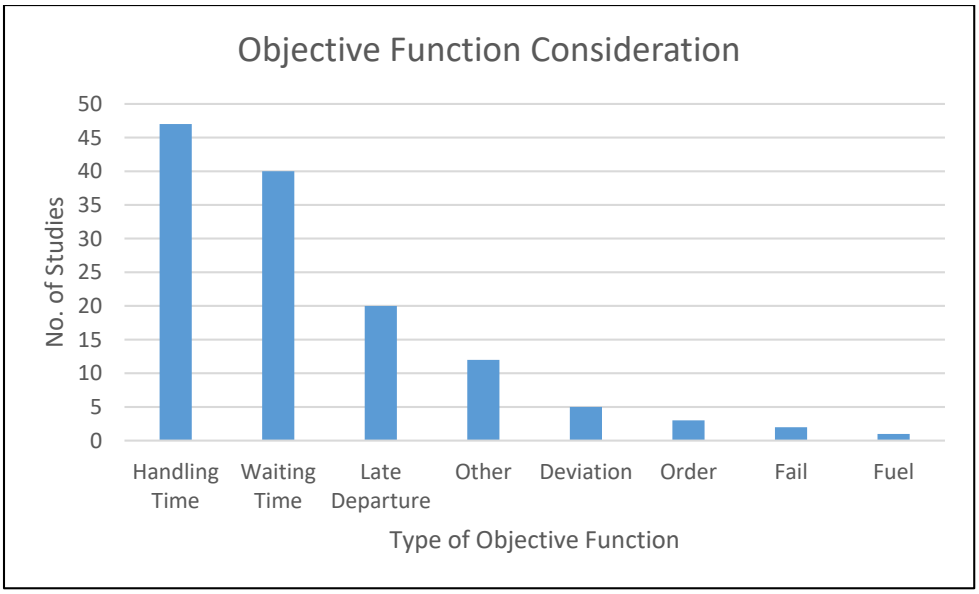

Figure 11 A distribution of the collected BSP studies by the objective function.

4.3. Contribution

There are two common approaches for setting the algorithmic parameters in EAs: 1- parameter tuning, and 2- parameter control. Parameter tuning is a conventional approach to determine the algorithmic parameters, which is based on evaluation of the candidate parameter combinations (the best parameter combination is selected based on a tradeoff between the solution quality and computational time). Whereas, there are three major parameter control strategies: (1) deterministic – the parameter values are updated based on some counter (e.g., computational time) without any feedback from the search; (2) adaptive – the parameter values are updated based on the feedback from the search; (3) self-adaptive – parameter values are encoded in the solutions and are updated throughout the evolution of the algorithm. Based on a detailed review of the BSP literature, the EA-based algorithms with parameter tuning have been widely used for solving BSPs. However, the parameter tuning strategy does not allow changing the EA parameters values throughout the search process, which generally worsens the algorithmic performance. One of the main objectives of this dissertation is to propose and evaluate alternative methods for setting parameters within EAs and to determine how these methods would affect the quality of obtained solutions.

In the first step of this dissertation, a Self-Adaptive Evolutionary Algorithm (SAEA) is developed to solve the proposed BSP mathematical model. In the mathematical model, the total turnaround time (including handling time and waiting time) and late departure of the vessels are minimized. Unlike typical EAs, presented in the BSP literature, the proposed SAEA has crossover and mutation probabilities encoded in its chromosomes and allows both crossover and mutation probabilities evolving throughout the search process. The developed SAEA algorithm will be evaluated against a typical EA, a Deterministic Parameter Control EA (DPCEA), and an Adaptive EA (AEA).

However, SAEA does not take any feedback from the search. Thus, an innovative EA, named as Augmented Self-Adaptive EA (ASAEA), is proposed in this dissertation. Specifically, in the proposed ASAEA, the algorithmic parameters are encoded in the solutions, and the search results are recorded by the algorithm. Those algorithmic parameters, which result in the most significant objective function improvements, are propagated in a certain portion of the population randomly after a pre-specified number of generations. Throughout the computational experiments, the proposed ASAEA will be compared against the EA, DPCEA, AEA, and SAEA algorithms, as well as other common metaheuristic algorithms, which have been commonly used in the BSP literature.

Furthermore, a part of this dissertation is allocated to improve performance of the island-based metaheuristics. In canonical island-based metaheuristics, the same type of a metaheuristic algorithm is executed on all the islands, while in this dissertation, different types of metaheuristics are deployed to cover the islands in the island-based algorithm. In particular, the metaheuristic algorithms, deployed in the island-based framework, are selected based on the features of the algorithms, in the way that they cover each other's drawbacks and provide superior solutions for a BSP. The proposed island-based algorithm is called the Universal Island-based Metaheuristic Algorithm (UIMA), which has four islands. An Evolutionary Algorithm, a Particle Swarm Optimization, a Differential Evolution, and an Estimation of Distribution Algorithm are adopted to cover four islands in UIMA. Performance of UIMA is evaluated against the algorithms adopted for the islands (i.e., four aforementioned algorithms were executed independently) and three single-solution based metaheuristic algorithms.

CHAPTER 5

AN EVOLUTIONARY ALGORITHM WITH SELF-ADAPTIVE PARAMETER CONTROL

The berth scheduling as a seaside operation at a multi-user marine container terminal (MCT), where the vessels from different liner shipping companies deliver and pick up containers, is investigated in this chapter of the dissertation. A discrete berthing layout is adopted for the considered MCT (see Figure 12). A group of n berthing positions ($J = \{1, \dots, n\}$) are considered to serve m arriving vessels $I = \{1, \dots, m\}$ at the MCT. The dynamic vessel arrival pattern is assumed for the MCT in this chapter of the dissertation. The vessel arrival times and handling times can be considered as uncertain factors in the model, while the scope of this chapter does not include modeling uncertainty in any of the aforementioned factors.

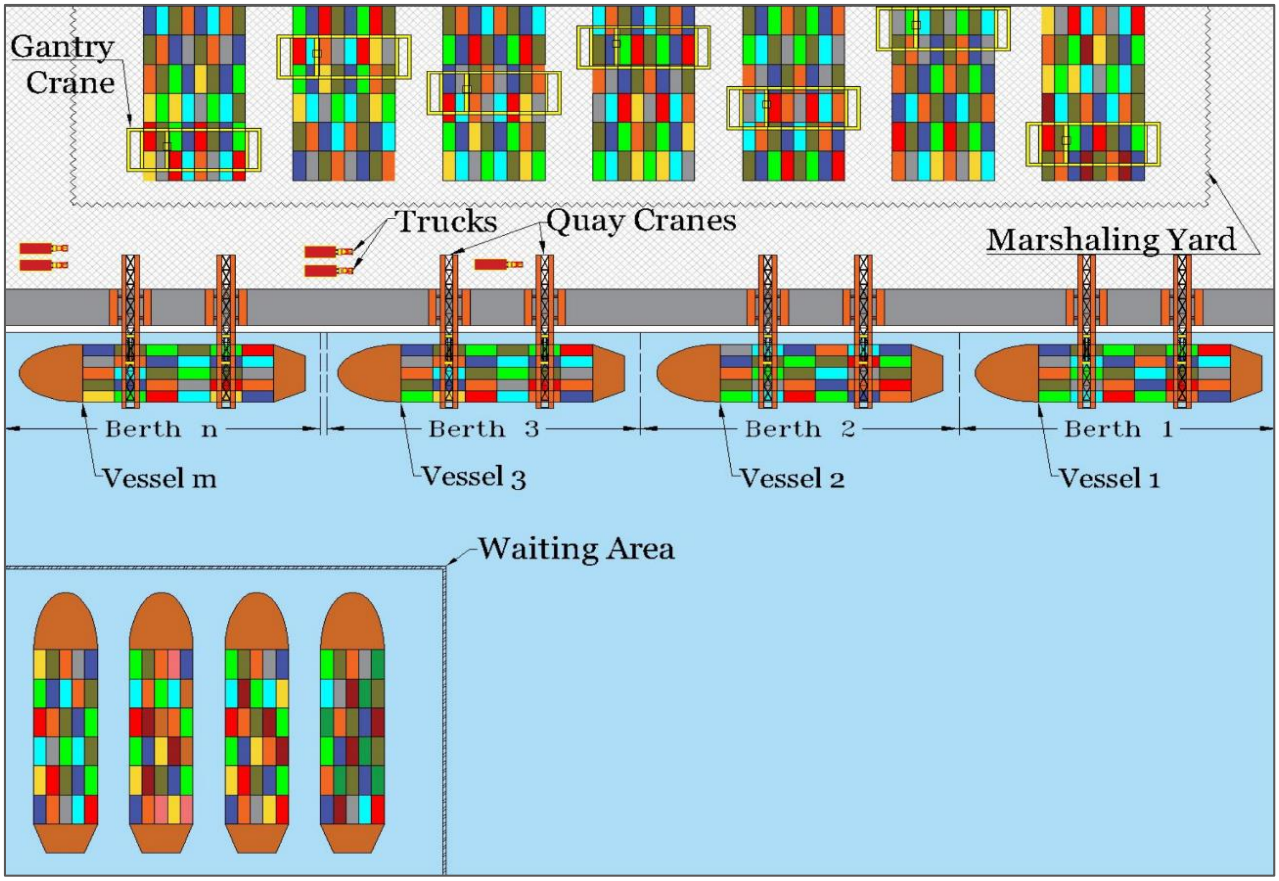

Figure 12 General layout of the MCT.

Once a vessel arrives at the MCT, it will be towed by tug boats to the assigned MCT berthing position. If the berthing position, assigned to the vessel, is not available at the moment, the vessel will be towed to the dedicated waiting area located in a close vicinity to the MCT. The MCT operator should consider the spatial requirements in assignment of vessels to the berthing positions, where the length of the vessel plus the horizontal safety clearance should be shorter than the length of the berthing position. Moreover, the summation of the vessel draft and the vertical safety clearance should be shorter than the depth of the berthing position. In the planning horizon, a preferred berthing position is defined for each one of the arriving vessels, where the MCT can provide the highest handling rate for a given vessel. Based on the contractual agreement with the MCT operator, a specific departure time is requested by the liner shipping company. The MCT operator is expected to pay penalties to the liner shipping companies in case of vessel late departures.

This chapter presents an Evolutionary Algorithm with Self-Adaptive Parameter Control for the considered BSP. In the self-adaptive parameter control strategy, the crossover and mutation probabilities are encoded in the solutions and evolved throughout the search process. Extensive numerical experiments are conducted to evaluate performance of EA with a self-adaptive parameter control strategy against some algorithms commonly used in the BSP literature.

5.1. Model Formulation

The nomenclature, adopted for the developed mixed-integer linear mathematical model for the discrete dynamic berth scheduling problem with spatial requirements (BSPSR), is presented in Table 5.

The objective function of the BSPSR is formulated in equation (1). The objective function (1) aims to minimize the total weighted turnaround time of vessels (which is composed of the total weighted vessel waiting and handling times) and the total weighted vessel late departures.

$$min\, \boldsymbol{Z} = [\sum_{i\in I}(W_i \boldsymbol{T}_i^{WT}) + \sum_{i\in I}\sum_{j\in J}\sum_{k\in K}\left(W_i T_{ij}^{HT} \boldsymbol{x}_{ijk}\right) + \sum_{i\in I}(W_i \boldsymbol{T}_i^{LD})] \quad (1)$$

Table 5 Description of the mathematical model components.

Model Component		**Description**
Type	**Nomenclature**	
Sets	$I = \{1, \dots, m\}$	set of arriving vessels (vessels)
	$J = \{1, \dots, n\}$	set of available berthing positions (berthing positions)
	$K = \{1, \dots, s\}$	set of vessel service orders (service orders)
Decision Variables	$\boldsymbol{x}_{ijk}, i \in I, j \in J, k \in K$	=1 if vessel i is assigned to berthing position j in service order k (=0 otherwise)
Auxiliary Variables	$\boldsymbol{T}_{ijk}^{ID}, i \in I, j \in J, k \in K$	idle time of berthing position j between the start service time of vessel i and a preceding vessel served as $k-1$ vessel (hrs.)
	$\boldsymbol{T}_i^{ST}, i \in I$	start service time of vessel i (hrs.)
	$\boldsymbol{T}_i^{FT}, i \in I$	finish service time of vessel i (hrs.)
	$\boldsymbol{T}_i^{WT}, i \in I$	waiting time of vessel i (hrs.)
	$\boldsymbol{T}_i^{LD}, i \in I$	late departure time of vessel i (hrs.)
Parameters	m	number of arriving vessels (vessels)
	n	number of available berthing positions (berthing positions)
	s	number of vessel service orders (service orders)
	$T_j^B, j \in J$	time when berthing position j becomes available in the planning horizon for the first time (hrs.)

	$T_i^A, i \in I$	arrival time of vessel i at the MCT (hrs.)
	$T_{ij}^{HT}, i \in I, j \in J$	handling time of vessel i at berthing position j (hrs.)
	$T_i^D, i \in I$	negotiated departure time of vessel i (hrs.)
	$L_i^V, i \in I$	length of vessel i (ft.)
	$L_j^B, j \in J$	length of berthing position j (ft.)
	$D_i^H, i \in I$	horizontal clearance requirement for vessel i (ft.)
	$H_i^V, i \in I$	draft of vessel i (ft.)
	$H_j^B, j \in J$	depth of berthing position j (ft.)
	$D_i^V, i \in I$	vertical clearance requirement for vessel i (ft.)
	$W_i, i \in I$	weight of vessel i (value ranging from 0.10 to 1.00)
	Γ	large positive number

Constraint set (2) demonstrates that each berthing position cannot serve more than one vessel at the time.

$$\sum_{i \in I} \boldsymbol{x}_{ijk} \leq 1 \; \forall j \in J, k \in K \tag{2}$$

Constraint set (3) indicates that one of the available berthing positions will serve a given vessel in any service order.

$$\sum_{j \in J} \sum_{k \in K} \boldsymbol{x}_{ijk} = 1 \; \forall i \in I \tag{3}$$

Constraint sets (4) and (5) ensure that vessels will be assigned to the berthing positions that are spatially compatible with their dimensions.

$$(L_i^V + D_i^H)\boldsymbol{x}_{ijk} \leq L_j^B \; \forall i \in I, j \in J, k \in K \tag{4}$$

$$(H_i^V + D_i^V)\boldsymbol{x}_{ijk} \leq H_j^B \; \forall i \in I, j \in J, k \in K \tag{5}$$

Constraint set (6) ensures that the start service time of each vessel will be after its arrival at the MCT.

$$\sum_{i' \in I : i' \neq i} \sum_{k' \in K : k' < k} \left(T_{i'j}^{HT} \boldsymbol{x}_{i'jk'} + \boldsymbol{T}_{i'jk'}^{ID} \right) + \boldsymbol{T}_{ijk}^{ID} - \left(T_i^A - T_j^B \right) \boldsymbol{x}_{ijk} \geq 0 \; \forall i \in I, j \in J, k \in K \tag{6}$$

Constraint set (7) estimates the start service time for each arriving vessel at the MCT.

$$\boldsymbol{T}_i^{ST} \geq \sum_{i' \in I: i' \neq i} \sum_{k' \in K: k' < k} \left(T_{i'j}^{HT} \boldsymbol{x}_{i'jk'} + \boldsymbol{T}_{i'jk'}^{ID} \right) + \boldsymbol{T}_{ijk}^{ID} + T_j^B - \Gamma(1 - \boldsymbol{x}_{ijk}) \; \forall i \in I, j \in J, k \in K \quad (7)$$

Constraint set (8) calculates the waiting time for each vessel arrived at the MCT.

$$\boldsymbol{T}_i^{WT} \geq \boldsymbol{T}_i^{ST} - T_i^A \; \forall i \in I \quad (8)$$

Constraint set (9) calculates the finish service time for each arriving vessel at the MCT.

$$\boldsymbol{T}_i^{FT} = \boldsymbol{T}_i^{ST} + \sum_{j \in J} \sum_{k \in K} (T_{ij}^{HT} \boldsymbol{x}_{ijk}) \; \forall i \in I \quad (9)$$

Constraint set (10) estimates the late departure time for each arriving vessel at the MCT.

$$\boldsymbol{T}_i^{LD} \geq \boldsymbol{T}_i^{FT} - T_i^D \; \forall i \in I \quad (10)$$

The nature of decision variables, auxiliary variables, and parameters of the BSPSR mathematical model are defined in the developed constraint sets (11)-(13).

$$\boldsymbol{x}_{ijk} \in \{0,1\} \; \forall i \in I, j \in J, k \in K \quad (11)$$

$$m, n, s \in N \quad (12)$$

$$\boldsymbol{T}_{ijk}^{ID}, \boldsymbol{T}_i^{ST}, \boldsymbol{T}_i^{FT}, \boldsymbol{T}_i^{WT}, \boldsymbol{T}_i^{LD}, T_j^B, T_i^A, T_{ij}^{HT}, T_i^D, L_i^V, L_j^B, D_i^H, H_i^V, H_j^B, D_i^V, W_i, \Gamma \in R^+ \; \forall i \in I, j \in J, k \in K \quad (13)$$

5.2. Algorithm Design

A detailed description of the self-adaptive EA (SAEA) is presented in this section of the dissertation. SAEA was developed to deal with the real-size problem instances of the BSPSR mathematical model within an acceptable computational time. The proposed SAEA algorithm deploys a self-adaptive parameter control strategy, whereas typical EAs deploy constant values for the algorithmic parameters. In the developed SAEA, the crossover and mutation probabilities (which are considered as the major EA parameters [Eiben and Smith, 2003; Sivanandam and

Deepa, 2008]) are encoded in the chromosomes and changed throughout the search process from one generation to another. The main SAEA steps are described in section 5.2.1 of the dissertation, while a detailed description of each SAEA step is presented in sections 5.2.2 – 5.2.9 of the dissertation.

5.2.1. SAEA main steps

The main SAEA steps are illustrated in Figure 13. In step 0, the data structures required for the BSPSR and SAEA parameters are initialized, and the generation count is set to 1 ($g = 1$) for the SAEA algorithm. The initial population is generated in step 1, whereas the fitness of the initial population is evaluated in step 2. Then, the algorithm enters the main loop in step 3 and updates the generation counter. Afterwards, the parent selection is implemented to identify the population that will be utilized by the crossover and mutation operators (step 4). Next, the SAEA algorithmic operators are applied to the parent chromosomes to produce the offspring chromosomes (step 5). Moreover, in step 5, all the produced offspring are checked to be repaired utilizing the repairing operator in case of existing any infeasible offspring. The fitness values of the repaired offspring chromosomes are calculated in step 6. Then, a selection operator is implemented to select the offspring chromosomes for the next generation (step 7). Considering the limited number of the generations (g^{lim}) as the stopping criterion, the SAEA algorithm is terminated once the criterion is met. The algorithm will return the best discovered solution (which leads to the berth schedule with the minimum sum of the total weighted vessel turnaround time and the total weighted vessel late departures) at termination.

5.2.2. Chromosome representation

The chromosomes (or individuals) are adopted in EAs to present a solution for the developed BSPSR (Eiben and Smith, 2003; Sivanandam and Deepa, 2008). The chromosomes adopted in SAEA are hybrid, as they are composed of an integer-coded part and a real-coded part. The integer-coded part of each chromosome is used to represent the assignment of a vessel to a berthing position to a service order (i.e., the BSPSR solution). On the other hand, the real-coded part of the chromosomes will be utilized to denote the crossover and mutation probabilities (that will be referred to as p^c and p^m, respectively). The chromosome components (e.g., berthing position identifiers, vessel identifiers, crossover and mutation probabilities) will be called as

"genes" (Eiben and Smith, 2003). The location of a gene along the chromosome will be referred to as "locus" (Eiben and Smith, 2003), where the value of a gene will be recognized as "allele" (Eiben and Smith, 2003). Considering the deployment of the self-adaptive parameter control strategy, the adopted chromosome representation is not common among canonical EAs, which have been used in the BSP literature. Figure 14 illustrates an example of the SAEA chromosome representation.

In Figure 14, a schedule for serving 8 vessels at the MCT is presented. Vessels "3", "2", and "7" are assigned to berth "1" (in that specific service order); whereas berth "2" is assigned to serve vessels "5", "8", "1", "6", and "4" (in that specific service order). Furthermore, the crossover probability of the presented chromosome is $p^c = 0.50$, while the mutation probability of each gene of the presented chromosome is $p^m = 0.05$. The length of each chromosome in the population can be determined as $|I| + 2$, where $|I|$ is the number of vessels, calling for service at the MCT, and "2", is the additional 2 genes required to store the information related to the crossover and mutation probabilities.

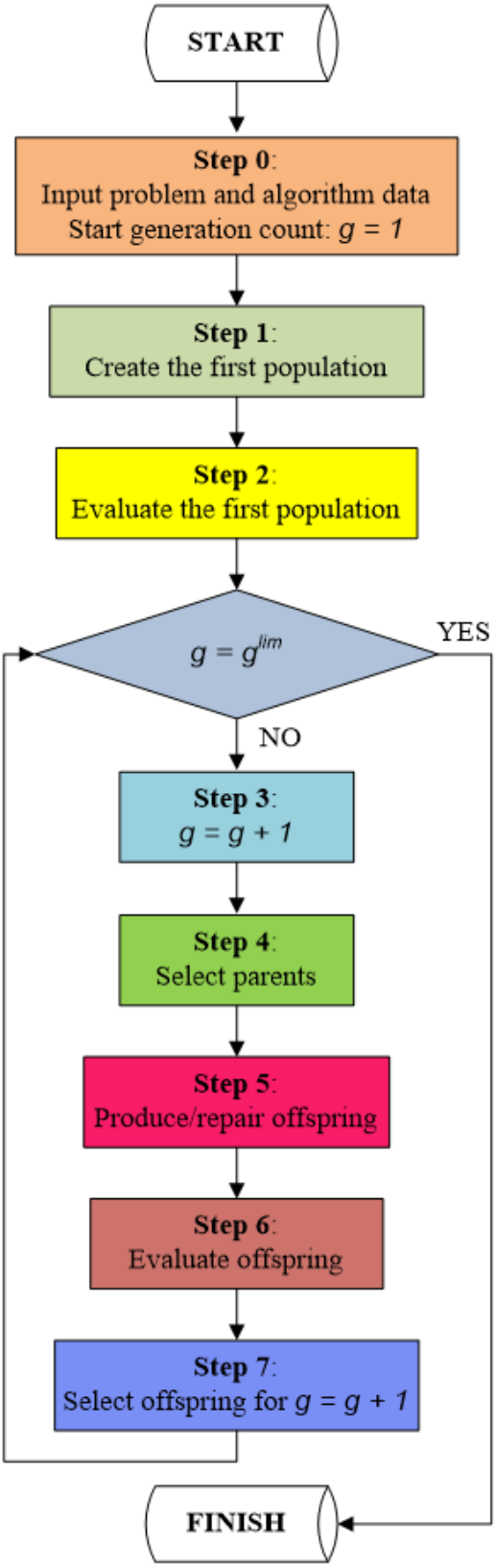

Figure 13 The main SAEA steps.

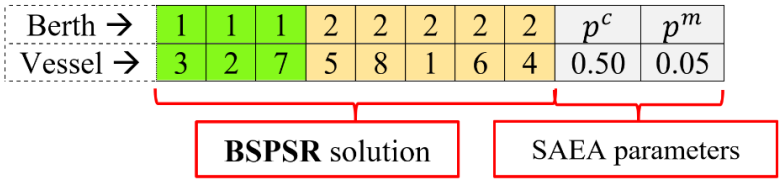

Figure 14 An example of the SAEA chromosome representation.

5.2.3. Population initialization

The first half of the SAEA population will be generated randomly (i.e., the assignment of vessels to berthing positions will be done randomly, where two random values, ranging between 0.01 and 1.00, will be chosen as the crossover and mutation probabilities). A local search heuristic will be developed to initialize the second half of the SAEA population for assigning the vessels to berthing positions to service orders, whereas the crossover and mutation probabilities will be still generated randomly. The local search heuristic is developed considering the First Come First Served (FCFS) policy. Regarding the spatial requirements, considered for the BSPSR mathematical model, a typical FCFS policy, which has been widely used in the BSP literature, will not be applicable in this chapter of the dissertation (i.e., a vessel cannot be assigned to the berthing position, which has smaller length and/or depth).

A FCFS with spatial requirements (FCFS-SR) heuristic is developed in this dissertation, which guarantees that the spatial requirements are not violated for the generated berth schedules. Algorithm 1 presents the main steps of the FCFS-SR heuristic. Notation $\widetilde{\boldsymbol{T}^B}$ is adopted in Algorithm 1 to denote the updated availability of berthing positions (measured in hrs.). Other than that, the rest of notations are adopted from section 5.1 of the dissertation. The data structures required for the FCFS-SR heuristic are initialized in step 0. In step 1, the arriving vessels at the MCT are sorted based on their arrival times. It is assumed that the availability of berthing positions is equal to the initial availability (step 2). Afterwards, the iterative procedure of the FCFS-SR heuristic is started (steps 4-13). In steps 5 and 6, the first available berthing position, which can accommodate the next arriving vessel (and satisfy both horizontal and vertical clearance requirements), is recognized. The first available berthing position is assigned the earliest service order in step 7. After that, the first available berthing position is considered to serve the vessel, which satisfies both horizontal and vertical clearance requirements, in the earliest service order

(step 8). The start and finish service times of vessels are calculated in steps 9 and 10. Considering the vessel finish service time, the availability of a berthing position is updated in step 11. The iterative procedure is terminated by the FCFS-SR heuristic, once all the arriving vessels at the MCT are assigned to the berthing positions to service orders.

Algorithm 1: First Come First Served with Spatial Requirements (FCFS-SR) Heuristic

$\boldsymbol{FCFS\text{-}SR}(I, J, K, T^B, T^A, T^{HT}, L^V, L^B, D^H, H^V, H^B, D^V)$

in: $I = \{1, \dots, m\}$ - set of vessels; $J = \{1, \dots, n\}$ - set of berths; $K = \{1, \dots, s\}$ - set of service orders; T^B - berth availability; T^A - vessel arrival times; T^{HT} - vessel handling times; L^V - length of vessels; L^B - length of berths; D^H - horizontal clearance requirements; H^V - draft of vessels; H^B - depth of berths; D^V - vertical clearance requirements

out: $\boldsymbol{x}$ - initial vessel to berth to service order assignment

0: $|\tilde{I}| \leftarrow m;\ |\boldsymbol{x}| \leftarrow m \cdot n \cdot s;\ |\widetilde{\boldsymbol{T}^B}| \leftarrow n;\ |\boldsymbol{T}^{ST}| \leftarrow m;\ |\boldsymbol{T}^{FT}| \leftarrow m$ ◁ Initialization

1: $\tilde{I} \leftarrow \boldsymbol{Sort}(I, T^A)$ ◁ Sort the vessels based on their arrival times at the MCT

2: $\widetilde{\boldsymbol{T}^B} \leftarrow T^B$ ◁ Set the initial availability of berthing positions

3: $i \leftarrow 1$

4: **for all** $i \in \tilde{I}$ **do**

5: $b \leftarrow \boldsymbol{find}(L^B \geq L_i^V + D_i^H \ \ \boldsymbol{and} \ \ H^B \geq H_i^V + D_i^V)$ ◁ Identify potential berthing positions for a vessel

6: $j \leftarrow argmin_b(\widetilde{\boldsymbol{T}_b^B})$ ◁ Select the first available berthing position

7: $k \leftarrow argmin_k(\boldsymbol{x}_{ijk})$ ◁ Determine the earliest service order

8: $\boldsymbol{x}_{ijk} \leftarrow 1$ ◁ Assign a vessel to the first available berthing position in the earliest service order

9: $\boldsymbol{T}_i^{ST} \leftarrow max(T_i^A; \widetilde{\boldsymbol{T}_j^B})$ ◁ Calculate the start vessel service time

10: $\boldsymbol{T}_i^{FT} \leftarrow \boldsymbol{T}_i^{ST} + T_{ij}^{HT}$ ◁ Calculate the finish vessel service time

11: $\widetilde{\boldsymbol{T}_j^B} \leftarrow \boldsymbol{T}_i^{FT}$ ◁ Update the availability of a berthing position

12: $i \leftarrow i + 1$

13: **end for**

14: **return** $\boldsymbol{x}$

Considering the deterministic nature of the FCFS-SR heuristic, all the generated chromosomes (solutions), which represent the assignment of vessels to berthing positions to service orders, will be identical. The latter will negatively affect the population diversity in the initial population. In order to increase the population diversity and boost the SAEA explorative capabilities, only half of the initial population is generated adopting the FCFS-SR heuristic, whereas another half will be generated randomly. The parameter tuning analysis, which is

described in section 5.4.2 of the dissertation, was adopted to determine the number of chromosomes in the population (i.e., population size – Θ).

5.2.4. Fitness function

The set of individuals in the population and the set of generations are defined as $Q = \{1, \dots, a\}$ and $G = \{1, \dots, b\}$, respectively. The fitness value of individual q in generation g (Fit_{qg}) is computed within the developed SAEA algorithm based on the following relationship:

$$Fit_{qg} = \boldsymbol{Z}_{qg} + \alpha\Psi_{qg}^{H} + \beta\Psi_{qg}^{V} \; \forall q \in Q, g \in G \tag{14}$$

According to equation (14), the fitness function includes the objective value ($\boldsymbol{Z}_{qg}$) plus the penalty term for violation of the horizontal clearance requirements ($\alpha\Psi_{qg}^{H}$, where α – penalty coefficient for horizontal clearance violation, measured in hrs./ft.; Ψ_{qg}^{H} – horizontal clearance violation for individual q in generation g, measured in ft.) and the penalty term for violation of the vertical clearance requirements ($\beta\Psi_{qg}^{V}$, where β – penalty coefficient for vertical clearance violation, measured in hrs./ft.; Ψ_{qg}^{V} – vertical clearance violation for individual q in generation g, measured in ft.). Implementation of the SAEA operations may generate some infeasible solutions; therefore, the penalty terms are defined to adjust the fitness function values for infeasible individuals (that violate the horizontal and/or vertical clearance requirements for vessels). Equation (15) was formulated to estimate the horizontal clearance violation for individual q in generation g (which is a sum of horizontal clearance violations of all the vessels that are served at the MCT berthing positions):

$$\Psi_{qg}^{H} = \sum_{i \in I} \sum_{j \in J} \sum_{k \in K} max\{0; [(L_i^V + D_i^H) - L_j^B]\boldsymbol{x}_{ijk}\} \; \forall q \in Q, g \in G \tag{15}$$

Similar to the horizontal clearance violation, the vertical clearance violation for individual q in generation g can be estimated using the similar equation as follows:

$$\Psi_{qg}^{V} = \sum_{i \in I} \sum_{j \in J} \sum_{k \in K} max\left\{0; [(H_i^V + D_i^V) - H_j^B]\boldsymbol{x}_{ijk}\right\} \; \forall q \in Q, g \in G \tag{16}$$

Based on equations (15) and (16), the higher the violation, the worse the fitness value; therefore, chances of the individuals to survive will increase along with decreasing degree of violation of horizontal and/or vertical clearance requirements. Determining a set of appropriate values for α and β (i.e., the penalty coefficients) has a high level of importance to effectively decrease the chance of survival of the infeasible individuals as compared to the feasible ones. The parameter tuning analysis was conducted to set the appropriate values for penalty coefficients α and β (which is described in section 5.4.2 of the dissertation).

5.2.5. Parent selection procedure

The first operation in the main loop of SAEA is the parent selection, which is conducted to determine the chromosomes for participating in the SAEA operations and offspring production in a given generation. A Roulette Wheel Selection, as one of the most commonly used selection operators in canonical Genetic Algorithms and Genetic Programming (Eiben and Smith, 2003), was adopted as the parent selection operator. The Roulette Wheel Selection (RWS) mechanism is based on assigning a portion of a roulette wheel to each individual in the population. The better the fitness value, the larger the assigned portion of the roulette wheel to a given individual (i.e., as the proposed BSPSR mathematical model is a minimization model, the individuals with lower objective function values should be assigned a larger portion of the roulette wheel). In each complete rotation of the roulette wheel, one individual is selected, and the roulette wheel is rotated until the required number of individuals are selected. Algorithm 2 outlines the main steps of the RWS mechanism.

The data structures required for the RWS mechanism are initialized in steps 0 and 1. In steps 2-5, the fitness values of the individuals are adjusted (i.e., Fit_g^{aux} values are estimated), considering the fact that the BSPSR mathematical model has a minimization objective function. Then, the normalization of the adjusted fitness values of the chromosomes is conducted in step 6 (note that the cumulative population fitness values after normalization will be equal to 1.00). Afterwards, an iterative procedure (steps 7-11) is started by RWS, where a random value in the range [0 1] is generated in step 8 (this is how the rotation of the roulette wheel is modeled). In step 9, the individual, whose portion in the roulette wheel covers the random value, generated in the

previous step, is selected as a parent (steps 9 and 10). Once the required number of parent chromosomes are selected, the iterative procedure is terminated by RWS.

Algorithm 2: Roulette Wheel Selection (RWS)

$\boldsymbol{RWS}(Pop_g, Fit_g)$

in: Pop_g - population in generation g; Fit_g - fitness of chromosomes in generation g

out: $Parents_g$ - parent chromosomes in generation g

0: $|Parents_g| \leftarrow \emptyset$; $|Fit_g^{aux}| \leftarrow |Pop_g|$; $|\overline{Fit}_g^{aux}| \leftarrow |Pop_g|$ ◁ Initialization

1: $q \leftarrow 1$

2: **while** $q \leq |Pop_g|$ **do**

3: $Fit_{qg}^{aux} \leftarrow 1/Fit_{qg}$ ◁ Adjust the fitness value of a given chromosome

4: $q \leftarrow q + 1$

5: **end while**

6: $\overline{Fit}_g^{aux} \leftarrow \boldsymbol{Normalize}(Fit_g^{aux})$ ◁ Normalize the adjusted fitness values of the population chromosomes

7: **while** $|Parents_g| < |Pop_g|$ **do**

8: $Val \leftarrow \boldsymbol{Rand}(0.00; 1.00)$ ◁ Generate a random value between 0.00 and 1.00

9: $i \leftarrow \boldsymbol{find}(\overline{Fit}_g^{aux} - Val > 0)$ ◁ Select the individual based on a randomly generated value

10: $Parents_g \leftarrow Parents_g \cup \{Pop_{ig}\}$ ◁ The selected individual is added to the set of parent chromosomes

11: **end while**

12: **return** $Parents_g$

5.2.6. SAEA operations

At this step, after the selection of the parent chromosomes by the SAEA algorithm, the offspring chromosomes will be produced by executing the crossover and mutation operators. As the adopted hybrid chromosome representation contains both integer- and real-coded genes, two custom operators were designed to accomplish the crossover and mutation operations. A detailed description of both crossover and mutation operations is presented later in this section.

5.2.6.1.Crossover operation

The crossover operation gives the EA algorithm the capability of exploration in various domains of the search space (Eiben and Smith, 2003; Sivanandam and Deepa, 2008). Considering the adopted hybrid chromosome representation (where integer numbers are deployed to represent the vessel to berthing position to service order assignment, while the crossover and mutation

probabilities are defined using real-valued numbers), typical crossover operators, which have been commonly used in the BSP literature (e.g., partially mapped crossover, one-point crossover, two-point crossover), will not be applicable for the proposed SAEA, as they may generate infeasible offspring chromosomes. The order crossover and the whole arithmetic crossover were combined to customize the crossover operator developed for SAEA. Figure 15 depicts an example of the SAEA crossover operation, where two parent chromosomes are selected at random from the available parent chromosomes. Note that the crossover probability of a given parent chromosome (i.e., the probability to apply the crossover operation to the parent chromosome) is determined by parameter p^c, which is encoded in each chromosome of the population (since the proposed SAEA algorithm is self-adaptive).

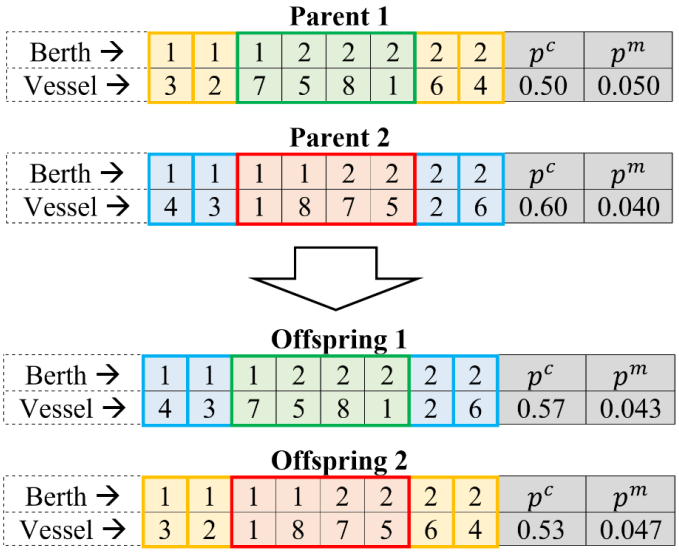

Figure 15 An example of the SAEA crossover operation.

In the developed crossover operator, the integer portion of the parent chromosomes undergoes the order crossover. At the beginning, the same segment of the integer portion of chromosome in two parent chromosomes is selected. In Figure 15, a portion of parent "1" with vessels "7", "5", "8", and "1" is copied to offspring "1". The length of the segment is chosen randomly by SAEA, and therefore, may vary from one crossover operation to another. Then, the genes, which are not included in the copied portion from offspring "1" are copied from parent "2". In the example, provided in Figure 15, the genes with vessels "4", "3", "2" and "6" are copied from parent "2" to offspring "1". The same procedure is implemented to generate the integer

portion of offspring "2". After generating the integer portion of the offspring chromosomes, the whole arithmetic crossover is adopted by the SAEA algorithm for the real-valued portion of the parent chromosomes. Two equations (17) and (18) are deployed to implement the crossover operator for the real-coded portion of the parent chromosomes:

$$Y_1 = aX_1 + (1 - a)X_2 \quad (17)$$

$$Y_2 = (1 - a)X_1 + aX_2 \quad (18)$$

where X_1 and X_2 are the alleles of the parent chromosomes, and Y_1 and Y_2 are the alleles of the offspring chromosomes. Note that X_1, X_2, Y_1, and Y_2 are corresponding real values for the crossover and mutation probabilities in the parent and offspring chromosomes. The offspring gene values Y_1 and Y_2 will be estimated according to the parent gene values X_1 and X_2, and randomly generated value a ($a \in [0.00; 1.00]$). As it is illustrated in Figure 15, the value of parameter a was assumed to be $a = 0.30$. Considering equation (17), the crossover probability for offspring "1" should be calculated as follows: $p_1^c = 0.30 \cdot 0.50 + (1 - 0.30) \cdot 0.60 = 0.57$. The same procedure should be conducted for calculating the crossover probability for offspring "2": $p_2^c = (1 - 0.30) \cdot 0.50 + 0.30 \cdot 0.60 = 0.53$. Furthermore, similar to the calculated crossover probabilities for the two offspring, the mutation probability for offspring "1" and "2" in the considered example was estimated as follows: $p_1^m = 0.30 \cdot 0.050 + (1 - 0.30) \cdot 0.040 = 0.043$; and $p_2^m = (1 - 0.30) \cdot 0.050 + 0.30 \cdot 0.040 = 0.047$. The parameter a will be set based on this equation: $a = U[0.00; 1.00]$ for the whole arithmetic crossover, where its value will vary within SAEA from one operation to another. Note that the notation $U[Val_1; Val_2]$ is used for the uniformly distributed pseudorandom numbers that vary between Val_1 and Val_2.

5.2.6.2. Mutation operation

The mutation operator is applied to each one of the offspring chromosomes produced via the crossover operation. Although the crossover operator explores the search space, the mutation exploits the identified domains of the search space in order to discover the solutions with better fitness values (Eiben and Smith, 2003; Sivanandam and Deepa, 2008). As the adopted chromosomes in this chapter of the dissertation are two-dimensional hybrid ones, similar to the crossover operation, typical mutation operators, which have been widely used in the BSP literature

(e.g., swap, insert, scramble, invert), may generate infeasible offspring chromosomes. Therefore, they could not be utilized for the adopted hybrid chromosome, and a custom mutation operator was proposed in this chapter of the dissertation. The main steps of the developed custom mutation operator are presented in Algorithm 3.

Algorithm 3: Custom Mutation Operator

$\boldsymbol{Mutation}(Offspring_g, p^m)$

in: $Offspring_g$ - offspring chromosomes in generation g; p^m - mutation probability

out: $\widetilde{Offspring}_g$ - mutated offspring chromosomes in generation g

0: $|\widetilde{Offspring}_g| \leftarrow |Offspring_g|$ ◁ Initialization

1: $q \leftarrow 1$

2: **while** $q \leq |Offspring_g|$ **do**

3: $\widetilde{Offspring}_{qg}^{p^c} \leftarrow \boldsymbol{Float}(Offspring_{qg}^{p^c}, p^m)$ ◁ Mutate the crossover probability

4: $\widetilde{Offspring}_{qg}^{p^m} \leftarrow \boldsymbol{Float}(Offspring_{qg}^{p^m}, p^m)$ ◁ Mutate the mutation probability

5: $Val_1 \leftarrow \boldsymbol{Rand}(0.00; 1.00)$ ◁ Generate a random value between 0.00 and 1.00

6: $Val_2 \leftarrow \boldsymbol{Rand}(0.00; 1.00)$ ◁ Generate a random value between 0.00 and 1.00

7: **if** $Val_1 \leq 0.50$ **then**

8: **if** $Val_2 \leq 0.50$ **then**

9: $\widetilde{Offspring}_{qg}^{berth} \leftarrow \boldsymbol{Insert}(Offspring_{qg}^{berth}, p^m)$ ◁ Mutate the genes with berthing positions

10: **else**

11: $\widetilde{Offspring}_{qg}^{vessel} \leftarrow \boldsymbol{Swap}(Offspring_{qg}^{vessel}, p^m)$ ◁ Mutate the genes with vessel identifiers

12: **end if**

13: **else**

14: **if** $Val_2 \leq 0.50$ **then**

15: $\widetilde{Offspring}_{qg}^{berth} \leftarrow \boldsymbol{Swap}(Offspring_{qg}^{berth}, p^m)$ ◁ Mutate the genes with berthing positions

16: **else**

17: $\widetilde{Offspring}_{qg}^{vessel} \leftarrow \boldsymbol{Insert}(Offspring_{qg}^{vessel}, p^m)$ ◁ Mutate the genes with vessel identifiers

18: **end if**

19: **end if**

20: $q \leftarrow q + 1$

21: **end while**

22: **return** $\widetilde{Offspring}_g$

The necessary data structure for implementing the mutation operation is initialized in step 0. Afterwards, the main loop of the custom mutation operator starts in step 2 and finishes in step

21. Specifically, the floating point mutation was adopted to mutate the real-valued portion of a given offspring chromosome (i.e., the crossover and mutation probabilities) in steps 3 and 4. Two random values (Val_1 and Val_2) in the range [0 1] are chosen in steps 5 and 6. The generated values Val_1 and Val_2 are used to randomly apply either the swap or insert mutation operators to the row of berthing positions or vessels in the given chromosome (steps 7-19). Once each offspring individual in the population is mutated by the custom mutation operator, the iterative procedure is terminated. Note that just one of the considered mutation operators is applied within the developed custom mutation operator to the berthing positions or to the vessels at a time to prevent significant genetic changes in the chromosomes due to implementing the mutation operator (which may further cause disruption of "building blocks" (Eiben and Smith, 2003; Sivanandam and Deepa, 2008) and lead to the low-quality offspring chromosomes).

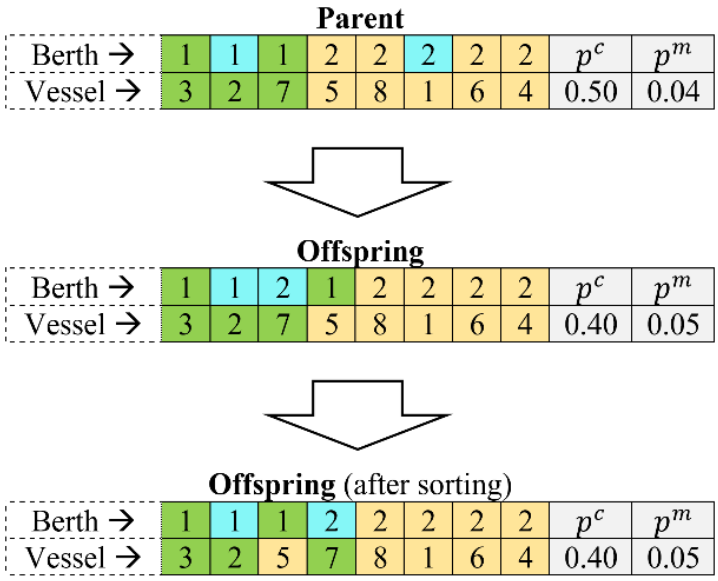

Figure 16 An example of the SAEA mutation operation: insert + floating point.

Figure 16 and Figure 17 provide examples of the SAEA mutation operations. In Figure 16, the floating point and insert mutation operators are applied to the parent chromosome, where the crossover and mutation probabilities are changed using the floating point mutation from 0.50 and 0.04 to 0.40 and 0.05, respectively. Meanwhile, the gene with berthing position "2" in locus "6" is shifted to locus "3" by implementing the insert mutation. In Figure 17, the floating point and swap mutation operators are applied to the parent chromosome. Specifically, the floating point

mutation operator was applied to the crossover and mutation probabilities and altered them from 0.45 and 0.01 to 0.30 and 0.03, respectively. Moreover, two randomly selected genes with vessels "6" and "5" are swapped using the swap mutation (see the integer portion of the offspring, representing the vessel identifiers). The number of genes, which should be changed due to implementing the insert and swap mutation operations, is defined by mutation probability parameter p^m, encoded in each offspring chromosome (since the proposed SAEA algorithm is self-adaptive).

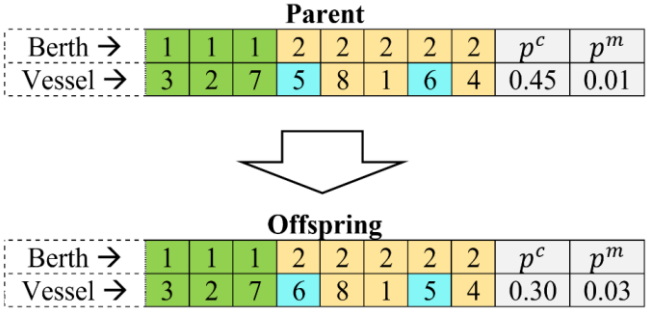

Figure 17 An example of the SAEA mutation operation: swap + floating point.

In the process of implementing the SAEA crossover and mutation operators, some infeasible offspring may be generated. In Figure 16, vessels "2" and "5", which are both assigned for service at berthing position "1", are interrupted by the gene with vessel "7", which is assigned for service at berthing position "2". The latter assignment disrupts the service order of vessels, which are assigned to berthing position "1", and generates an infeasible offspring chromosome. In order to repair the infeasible offspring chromosomes, the genes with vessel identifiers will be sorted by berthing positions after application of the custom mutation operator (see Figure 16). In the provided example, the genes with vessels "7" and "5" exchange their loci, which resolves disruption of the vessel service order at berthing positions "1" and "2".

5.2.7. Offspring selection procedure

After implementing the SAEA operations, all the produced offspring chromosomes are evaluated. Then, the offspring selection procedure is executed by the SAEA algorithm in order to select the chromosomes that will survive in the given generation and will be transferred to the next

generation. A Tournament Selection was deployed in this chapter for the offspring selection. Algorithm 4 represents the main steps of the Tournament Selection mechanism.

Algorithm 4: Tournament Selection

$\boldsymbol{TourSel}(Offpring_g, Fit_g, \Omega, \Phi)$

in: $Offpring_g$ - offspring chromosomes in generation g; Fit_g - fitness of individuals in generation g, Ω - tournament size, Φ - number of individuals selected at each tournament

out: Pop_{g+1} - population chromosomes in generation $g+1$

0: $|Pop_{g+1}| \leftarrow \emptyset$ ◁ Initialization

1: **while** $|Pop_{g+1}| \neq |Offpring_g|$ **do**

2: $[Tour, Fit_g^{Tour}] \leftarrow \boldsymbol{RandSel}(Offpring_g, Fit_g, \Omega)$ ◁ Select the individuals for the tournament

3: $q \leftarrow 1$

4: **while** $q \leq \Phi$ **do**

5: $i \leftarrow argmin(Fit_g^{Tour})$ ◁ Select the fittest individual from the tournament

6: $Pop_{g+1} \leftarrow Pop_{g+1} \cup \{Tour_i\}$ ◁ Add the selected individual to the next generation chromosomes

7: $Tour \leftarrow Tour - \{Tour_i\}$ ◁ Remove the selected individual from the tournament

8: $q \leftarrow q + 1$

9: **end while**

10: **end while**

11: **return** Pop_{g+1}

The data structure required for implementing the offspring selection by the Tournament Selection mechanism is initialized in step 0. In step 2, Ω individuals are randomly selected by the Tournament Selection mechanism (where Ω – tournament size). The selected chromosomes (referred to as $Tour$) and also the fitness values for the selected chromosomes (referred to as Fit_g^{Tour}) are kept in step 2. Then, Φ individuals with the best fitness values (where Φ – number of individuals selected at each tournament) are selected from the tournament to be transferred to the next generation chromosomes (steps 4-9). Once the sufficient number of chromosomes for the next generation are selected, the tournament selection is terminated (steps 1-10). A Binary Tournament Selection was adopted in this chapter, where two individuals are randomly selected from the population at each tournament (i.e., $\Omega = 2$), and only one individual that has better fitness value will be transferred to the next generation ($\Phi = 1$).

5.2.8. Elitism

The scope of this chapter of the dissertation includes the deployment of the "Elitism" strategy, where the fittest individual is stored before implementing the parent selection and SAEA operators. The stored fittest individual will be moved to the next generation along with the selected offspring chromosomes. The application of the selection, crossover, and mutation operators does not guarantee that the fitness of the produced offspring chromosomes will be better as compared to the fitness of the parent chromosomes (e.g., the crossover and mutation operators may disrupt "building blocks" of the parent chromosomes, which may further worsen fitness of the offspring chromosomes [Eiben and Smith, 2003; Sivanandam and Deepa, 2008]). Hence, the Elitism strategy plays a very important role in the design of EAs.

5.2.9. Termination criterion

The maximum number of generations (g^{lim}) was considered as the stopping criterion in the developed SAEA algorithm, which means that the algorithm is terminated once the number of generations is equal to the considered maximum number of generations. This stopping criterion has been commonly used in EAs, proposed in the BSP literature (Nishmura et al., 2001; Imai et al., 2008; Dulebenets, 2017; Dulebenets et al., 2017a; Dulebenets, 2018c).

5.3. Complexity Analysis

The computational complexity analysis of the developed SAEA algorithm is the point of focus in this section of the dissertation. The major algorithmic steps have to be analyzed to determine the computational complexity of the SAEA algorithm. Algorithm 5 outlines the main SAEA steps to be considered as a part of the complexity analysis.

The necessary data structures for executing the SAEA algorithm are initialized in step 0. In step 1, the values of the generation and counter are set to "1". The initial population is produced in steps 2-6 (note that half of the population is generated utilizing the FCFS-SR heuristic, and another half is generated randomly – refer to section 5.2.3 of the dissertation for more details). The fitness values of the initial population chromosomes are calculated in step 7. Afterwards, the iterative procedure of SAEA (steps 8-18) is started. The generation counter is updated in step 9. For implementing the elitism strategy, a copy of the fittest individual is stored in step 10 before

applying any algorithmic operators. In step 11, the parent chromosomes are selected utilizing the RWS mechanism. In order to produce the offspring chromosomes, the crossover operation is executed in step 12.

Algorithm 5: Self-Adaptive Evolutionary Algorithm

$\boldsymbol{SAEA}(\Theta, p^c, p^m, \Omega, \Phi, g^{lim}, Data, I, J, K, T^B, T^A, T^{HT}, L^V, L^B, D^H, H^V, H^B, D^V)$

in: Θ - population size; p^c - crossover probability; p^m - mutation probability; Ω - tournament size, Φ - number of individuals selected at each tournament; g^{lim} - maximum allowable number of generations; $Data$ - input data for the **BSPSR** mathematical model; $I = \{1, \dots, m\}$ - set of vessels; $J = \{1, \dots, n\}$ - set of berths; $K = \{1, \dots, s\}$ - set of service orders; T^B - berth availability; T^A - vessel arrival times; T^{HT} - vessel handling times; L^V - length of vessels; L^B - length of berths; D^H - horizontal clearance requirements; H^V - draft of vessels; H^B - depth of berths; D^V - vertical clearance requirements

out: $BestSol$ - the best berth schedule

0: $|Pop| \leftarrow \Theta$; $|Fit| \leftarrow \Theta$; $|Parents| \leftarrow \Theta$; $|Offspring| \leftarrow \Theta$; $|\widetilde{Offspring}_g| \leftarrow \Theta$; $|Best| \leftarrow \emptyset$

1: $g \leftarrow 1$; $counter \leftarrow 1$

2: **while** $counter \leq \Theta$ **do**

3: $\boldsymbol{x} \leftarrow \boldsymbol{PopInit}(I, J, K, T^B, T^A, T^{HT}, L^V, L^B, D^H, H^V, H^B, D^V)$ ◁ Generate a chromosome

4: $Pop_g \leftarrow Pop_g \cup \{\boldsymbol{x}\}$ ◁ Append the generated chromosome

5: $counter \leftarrow counter + 1$

6: **end while**

7: $Fit_g \leftarrow \boldsymbol{FitnessEval}(Data, Pop_g)$ ◁ Evaluate fitness of the initial population

8: **while** $g \leq g^{lim}$ **do**

9: $g \leftarrow g + 1$ ◁ Update the generation count

10: $Best \leftarrow argmin(Fit_g)$ ◁ Store a copy of the fittest individual

11: $Parents_g \leftarrow \boldsymbol{RWS}(Pop_g, Fit_g)$ ◁ Identify the parent chromosomes

12: $Offspring_g \leftarrow \boldsymbol{Crossover}(Parents_g, p^c)$ ◁ Produce the offspring chromosomes

13: $\widetilde{Offspring}_g \leftarrow \boldsymbol{Mutation}(Offspring_g, p^m)$ ◁ Mutate the offspring chromosomes

14: $\widetilde{Offspring}_g \leftarrow \boldsymbol{Repair}(\widetilde{Offspring}_g)$ ◁ Repair the mutated offspring

15: $Fit_g \leftarrow \boldsymbol{FitnessEval}(Data, \widetilde{Offspring}_g)$ ◁ Evaluate fitness of the mutated offspring

16: $Pop_{g+1} \leftarrow Pop_{g+1} \cup \{Best\}$ ◁ Transfer the fittest individual to generation $g + 1$

17: $Pop_{g+1} \leftarrow \boldsymbol{TourSel}(\widetilde{Offspring}_g, Fit_g, \Omega, \Phi)$ ◁ Identify the chromosomes for generation $g + 1$

18: **end while**

19: $BestSol \leftarrow argmin(Fit_{Best} \cup Fit_g)$ ◁ Identify the best berth schedule

20: **return** $BestSol$

The mutation function is applied to the generated offspring chromosomes in step 13. All the generated offspring are checked to be repaired in case of infeasibility in step 14. The generated and repaired offspring chromosomes are evaluated in step 15. The elitism strategy is conducted in step 16 to transfer the fittest individual to the next generation. The Tournament Selection mechanism is implemented in step 17 to select a group of offspring chromosomes for the next generation. Upon the satisfaction of the stopping criterion (i.e., reaching a predetermined number of generations), the SAEA algorithm terminates the iterative procedure. The best discovered solution (i.e., representing the berth schedule with the minimum sum of the total weighted vessel turnaround time and the total weighted vessel late departures) is returned in step 20.

Throughout the SAEA pseudocode, a total of 6 major functions are introduced within the main loop (steps 8-18), including the following:

1) $\boldsymbol{RWS}(Pop_g, Fit_g)$ with the computational complexity $O(\Theta)$ – see Algorithm 2 (which does not include any sorting procedures);
2) $\boldsymbol{Crossover}(Parents_g, p^c)$ with the computational complexity $O(\Theta \cdot [|I| + 2])$, i.e. the computational complexity of the developed crossover operator is proportional to both population size (Θ) (as the probability of a given parent to participate in the crossover operation is checked for each existing parent chromosome in the SAEA population) and the chromosome length (defined as $[|I| + 2]$ – see section 5.2.6.1 of the dissertation for more details);
3) $\boldsymbol{Mutation}(Offspring_g, p^m)$ with the computational complexity $O(\Theta \cdot [|I| + 2])$, i.e. the same as the $\boldsymbol{Crossover}$ function, the computational complexity of the developed mutation operator (see Algorithm 3) is a function of the population size (Θ) and the chromosome length;
4) $\boldsymbol{Repair}(\widetilde{Offspring_g})$ with the computational complexity $O(\Theta \cdot log\, \Theta)$ (Sedgewick, 1998), as it needs sorting (see section 5.2.6.2 of the dissertation);
5) $\boldsymbol{FitnessEval}(Data, \widetilde{Offspring_g})$ with the computational complexity $O(\Theta \cdot [|I| + 2])$;
6) $\boldsymbol{TourSel}(\widetilde{Offspring_g}, Fit_g, \Omega, \Phi)$ with the computational complexity $O(\Theta^2)$ – see Algorithm 4.

The scope of this chapter of the dissertation will include the analysis of the large-size problem instances with up to 100 arriving vessels at the MCT (see section 5.4.1 of the dissertation for more details about the considered problem instances), while a group of different population

size values (between 40 to 60 individuals) is adopted throughout the parameter tuning analysis (see section 5.4.2 of the dissertation for more details regarding the parameter tuning analysis). Hence, $\Theta \cdot [|I| + 2] \gg \Theta^2$, and the computational complexity of the SAEA algorithm will be $O(g^{lim} \cdot \Theta \cdot [|I| + 2])$. Note that the computer hardware performance, operating system, programming language adopted, programming skills of the developer, etc. are some other factors that may affect the computational time of the SAEA algorithm along with the computational complexity (Pelusi et al., 2018).

5.4. Numerical Experiments

The numerical experiments, conducted to make a comparison between the proposed SAEA algorithm and the alternative EA algorithms, are described in this section of the dissertation. In particular, performance of the developed SAEA algorithm, which is based on the self-adaptive parameter control strategy, was assessed according to a comparative analysis against the following EA algorithms: (1) Adaptive EA (AEA) – an EA that is structured based on the adaptive parameter control strategy (i.e., the crossover and mutation probabilities are altered based on the feedback from the search and they are not encoded in the chromosomes); (2) Deterministic Parameter Control EA (DPCEA) – an EA that is structured based on the deterministic parameter control strategy (i.e., the crossover and mutation probabilities are altered based on a certain counter without any feedback from the search and they are not encoded in the chromosomes); and (3) typical EA algorithm – an EA that is structured based on the parameter tuning and does not use any parameter control strategies (i.e., the crossover and mutation probabilities do not alter throughout evolution of the EA algorithm and are set based on the parameter tuning). A detailed description of the AEA algorithm is provided in Dulebenets (2018c), where the algorithmic parameters are changed, if the objective function remains constant after a predetermined number of generations (g^{max}). Moreover, an exhaustive explanation about the DPCEA algorithm is provided in Dulebenets (2017), where the algorithmic parameters are updated based on a predetermined piecewise function (i.e., the number of generations is tracked as a counter for changing the algorithmic parameters).

Furthermore, to assess the quality of the solutions obtained for the small-size problem instances, the optimality gap analysis was conducted for all the adopted solution algorithms. The

SAEA, AEA, DPCEA, and EA algorithms were encoded in the MATLAB 2016a environment (Mathworks, 2016a) on a CPU with Dell Intel® Core™ i7 Processor, 32 GB of RAM, and Windows 10 Operating System. The General Algebraic Modeling System (GAMS) environment (GAMS, 2019) was utilized to conduct the optimality gap analysis, and CPLEX was used to solve the BSPSR mathematical model to the global optimality. The following sections of the dissertation elaborate on the input data generation for the BSPSR mathematical model, parameter tuning analysis for the developed solution algorithms, comparative analysis against the exact optimization algorithm, and comprehensive evaluation of the algorithms in terms of various performance indicators (including objective function and computational time values, convergence patterns, evolution of the self-adaptive algorithmic parameter values, algorithmic stability, and changes in the population diversity).

5.4.1. Input data generation

The parameter values selected for the BSPSR mathematical model were extracted from the available BSP literature and relevant online resources (Nishimura et al., 2001; Imai et al., 2008; Cheong et al., 2010; Lu et al., 2011; Yang et al., 2012; Imai et al., 2013; Rodriguez-Molins et al., 2014; Hu, 2015; Frojan et al., 2015; Dulebenets, 2016; Dulebenets, 2017; Dulebenets et al., 2017a,b; Dulebenets, 2018a-g; Dulebenets et al., 2018a,b; Dulebenets, 2019a-c; Abioye et al., 2019; Kavoosi et al., 2019; and Eniram, 2019). Table 6 presents the adopted parameter values. The beginning of the planning horizon was assumed as the first availability time of the berthing positions (i.e., time "0"): $T_j^B = 0 \; \forall j \in J$ (hrs.). It was assumed that the arrival pattern of vessels at the MCT followed an exponential distribution with the average inter-arrival time (dT) of 2 hrs., i.e. $dT = exp[2]$ (hrs.). The latter statistical distribution has been widely used in the BSP literature to model vessel arrivals (Imai et al., 2008; Imai et al., 2013; Dulebenets et al., 2017a). The following relationship was developed to estimate the handling time of vessels at their "preferred berthing positions": $T_i^{HT^*} = U[8; 20] \; \forall i \in I$ (hrs.), where $U[Val_1; Val_2]$ is a notation used for uniformly distributed pseudorandom numbers that vary between Val_1 and Val_2. Each vessel was randomly assigned the "preferred berthing position" from the available berthing positions as follows: $j_i^* = U[1; n] \; \forall i \in I$ (berthing position), where j_i^* is a notation used for "preferred berthing position" of vessel i. It was assumed that the handling time of a given vessel should increase, if the vessel was not assigned for service at its "preferred berthing position" based on the following

relationship: $T_{ij}^{HT} = T_i^{HT^*} \cdot (1 + 0.03 \cdot |j - j_i^*|)\ \forall i \in I, j \in J$ (hrs.). Considering the vessel arrival times and handling times at "preferred berthing positions", the negotiated vessel departure times were generated as follows: $T_i^D = T_i^A + T_i^{HT^*} \cdot U[1.20; 1.50]\ \forall i \in I$ (hrs.).

It was assumed that the following types of vessels were expected to be served by the MCT operator: (1) Panamax (length – 820.2 ft., draft – 41.0 ft.); (2) Panamax Max (length – 951.4 ft., draft – 41.0 ft.); (3) Post Panamax (length – 935.0 ft., draft – 42.7 ft.); (4) Post Panamax Plus (length – 984.3 ft., draft – 47.6 ft.); (5) New Panamax (length – 1200.8 ft., draft – 49.9 ft.); and (6) Triple E (length – 1312.3 ft., draft – 50.9 ft.). The available literature was utilized to adopt the vessel dimensions (Eniram, 2019). In order to set the length of berthing positions, the minimum and maximum length values of the aforementioned vessel types were considered, and following relationships were developed: $L_j^B = 1.20 \cdot min_i(L_i^V) + U[0; 1.20 \cdot max_i(L_i^V) - min_i(L_i^V)]\ \forall j \in J$ (ft.). In a similar way, depth of berthing positions was estimated according to the minimum and maximum draft values of the aforementioned vessel types as follows: $H_j^B = 1.20 \cdot min_i(H_i^V) + U[0; 1.20 \cdot max_i(H_i^V) - min_i(H_i^V)]\ \forall j \in J$ (ft.). All the vessels were assigned the horizontal clearance requirement using the following relationship: $D_i^H = U[50; 100]\ \forall i \in I$ (ft.), where the vertical clearance requirement was set as follows: $D_i^V = U[4; 8]\ \forall i \in I$ (ft.).

The weights assigned to the vessels in the objective function were generated using the following relationship: $W_i = U[0.10; 1.00]\ \forall i \in I$. The large positive number was set to $\Gamma = 10000$. A total of 60 problem instances were developed by changing the number of arriving vessels and the number of available berthing positions considering the generated input data. The following classifications were considered for the generated problem instances: (1) small-size problem instances (1-30), where the number of arriving vessels was changed from 5 to 14 with an increment of one vessel, and the number of available berthing positions was altered from 2 to 4 with an increment of one berthing position; and (2) large-size problem instances (31-60), where the number of arriving vessels was changed from 55 to 100 with an increment of five vessels, and the number of available berthing positions was changed from 4 to 8 with an increment of two berthing positions. Considering the limited capabilities of the exact optimization algorithm to solve the large-size problem instances of the BSPSR mathematical model within a reasonable computational

time, only the small-size problem instances will be used to compare the SAEA, AEA, DPCEA, and EA algorithms against the exact optimization algorithm. However, a comprehensive evaluation of the developed solution algorithms in terms of various performance indicators was undertaken using the large-size problem instances. Sections 5.4.3 to 5.4.8 of the dissertation include the aforementioned analyses.

Table 6 Adopted parameter values.

Parameter	**Selected Value**
Number of arriving vessels: m (vessels)	Varies depending on the problem instance
Number of available berthing positions: n (berthing positions)	Varies depending on the problem instance
Number of vessel service orders: s (service orders)	$s = m$ (assuming that all vessels can be assigned to a berth)
Time when berthing position j becomes available in the planning horizon at the first time: $T_j^B, j \in J$ (hrs.)	$T_j^B = 0 \; \forall j \in J$
Arrival time of vessel i at the MCT: $T_i^A, i \in I$ (hrs.)	$T_{i+1}^A = T_i^A + dT \; \forall i \in I$
Average vessel inter-arrival time: dT (hrs.)	$dT = exp[2]$
Handling time of vessel i at "preferred berthing position": $T_i^{HT^*}, i \in I$ (hrs.)	$T_i^{HT^*} = U[8; 20] \; \forall i \in I$
"Preferred berthing position" for vessel i: $j_i^*, i \in I$ (berthing position)	$j_i^* = U[1; n] \; \forall i \in I$
Handling time of vessel i at berthing position j: $T_{ij}^{HT}, i \in I, j \in J$ (hrs.)	$T_{ij}^{HT} = T_i^{HT^*} \cdot (1 + 0.03 \cdot \lvert j - j_i^* \rvert) \; \forall i \in I, j \in J$
Negotiated departure time of vessel i: $T_i^D, i \in I$ (hrs.)	$T_i^D = T_i^A + T_i^{HT^*} \cdot U[1.20; 1.50] \; \forall i \in I$
Length of vessel i: $L_i^V, i \in I$ (ft.)	$[820.2; 951.4; 935.0; 984.3; 1200.8; 1312.3]$
Length of berthing position j: $L_j^B, j \in J$ (ft.)	$L_j^B = 1.20 \cdot min_i(L_i^V) + U[0; 1.20 \cdot max_i(L_i^V) - min_i(L_i^V)] \; \forall j \in J$
Horizontal clearance requirement for vessel i: $D_i^H, i \in I$ (ft.)	$D_i^H = U[50; 100] \; \forall i \in I$
Draft of vessel i: $H_i^V, i \in I$ (ft.)	$[41.0; 41.0; 42.7; 47.6; 49.9; 50.9]$
Depth of berthing position j: $H_j^B, j \in J$ (ft.)	$H_j^B = 1.20 \cdot min_i(H_i^V) + U[0; 1.20 \cdot max_i(H_i^V) - min_i(H_i^V)] \; \forall j \in J$
Vertical clearance requirement for vessel i: $D_i^V, i \in I$ (ft.)	$D_i^V = U[4; 8] \; \forall i \in I$
Weight of vessel i: $W_i, i \in I$	$W_i = U[0.10; 1.00] \; \forall i \in I$
Large positive number: Γ	10000

5.4.2. Parameter tuning analysis

In order to evaluate performance of the developed SAEA, AEA, DPCEA, and EA algorithms, the values of the constant parameters in these algorithms must be determined utilizing a parameter tuning analysis. The algorithmic parameters, which should be tuned in SAEA are as follows: (1) population size – Θ; (2) penalty terms for infeasible individuals – α and β (see section 5.2.4 of the dissertation for description of the latter parameters); and (3) maximum allowable number of generations – g^{lim}. It was assumed that the adopted penalty terms for violation of the

horizontal and vertical clearance requirements were equal ($\alpha = \beta$). The AEA and DPCEA algorithms have some more parameters to control the crossover and mutation probabilities throughout the search process other than the 3 aforementioned parameters for SAEA. In particular, three more parameters should be tuned for AEA: (1) crossover probability values – p^{cv}; (2) mutation probability values – p^{mv}; and (3) adaptive criterion – g^{max} (i.e., the maximum number of generations without changes in the objective function). The crossover and mutation probabilities will be continuously updating in AEA considering the assigned crossover and mutation probability values (in a sequential manner) based on the feedback from the search (i.e., if no changes in the objective function occurred after a pre-determined number of generations) (Dulebenets, 2018c).

Moreover, three additional parameters should be tuned for DPCEA: (1) crossover probability values at the beginning and at the end of the piecewise function – $\widetilde{p^{cv}}$; (2) mutation probability values at the beginning and at the end of the piecewise function – $\widetilde{p^{mv}}$; and (3) number of segments in the piecewise function – n^{seg}. Figure 18 illustrates an example of how the crossover and mutation probabilities are altered throughout the DPCEA run. In the provided example, the piecewise function includes 3 segments ($n^{seg} = 3$), where $\widetilde{p^{cv}}$ and $\widetilde{p^{mv}}$ values are set to $\widetilde{p^{cv}} = [0.80, 0.40]$ and $\widetilde{p^{mv}} = [0.04, 0.02]$, respectively, and the stopping criterion is set to $g^{lim} = 3000$ generations. Hence, the algorithmic parameters will be set to $p^c = 0.80$ and $p^m = 0.04$ by DPCEA for the first 1000 generations; then, for the next 1000 generations (i.e., from generation 1001 to 2000) the adopted crossover and mutation probabilities will be equal to $p^c = 0.60$ and $p^m = 0.03$; and the crossover and mutation probabilities will be set to $p^c = 0.40$ and $p^m = 0.02$ between generations 2001 and 3000. The EA algorithm has two additional parameters for defining the crossover and mutation probabilities (i.e., p^c and p^m) along with the three SAEA parameters. Note that the crossover and mutation probabilities of EA do not change throughout the search process.

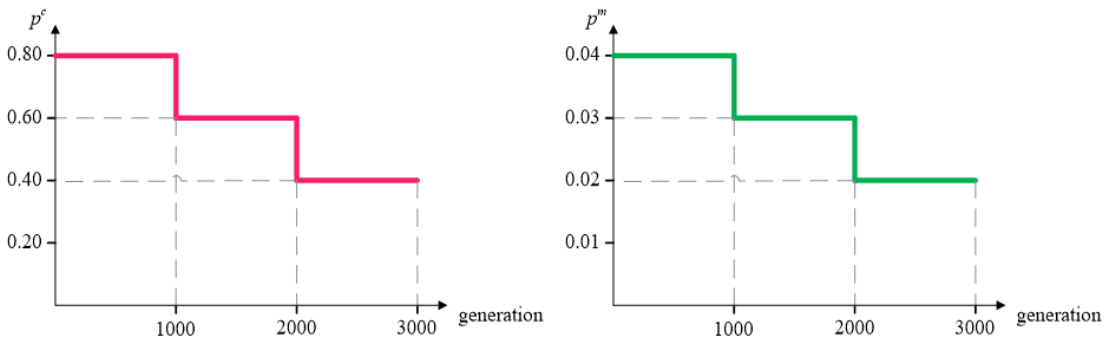

Figure 18 Examples of piecewise functions for the crossover and mutation probabilities within DPCEA.

The parameter tuning analysis was conducted to set the parameter values for the SAEA, AEA, DPCEA, and EA algorithms. In order to perform the parameter tuning analysis, the "full factorial design" methodology was utilized, where each algorithmic parameter (or "factor" – f) had a set of candidate values (or "levels" – l). Each parameter of the developed solution algorithms was tested for a total of three candidate values, which corresponds to 3^f factorial design. In order to perform the parameter tuning, a total of five problem instances were selected at random from the generated large-size problem instances (described in section 5.4.1 of the dissertation). A total of ten replications were conducted to estimate the average values of the objective function and computational time for each parameter combination and each algorithm due to the stochastic nature of the developed solution algorithms.

Table 7 represents the parameter tuning analysis results, which includes the following information: (1) algorithm; (2) parameter description; (3) considered candidate values; and (4) the best value, which was identified based on a tradeoff between the computational time and objective function values. Note that, the SAEA algorithm has the least number of algorithmic parameters, which should be tuned, among all the developed solution algorithms in this chapter of the dissertation. The latter shows the advantage of the self-adaptive parameter control strategy, as the number of algorithmic parameters is reduced, and less computational efforts are needed for conducting the parameter tuning analysis.

Table 7 The parameter tuning analysis results for the developed solution algorithms.

Algorithm	Parameter	Candidate Values	Best Value	Algorithm	Parameter	Candidate Values	Best Value
SAEA	Population size (Θ)	[40; 50; 60]	60	DPCEA	Crossover probability values ($\widetilde{p^{cv}}$)	[0.80, 0.30; 0.90, 0.40; 0.30, 0.80]	[0.80, 0.30]
SAEA	Penalty terms for infeasible individuals (α, β)	[50; 75; 100]	100	DPCEA	Mutation probability values ($\widetilde{p^{mv}}$)	[0.06, 0.01; 0.07, 0.02; 0.01,0.06]	[0.06, 0.01]
SAEA	Maximum allowable number of generations (g^{lim})	[1000; 2000; 3000]	3000	DPCEA	Number of segments in the piecewise function (n^{seg})	[10; 15; 20]	10
AEA	Population size (Θ)	[40; 50; 60]	60	DPCEA	Penalty terms for infeasible individuals (α, β)	[50; 75; 100]	100
AEA	Crossover probability values (p^{cv})	[0.80÷0.30; 0.90÷0.40; 0.30÷0.80][1]	[0.80÷0.30]	DPCEA	Maximum allowable number of generations (g^{lim})	[1000; 2000; 3000]	3000
AEA	Mutation probability values (p^{mv})	[0.06÷0.01; 0.07÷0.02; 0.01÷0.06][2]	[0.06÷0.01]	EA	Population size (Θ)	[40; 50; 60]	60
AEA	Adaptive criterion (g^{max})	[100; 200; 300]	300	EA	Crossover probability (p^{c})	[0.30; 0.50; 0.80]	0.80
AEA	Penalty terms for infeasible individuals (α, β)	[50; 75; 100]	100	EA	Mutation probability (p^{m})	[0.01; 0.04; 0.06]	0.06
AEA	Maximum allowable number of generations (g^{lim})	[1000; 2000; 3000]	3000	EA	Penalty terms for infeasible individuals (α, β)	[50; 75; 100]	100
DPCEA	Population size (Θ)	[40; 50; 60]	60	EA	Maximum allowable number of generations (g^{lim})	[1000; 2000; 3000]	3000

Notes: 1 - the crossover probability values were changed with an increment of 0.10 within the AEA algorithm; 2 - the mutation probability values were changed with an increment of 0.01 within the AEA algorithm.

5.4.3. Comparison against the exact optimization algorithm

A comparative analysis of the developed SAEA, AEA, DPCEA, and EA algorithms against the exact optimization algorithm is presented in this section of the dissertation. The GAMS environment was adopted to encode the BSPSR mathematical model. The latter model was solved utilizing CPLEX, which is a common solution tool in operations research for solving large-scale mixed-integer linear programming models to the global optimality. The relative optimality gap of CPLEX was restricted to 0.01%, whereas the computational time was limited to 7200 sec., and the

rest of settings were left default. All the 30 small-size problem instances (i.e., instances 1-30, explained in section 5.4.1) were solved utilizing CPLEX. In the meantime, the developed small-size problem instances were solved using the SAEA, AEA, DPCEA, and EA algorithms as well. Each one of the small-size problem instances were solved 10 times using each one of the developed algorithms to estimate the average objective function and computational time values. Table 8 summarizes the comparative analysis results, which provides the following information for each small-size problem instance: (1) instance number; (2) number of vessels; (3) number of berths; (4) average objective function value for each solution algorithm; and (5) average computational time (denoted as CPU) for each solution algorithm.

The optimality gap for algorithm a (Δ_a) was estimated utilizing the following relationship: $\Delta_a = \frac{f_a - f_{CPLEX}}{f_{CPLEX}}$, where f_a – average objective function value, provided by algorithm a; f_{CPLEX} – the optimal objective function value, provided by CPLEX. The optimality gap values, estimated for each algorithm and each one of the small-size problem instances, are presented in Figure 19. The results, which are provided in Figure 19, demonstrate that CPLEX was able to find the global optimal solution only for 9 small-size problem instances (1-8 and 10) with up to 8 arriving vessels at the MCT. As the additional spatial requirements were modeled in berth scheduling, the complexity of the mathematical model significantly increased and CPLEX was unable to solve all the small-size problem instances in the restricted computational time (equal to 7200 sec.). For example, Dulebenets (2018c) did not consider the additional spatial requirements in the mathematical model, and CPLEX could solve the problem instances with up to 12 vessels within 7200 sec. According to the optimality gap analysis results presented in Figure 19, the SAEA, AEA, DPCEA, and EA algorithms returned the global optimal solution for the problem instances that were solved by CPLEX within the imposed time limit (i.e., the optimality gaps of the developed solution algorithms were found to be equal to "0" for problem instances 1-8 and 10).

Table 8 Comparison of the developed solution algorithms against CPLEX.

Instance	Number of Vessels	Number of Berths	CPLEX		EA		DPCEA		AEA		SAEA	
			Objective, hrs.	CPU, sec.	Objective, hrs.	CPU, sec.	Objective, hrs.	CPU, sec.	Objective, hrs.	CPU, sec.	Objective, hrs.	CPU, sec.
1	5	2	57.381	1.70	57.381	6.63	57.381	9.38	57.381	10.85	57.381	11.42
2	5	3	42.203	4.52	42.203	6.63	42.203	9.14	42.203	10.76	42.203	11.46
3	5	4	35.923	5.92	35.923	6.62	35.923	9.17	35.923	10.62	35.923	11.08
4	6	2	102.008	13.82	102.008	6.91	102.008	9.59	102.008	11.34	102.008	11.77
5	6	3	72.644	153.00	72.644	6.88	72.644	9.61	72.644	11.32	72.644	11.96
6	6	4	58.514	421.02	58.514	6.88	58.514	9.59	58.514	11.21	58.514	11.67
7	7	2	120.643	187.49	120.643	7.15	120.643	10.00	120.643	11.87	120.643	12.53
8	7	3	86.639	3344.27	86.639	7.13	86.639	10.01	86.639	11.64	86.639	12.03
9	7	4	67.686	7203.40	67.261	7.08	67.261	10.06	67.261	11.43	67.261	12.18
10	8	2	173.560	3878.79	173.560	7.48	173.560	10.40	173.560	12.32	173.560	13.07
11	8	3	116.613	7207.12	115.379	7.39	115.379	10.41	115.379	12.09	115.379	12.83
12	8	4	87.940	7206.69	86.402	7.36	86.402	10.40	86.402	11.97	86.402	12.69
13	9	2	207.567	7207.17	203.697	7.68	203.697	10.83	203.697	12.79	203.697	13.41
14	9	3	138.080	7205.43	133.192	7.68	133.192	10.84	133.192	12.40	133.192	13.17
15	9	4	100.693	7205.23	96.255	7.68	96.255	10.82	96.255	12.41	96.255	12.94
16	10	2	260.008	7206.87	247.721	8.01	247.721	11.22	247.721	13.28	247.721	13.77
17	10	3	168.606	7204.10	160.394	7.97	160.394	11.23	160.394	12.96	160.394	13.62
18	10	4	121.324	7204.73	115.294	8.01	115.294	11.20	115.294	12.89	115.294	13.60
19	11	2	351.252	7204.96	332.783	8.30	332.783	11.63	332.783	14.03	332.783	14.64
20	11	3	234.583	7203.95	218.844	8.25	218.644	11.64	218.644	13.43	218.644	14.49
21	11	4	161.140	7203.51	149.983	8.27	149.703	11.77	149.703	13.25	149.703	13.92
22	12	2	458.371	7202.79	426.017	8.54	425.270	12.06	425.126	13.76	425.126	14.68
23	12	3	290.278	7205.62	267.282	8.56	266.475	12.13	266.188	13.24	266.188	14.41
24	12	4	204.840	7202.84	188.303	8.60	187.840	12.14	187.188	13.20	187.188	14.45
25	13	2	559.488	7202.04	514.104	8.90	513.240	12.49	512.251	14.07	511.228	15.86
26	13	3	336.885	7202.05	310.100	8.84	309.459	12.48	308.718	13.71	307.714	15.13
27	13	4	235.101	7202.10	213.489	8.86	212.951	12.54	212.496	13.75	211.555	14.93
28	14	2	669.643	7201.96	607.726	9.21	606.304	12.94	604.744	14.63	602.522	16.06
29	14	3	431.216	7201.61	391.360	9.11	390.528	12.93	389.541	14.13	387.749	15.57
30	14	4	294.092	7201.42	264.557	9.20	263.990	12.95	262.488	14.19	261.114	15.48
Average:			**208.164**	**5309.87**	**195.322**	**7.86**	**195.077**	**11.05**	**194.833**	**12.65**	**194.554**	**13.49**

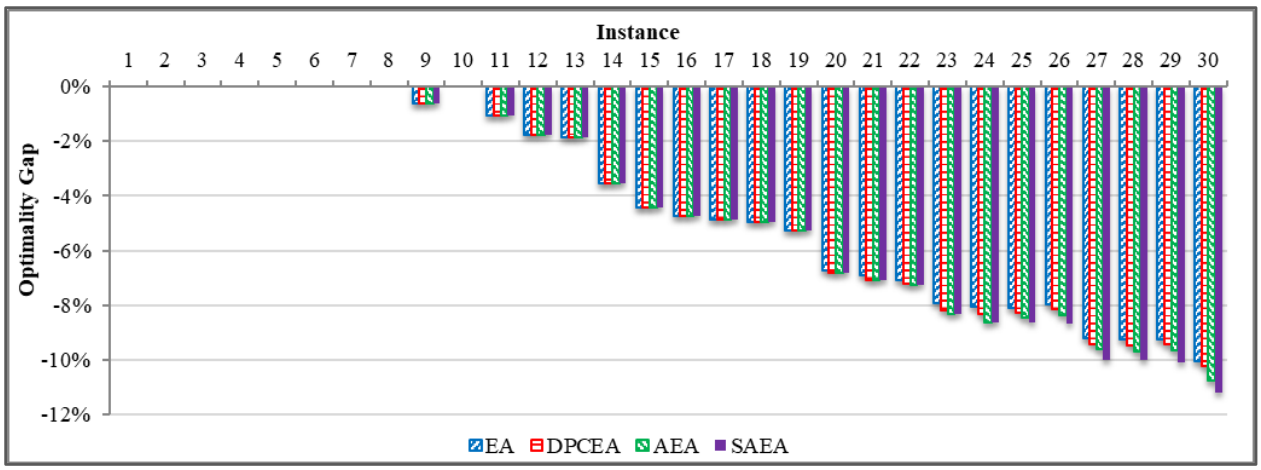

Figure 19 The optimality gaps of the developed solution algorithms.

Besides, CPLEX was not able to solve the rest of small-size problem instances to the global optimality within the time limit imposed (i.e., problem instances 9 and 11-30 in Table 8). Furthermore, the objective function values returned by the developed solution algorithms were superior when comparing to the objective function values obtained by CPLEX after 7200 sec. Therefore, negative optimality gap values were obtained for problem instances 9 and 11-30 (see Figure 19). The numerical experiments, conducted over the generated small-size problem instances, showed the superiority of the SAEA, AEA, DPCEA, and EA algorithms as compared to CPLEX on average by 4.33%, 4.26%, 4.19%, and 4.12%, respectively. The SAEA, AEA, DPCEA, and EA algorithms solved the small-size problem instances in 13.49 sec., 12.65 sec., 11.05 sec., and 7.86 sec., respectively. Considering the results obtained by the optimality gap analysis, it can be concluded that introducing the additional operational constraints in the mathematical model can significantly affect performance of the exact optimization algorithms (e.g., CPLEX, GUROBI, MOSEK), which demonstrates the necessity of developing efficient heuristic and metaheuristic algorithms.

A comprehensive evaluation of the SAEA, AEA, DPCEA, and EA algorithms was conducted using the large-size problem instances. Various performance indicators, including objective function and computational time values, convergence patterns, evolution of the self-adaptive algorithmic parameter values, algorithmic stability, and changes in the population diversity were adopted to compare performance of the developed algorithms against each other. In order to estimate the average values of performance indicators for each algorithm, a total of 10

replications were executed for each one of the large-size problem instances. Results are reported in sections 5.4.4-5.4.8 of the dissertation.

5.4.4. Objective function and computational time values

Two important indicators, which have to be considered in evaluation of the algorithmic performance are objective function and computational time values. The quality of solutions is defined using the objective function values, which is of primary interest to the decision makers (i.e., the MCT operator). However, the practically favorable algorithms do not require a prohibitively large amount of computational time in order to obtain good-quality solutions (e.g., the exact optimization algorithms generally require a remarkably large amount of computational time to solve the problems of a high complexity). Table 9 represents the objective function and computational time values retrieved for all the developed solution algorithms and large-size problem instances. Specifically, the following information for each large-size problem instance is presented in Table 9: (1) instance number; (2) number of vessels; (3) number of berths; (4) average objective function value for each solution algorithm; and (5) average computational time (denoted as CPU) for each solution algorithm.

The results demonstrated that the SAEA algorithm returned the objective function values at termination that were superior to the objective function values, returned by the AEA, DPCEA, and EA algorithms, on average by 4.01%, 6.83%, and 11.84%, respectively, over the considered large-size problem instances. Encoding the crossover and mutation probabilities in the SAEA chromosomes provides these probability values with the capability to evolve throughout the algorithmic run. The latter capability further allows more efficient exploration and exploitation of the search space, which justifies a superior performance of the SAEA algorithm over the other algorithms. The adaptive parameter control strategy, deployed within the AEA algorithm, showed more efficient performance as compared to the deterministic parameter control strategy, adopted within the DPCEA algorithm. The excellence of AEA over DPCEA can be justified by the fact that AEA had been adjusting the crossover and mutation probabilities based on the feedback from the search, while DPCEA had been altering the crossover and mutation probabilities based on the piecewise function ignoring any feedback from the search. The EA algorithm demonstrated the worst performance, as the crossover and mutation probabilities were not altered throughout the

algorithmic run, and the constant values of the latter algorithmic parameters were determined based on the parameter tuning. In conclusion, the explorative and exploitative capabilities of the EA algorithm were significantly limited by constant crossover and mutation probability values.

Table 9 The objective function and computational time values for the developed solution algorithms and large-size problem instances.

Instance	Number of Vessels	Number of Berths	EA		DPCEA		AEA		SAEA	
			Objective, hrs.	CPU, sec.	Objective, hrs.	CPU, sec.	Objective, hrs.	CPU, sec.	Objective, hrs.	CPU, sec.
31	55	4	2377.761	22.21	2258.330	31.40	2211.918	33.54	2164.059	39.32
32	55	6	899.983	21.67	854.218	31.37	825.216	33.72	788.960	39.02
33	55	8	455.552	21.72	447.148	31.50	442.022	34.46	437.173	39.17
34	60	4	2800.193	23.16	2631.300	33.63	2591.382	35.80	2525.668	40.99
35	60	6	1011.343	22.40	975.456	33.83	924.069	35.91	886.146	41.64
36	60	8	496.764	22.38	491.083	34.21	484.617	37.24	479.352	42.03
37	65	4	3205.971	24.41	3019.285	36.10	2988.301	38.02	2896.303	44.05
38	65	6	1119.997	24.43	1065.933	35.95	1030.017	39.85	985.366	43.56
39	65	8	545.141	24.46	538.879	35.90	532.089	41.19	525.534	43.78
40	70	4	3544.719	25.36	3357.861	37.94	3253.702	42.02	3168.028	47.18
41	70	6	1213.662	25.45	1148.926	38.04	1092.158	42.38	1045.119	46.54
42	70	8	572.387	25.60	568.528	38.21	560.240	43.33	556.178	47.27
43	75	4	4034.567	27.17	3740.517	40.11	3681.803	43.28	3544.645	49.66
44	75	6	1324.011	27.01	1248.610	40.16	1197.125	43.64	1113.764	49.39
45	75	8	617.365	26.94	611.086	40.63	602.913	45.16	597.957	49.78
46	80	4	4472.340	28.16	4153.364	42.18	4088.902	45.33	3902.842	51.80
47	80	6	1420.608	28.10	1343.371	42.21	1271.579	45.98	1193.858	52.39
48	80	8	654.161	28.19	646.792	42.37	638.393	47.51	630.520	52.22
49	85	4	5071.830	29.82	4695.141	44.42	4591.242	47.78	4396.188	54.48
50	85	6	1544.441	29.88	1469.839	44.64	1399.679	47.67	1310.340	56.30
51	85	8	717.262	30.00	706.896	44.89	695.856	48.94	687.353	55.98
52	90	4	5566.083	31.30	5197.091	46.69	5052.644	50.32	4791.672	57.59
53	90	6	1718.440	31.41	1616.241	46.72	1557.377	50.94	1443.065	58.82
54	90	8	777.947	31.48	767.060	46.99	751.242	53.37	738.916	59.11
55	95	4	6260.245	32.81	5745.434	48.82	5625.241	52.88	5252.036	61.11
56	95	6	1921.365	32.82	1815.067	48.84	1747.256	53.96	1618.356	60.67
57	95	8	851.889	32.84	840.441	49.14	821.427	56.16	803.427	61.73
58	100	4	7028.538	34.00	6519.833	51.18	6361.047	55.59	5959.020	64.96
59	100	6	2144.635	34.16	2024.188	51.24	1952.334	55.57	1805.210	63.55
60	100	8	939.400	34.30	917.367	51.39	901.137	56.36	874.526	63.67
Average:			**2176.953**	**27.79**	**2047.176**	**41.36**	**1995.764**	**45.26**	**1904.053**	**51.26**

The average computational time over the generated large-size problem instances for the SAEA, AEA, DPCEA, and EA algorithms comprised 51.26 sec., 45.26 sec., 41.36 sec., and 27.79 sec., respectively. An increasing time complexity of the SAEA algorithm can be supported by the fact that SAEA has longer chromosomes - due to encoding additional genes in the chromosomes for the crossover and mutation probabilities. Furthermore, separate algorithmic operators for those additional genes (see section 5.2.6 of the dissertation for more details) imposed an increasing computational time. The maximum SAEA computational time, which did not exceed 64.96 sec.,

can be considered as acceptable from the practical standpoint. Accordingly, the MCT operator will be able to rely on SAEA as a decision support tool to develop berth schedules within a relatively short span of time.

Table 10 The one-way ANOVA analysis results.

Instance	Number of Vessels	Number of Berths	SAEA vs. EA		SAEA vs. DPCEA		SAEA vs. AEA	
			***F*-statistic**	***p*-value**	***F*-statistic**	***p*-value**	***F*-statistic**	***p*-value**
31	55	4	87.333	2.51E-08	15.560	9.50E-04	8.707	8.55E-03
32	55	6	185.969	6.28E-11	127.441	1.34E-09	59.595	4.06E-07
33	55	8	200.004	3.44E-11	45.171	2.66E-06	11.298	3.48E-03
34	60	4	111.444	3.86E-09	32.717	2.01E-05	19.747	3.14E-04
35	60	6	88.500	2.27E-08	168.485	1.41E-10	47.567	1.89E-06
36	60	8	117.671	2.52E-09	61.920	3.10E-07	19.533	3.31E-04
37	65	4	72.669	9.80E-08	45.253	2.63E-06	16.209	7.93E-04
38	65	6	128.470	1.26E-09	186.894	6.02E-11	36.215	1.09E-05
39	65	8	273.436	2.50E-12	142.323	5.55E-10	39.893	5.94E-06
40	70	4	199.458	3.52E-11	102.829	7.20E-09	27.968	4.99E-05
41	70	6	196.358	4.00E-11	122.740	1.81E-09	17.452	5.66E-04
42	70	8	162.601	1.89E-10	54.086	7.95E-07	8.545	9.08E-03
43	75	4	178.851	8.65E-11	58.665	4.53E-07	48.762	1.60E-06
44	75	6	532.901	7.99E-15	162.142	1.93E-10	38.115	7.91E-06
45	75	8	149.522	3.73E-10	78.246	5.69E-08	10.726	4.21E-03
46	80	4	250.318	5.27E-12	41.776	4.42E-06	26.552	6.68E-05
47	80	6	126.959	1.38E-09	160.559	2.09E-10	40.397	5.48E-06
48	80	8	291.768	1.44E-12	155.394	2.73E-10	24.430	1.05E-04
49	85	4	113.627	3.31E-09	56.399	5.96E-07	23.833	1.20E-04
50	85	6	208.291	2.45E-11	158.601	2.31E-10	63.882	2.48E-07
51	85	8	333.197	4.63E-13	160.421	2.11E-10	45.943	2.38E-06
52	90	4	149.621	3.71E-10	141.627	5.77E-10	39.692	6.13E-06
53	90	6	279.916	2.05E-12	193.527	4.52E-11	45.672	2.48E-06
54	90	8	116.989	2.64E-09	75.017	7.77E-08	21.291	2.15E-04
55	95	4	245.548	6.19E-12	130.574	1.10E-09	62.830	2.79E-07
56	95	6	593.439	3.12E-15	78.704	5.45E-08	53.649	8.41E-07
57	95	8	294.788	1.32E-12	145.754	4.58E-10	47.897	1.81E-06
58	100	4	185.464	6.42E-11	174.531	1.06E-10	30.213	3.21E-05
59	100	6	293.450	1.37E-12	148.309	3.98E-10	74.222	8.40E-08
60	100	8	136.192	7.89E-10	123.778	1.69E-09	32.833	1.97E-05
Average:			**210.158**	**5.43E-09**	**111.648**	**3.27E-05**	**34.789**	**9.33E-04**

In addition to the analysis of the average objective function and computational time values, a detailed statistical analysis was conducted for the objective function values, obtained over ten replications for each solution algorithm and each large-size problem instance. In detail, the one-way analysis of variance (ANOVA) was conducted. The null hypothesis of ANOVA was stated as follows: “the objective function values, obtained over ten replications by the SAEA algorithm, were equal to the objective function values, obtained over ten replications by an alternative

algorithm (i.e., EA, DPCEA, and AEA), for a given problem instance". The one-way ANOVA analysis was conducted for three tests for each large-size problem instance, including: (i) SAEA vs. EA; (ii) SAEA vs. DPCEA; and (iii) SAEA vs. AEA. Table 10 includes the results obtained by conducting the one-way ANOVA analysis, which provides the following information for each large-size problem instance: (1) instance number; (2) number of vessels; (3) number of berths; (4) F-statistic for each test; and (5) p-value for each test.

In the implemented one-way ANOVA analysis, F-statistic is defined as the ratio of the mean squared errors (errors quantify the differences between the objective function values, which were obtained by the algorithms), while p-value represents the probability that the test statistic may take a value greater than the value of the computed test statistic. The results demonstrate fairly small p-values for all ANOVA tests and all the generated large-size problem instances (i.e., the maximum p-value did not exceed 9.08E-03). It can be concluded from the latter finding that the objective function values, obtained over ten replications by the SAEA algorithm, were statistically lower than the objective function values, obtained over ten replications by the alternative algorithms (i.e., EA, DPCEA, and AEA), for all the large-size problem instances (i.e., the null hypothesis of the ANOVA test was rejected).

5.4.5. Convergence patterns

One of the other performance indicators for evaluating the developed algorithms is the analysis of algorithmic convergence, where it allows to monitor changes in the objective function values from the population initialization step until termination of the algorithm. The convergence pattern analysis was implemented for all the developed solution algorithms and large-size problem instances. The convergence patterns of the SAEA, AEA, DPCEA, and EA algorithms for the first replication of problem instances 49-60 are illustrated in Figure 20 (12 large-size problem instances with the largest amount of vessels, calling for service at the MCT). Note that similar tendencies were observed in the algorithmic convergence patterns for all the other developed problem instances.

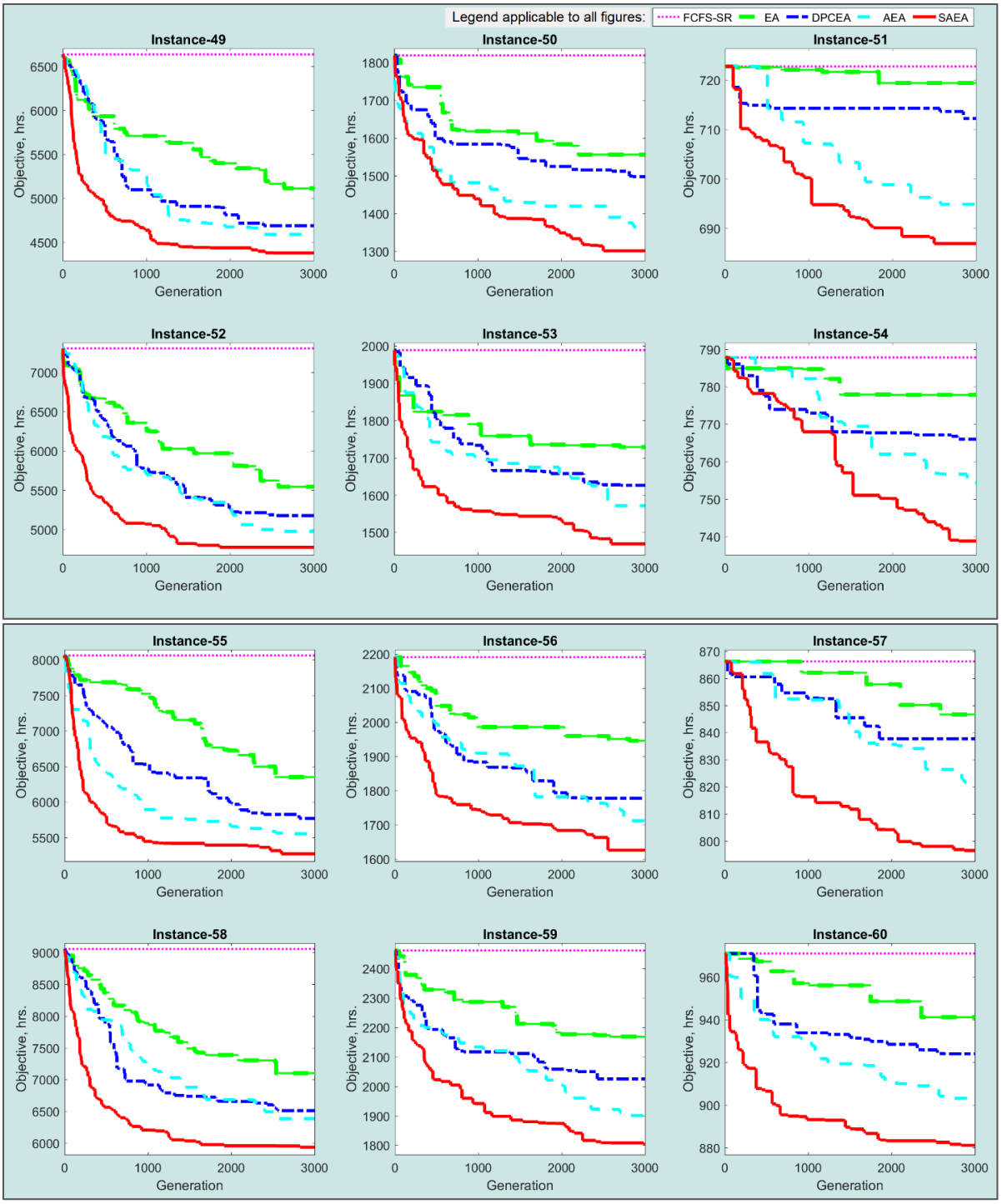

Figure 20 The algorithmic convergence patterns for problem instances 49-60.

As the SAEA, AEA, DPCEA, and EA algorithms deploy the same population initialization mechanism (described in section 5.2.3 of the dissertation), then all the developed solution algorithms started the search process from the same point in the convergence pattern diagrams. It

can be observed that the SAEA algorithm discovered the promising domains of the search space much faster as compared to the AEA, DPCEA, and EA algorithms. Such finding can be supported due to the application of the self-adaptive parameter control strategy for the crossover and mutation probabilities. Adjusting the crossover and mutation probabilities based on the feedback from the search by the AEA algorithm, enables it to move along the search space more efficiently as compared to the DPCEA algorithm. Besides, it can be observed from the analysis of algorithmic convergence patterns that setting the constant values for the crossover and mutation probabilities (i.e., the parameter tuning strategy, adopted within the EA algorithm) significantly worsens the capability of discovering more promising areas in the search space, and accordingly leads to finding low-quality solutions at termination.

A detailed analysis of the convergence speed was conducted for all the developed solution algorithms in this chapter of the dissertation. The convergence speed was evaluated based on the convergence rate value for each solution algorithm (Pelusi et al. 2018). According to equation (19), the convergence rate of algorithm a (CR_a) was calculated based on the number of fitness function evaluations until finding its minimum value (NF_a^*), and the total number of fitness function evaluations (NF_a^{TOT}) (Pelusi et al., 2018).

$$CR_a = \frac{NF_a^*}{NF_a^{TOT}} \tag{19}$$

The convergence rate values were calculated for all the developed solution algorithms, performed replications, and large-size problem instances. Table 11 presents the results including the following information for each large-size problem instance: (1) instance number; (2) number of vessels; (3) number of berths; (4) average convergence rate value over ten replications (denoted as $avg(CR)$) for each solution algorithm; and (5) convergence rate standard deviation over ten replications (denoted as $std(CR)$) for each solution algorithm.

The average convergence rate values over the performed replications and generated large-size problem instances for the SAEA, AEA, DPCEA, and EA algorithms comprised 0.916, 0.954, 0.889, and 0.860, respectively. As the convergence rate values are fairly large for all the solution algorithms, it can be indicated that all the developed algorithms kept discovering superior solutions

throughout the search process. However, considering the convergence rate values, the EA algorithm performance was not as promising as the other developed algorithms. Furthermore, although relatively large convergence rate values were recorded for all the developed solution algorithms, the SAEA algorithm was able to discover more promising solutions for all the large-size problem instances (as discussed in section 5.4.4 of the dissertation).

Table 11 The convergence rate values for the developed solution algorithms and large-size problem instances.

Instance	Number of Vessels	Number of Berths	EA		DPCEA		AEA		SAEA	
			$avg(CR)$	$std(CR)$	$avg(CR)$	$std(CR)$	$avg(CR)$	$std(CR)$	$avg(CR)$	$std(CR)$
31	55	4	0.904	0.050	0.863	0.086	0.959	0.030	0.837	0.159
32	55	6	0.853	0.150	0.869	0.117	0.959	0.039	0.873	0.115
33	55	8	0.941	0.043	0.838	0.130	0.943	0.032	0.897	0.119
34	60	4	0.948	0.046	0.922	0.082	0.939	0.048	0.921	0.079
35	60	6	0.912	0.073	0.908	0.072	0.933	0.073	0.911	0.077
36	60	8	0.804	0.162	0.784	0.215	0.955	0.052	0.948	0.041
37	65	4	0.923	0.076	0.841	0.180	0.952	0.065	0.822	0.149
38	65	6	0.845	0.103	0.906	0.077	0.941	0.057	0.906	0.110
39	65	8	0.659	0.269	0.848	0.125	0.966	0.034	0.946	0.033
40	70	4	0.914	0.082	0.967	0.028	0.950	0.037	0.885	0.157
41	70	6	0.867	0.113	0.922	0.051	0.944	0.019	0.922	0.068
42	70	8	0.668	0.277	0.889	0.102	0.959	0.044	0.931	0.076
43	75	4	0.885	0.135	0.907	0.066	0.961	0.032	0.877	0.135
44	75	6	0.871	0.104	0.934	0.091	0.956	0.041	0.873	0.134
45	75	8	0.828	0.172	0.897	0.075	0.955	0.019	0.915	0.085
46	80	4	0.886	0.102	0.914	0.058	0.954	0.052	0.948	0.046
47	80	6	0.812	0.178	0.933	0.057	0.953	0.042	0.912	0.073
48	80	8	0.771	0.206	0.885	0.071	0.965	0.032	0.957	0.043
49	85	4	0.969	0.037	0.921	0.072	0.953	0.036	0.898	0.103
50	85	6	0.878	0.109	0.921	0.059	0.956	0.039	0.939	0.093
51	85	8	0.799	0.331	0.839	0.132	0.947	0.047	0.950	0.038
52	90	4	0.945	0.031	0.890	0.102	0.937	0.047	0.854	0.127
53	90	6	0.912	0.042	0.852	0.095	0.956	0.044	0.916	0.059
54	90	8	0.775	0.231	0.809	0.195	0.969	0.027	0.974	0.021
55	95	4	0.951	0.031	0.959	0.053	0.950	0.049	0.950	0.048
56	95	6	0.878	0.101	0.931	0.072	0.965	0.043	0.932	0.090
57	95	8	0.824	0.167	0.803	0.140	0.969	0.023	0.949	0.074
58	100	4	0.933	0.063	0.917	0.068	0.970	0.030	0.916	0.087
59	100	6	0.907	0.093	0.926	0.061	0.946	0.034	0.952	0.049
60	100	8	0.744	0.127	0.880	0.118	0.968	0.025	0.958	0.035
Average:			**0.860**	**0.123**	**0.889**	**0.095**	**0.954**	**0.040**	**0.916**	**0.084**

5.4.6. Evolution of the self-adaptive algorithmic parameter values

As it is mentioned earlier in this dissertation, unlike the AEA, DPCEA, and EA algorithms, the crossover and mutation probabilities for the SAEA algorithm are encoded in the chromosomes (see Figure 14) and evolve throughout the search process. In order to track the evolution of the crossover and mutation probability values, they were recorded after the offspring selection for the whole population at each generation for each one of the generated large-size problem instances. The evolution of the average crossover and mutation probabilities (over all the chromosomes of the population) within the SAEA algorithm for the first replication of problem instances 49-60 (12 large-size problem instances with the largest amount of vessels, calling for service at the MCT) are illustrated in Figure 21 and Figure 22. Note that similar tendencies were observed in evolution of the average crossover and mutation probabilities for all the other developed problem instances.

According to Figure 21 and Figure 22, the pattern of changes in the crossover and mutation probability values within the SAEA algorithm is fairly complex as compared to other EA-based algorithms (e.g., DPCEA, where changes in the crossover and mutation probability values can be described using the piecewise function – see section 5.4.2 of the dissertation for more details). It can be observed that at the beginning of the search process, SAEA considers relatively high crossover and mutation probability values (i.e., the crossover probability is set to ≈0.60÷0.70, while the mutation probability is set to ≈0.035÷0.045). Besides, as it is getting close to the convergence, the crossover and mutation probability values are substantially reduced by SAEA (i.e., the crossover probability is reduced to ≈0.40÷0.50, while the mutation probability is set to ≈0.015÷0.025). The latter phenomenon can be explained by the fact that high crossover and mutation probability values at the beginning of the search process enable the SAEA algorithm to explore promising domains of the search space, while lower crossover and mutation probability values allow the SAEA algorithm focusing on exploitation of the promising domains identified towards convergence.

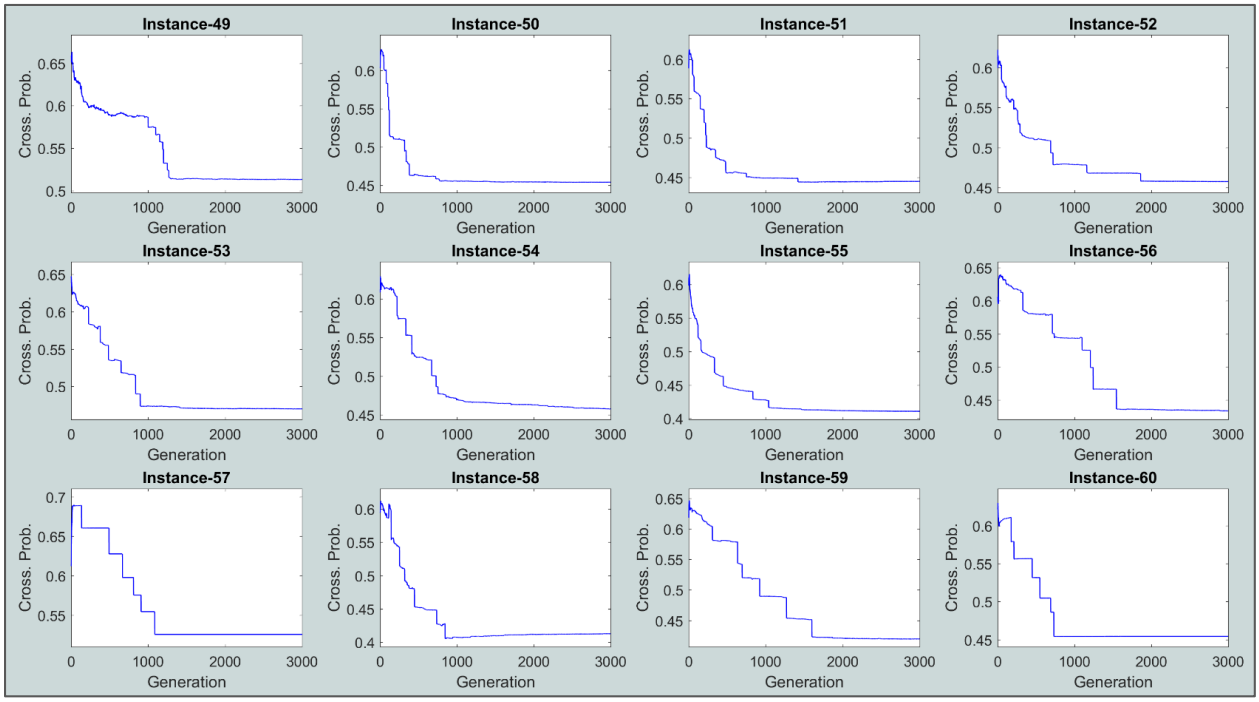

Figure 21 Evolution of the average crossover probability values for problem instances 49-60.

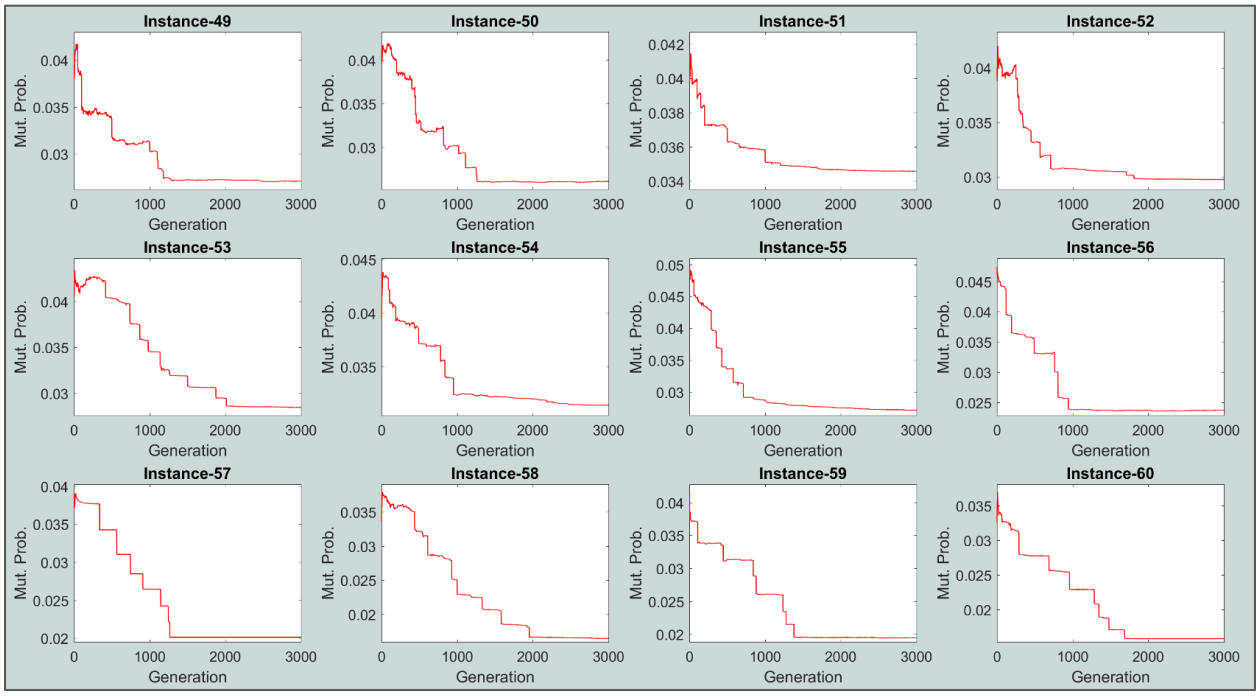

Figure 22 Evolution of the average mutation probability values for problem instances 49-60.

5.4.7. Algorithmic stability

Due to stochastic nature of the SAEA, AEA, DPCEA, and EA algorithms, the objective function value at termination, returned by the algorithms may vary from one algorithmic replication to another for a given problem instance. However, to consider an algorithm as a reliable decision support tool, the obtained objective function values by the algorithm should not be changed significantly from one replication to another (e.g., the berth schedules will significantly vary in terms of the total weighted vessel turnaround time and the total weighted vessel late departures). Hence, the stabilities of all the developed solution algorithms are evaluated in this chapter of the dissertation. Specifically, considering the variability in the objective function values at termination, the algorithmic stability was assessed over ten replications, which were performed for each large-size problem instance. The coefficient of variation was adopted as an indicator for the objective function variability. Figure 23 includes the values estimated for the objective function coefficient of variation over ten replications for each one of the developed solution algorithms and all the large-size problem instances.

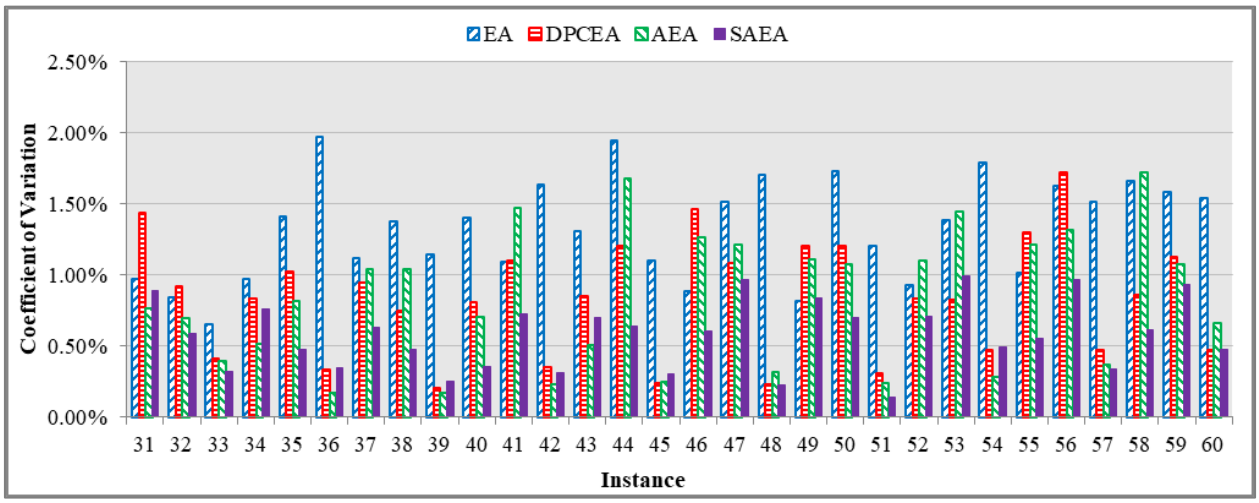

Figure 23 The objective function coefficient of variation values for the developed solution algorithms and large-size problem instances.

According to the presented results in Figure 23, the objective function coefficient of variation did not exceed 1.98% over all the large-size problem instances for the presented solution algorithms, which proves the stability of the developed algorithms. It can be observed that lower objective function coefficient of variation values were typically obtained by the SAEA algorithm.

Besides, higher objective function coefficient of variation values were generally recorded for the EA algorithm.

5.4.8. Changes in population diversity

One of the critical factors in the design of EAs is the population diversity. A diverse population should be a combination of high-quality individuals (a.k.a., "super-individuals"), and also individuals that have medium and low quality. A diverse population enables the EA algorithm to explore various domains of the search space more efficiently (Eiben and Smith, 2003; Sivanandam and Deepa, 2008). A population with a low diversity, especially at the beginning of the algorithmic run, may negatively affect the EA performance and lead to the "premature convergence". The lack of the explorative and exploitative capabilities may generally cause the "premature convergence" of the algorithm, as it cannot discover any other promising domains of the search space and converges at one of the local optima. The population diversity for all the developed solution algorithms was calculated in the conducted numerical experiments.

The standard deviation (STD) of the fitness function values of the population chromosomes is considered as an indicator of the population diversity for each algorithm. The population fitness STD values were estimated at each generation of the developed solution algorithms for each replication and large-size problem instance. The population fitness STD values throughout the evolution of the SAEA, AEA, DPCEA, and EA algorithms for the first replication of problem instances 55-60 are illustrated in Figure 24. Note that the rest of replications and problem instances showed similar tendencies for the algorithmic population diversity changes.

The conducted numerical experiments demonstrate that all the developed solution algorithms kept a diverse population, especially at the beginning of the search process (i.e., within the first ≈50 generations). Considering the obtained results, the population fitness STD values even exceeded 1000 hrs. for certain solution algorithms (e.g., the maximum population fitness STD values of the SAEA and EA algorithms comprised 1138.08 hrs. and 1096.09 hrs., respectively, for problem instance 56). The adopted population initialization mechanism (i.e., the half of the population was initialized using the FCFS-SR heuristic, and another half was generated randomly) can justify such a high population diversity at the beginning of the search process. Note that

although a random population initialization may negatively affect fitness of the generated chromosomes, it allows maintaining a diverse population.

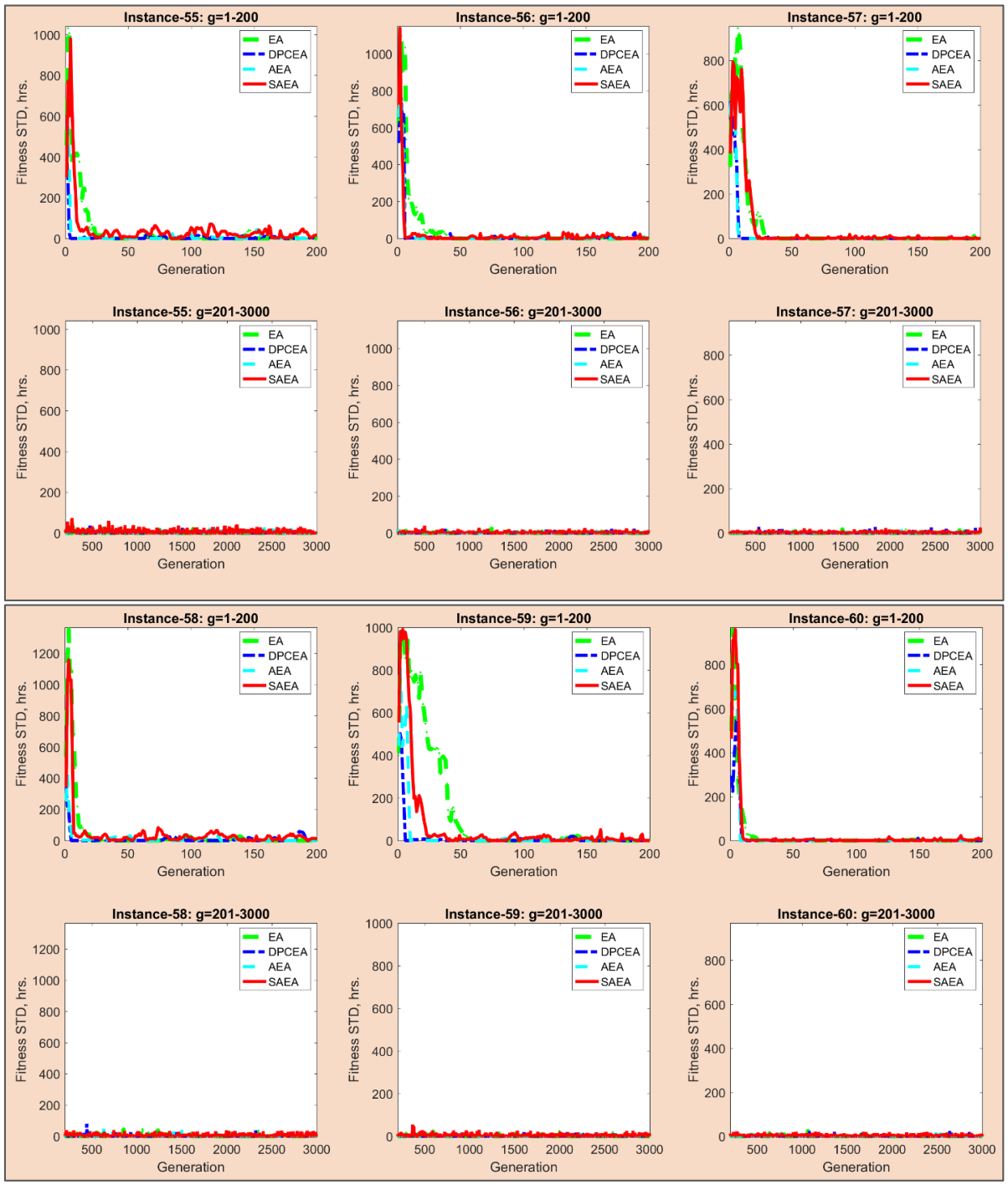

Figure 24 The population fitness STD values for the developed solution algorithms and problem instances 55-60.

Moreover, it can be observed that the population fitness STD significantly decreased after ≈50 generations. The latter can be justified by the fact that the developed solution algorithms identified a set of promising domains of the search space and focused on exploitation of these domains. The results demonstrated that all the developed solution algorithms were able to maintain a diverse population, however, application of the self-adaptive parameter control strategy (within the SAEA algorithm) was found to be more efficient for the search process in comparison with other parameter control strategies and the parameter tuning strategy, as the SAEA algorithm outperformed the other solution algorithms in terms of the objective function values at termination (see section 5.4.4 of the dissertation for more details).

5.5. Conclusions for Chapter 5

This chapter proposes a self-adaptive EA for the BSP, where the crossover and mutation probabilities (which are considered as the major parameters of EAs) are encoded in the chromosomes and evolve throughout the search process from one generation to another. Comprehensive numerical experiments were conducted to evaluate performance of the developed self-adaptive EA against the alternative EAs, which were based on the deterministic parameter control, adaptive parameter control, and parameter tuning strategies, respectively. The stability of all the considered solution algorithms in terms of the objective function values at termination and also the capability of the algorithms to keep the adequate population diversity were proven throughout the numerical experiments. The computational experiments conducted over the small-size problem instances demonstrated a superior performance of developed solution algorithms as compared to the exact optimization algorithm in terms of both objective function and computational time values due to the complexity of the proposed mathematical model.

As for the objective function values at termination, the developed self-adaptive EA outperformed the EAs with adaptive parameter control, deterministic parameter control, and parameter tuning strategies on average by 4.01%, 6.83%, and 11.84%, respectively, over the generated large-size problem instances. On the other hand, the computational time of the self-adaptive EA did not exceed 64.96 sec. over the large-size problem instances, which can be considered as acceptable from the practical point of view. A superior performance of the self-adaptive EA as compared to the other algorithms can be explained by the fact that the crossover

and mutation probabilities are encoded in the chromosomes and evolve throughout the algorithmic run, which further results in more efficient exploration and exploitation of the search space. Hence, the proposed solution algorithm can assist the marine container terminal operators with the design of cost-effective berth schedules.

CHAPTER 6

AN EVOLUTIONARY ALGORITHM WITH AUGMENTED SELF-ADAPTIVE PARAMETER CONTROL

The main focus of this chapter of the dissertation is to model the berth scheduling as a seaside operation at the MCT, where the containers have to be loaded/unloaded to/from the vessels that are managed by different liner shipping companies. As it is illustrated in Figure 25, a set of arriving vessels ($V = \{1, \dots, m\}$) will be served at the available berthing positions ($B = \{1, \dots, n\}$) at the MCT. A discrete berthing layout and also a dynamic vessel arrival pattern are adopted in this chapter of the dissertation. Note that uncertainty in the arrival time of vessels because of unpredictable issues (e.g., inclement weather conditions, vessel technical issues, human errors, etc.) is not explicitly considered in the developed mathematical model. Upon arrival of a given vessel at the MCT, a certain number of push boats will tow the vessel to the pre-assigned berthing position, and if the berthing position cannot serve the vessel at the moment, the vessel will be towed to the particular waiting area at the MCT.

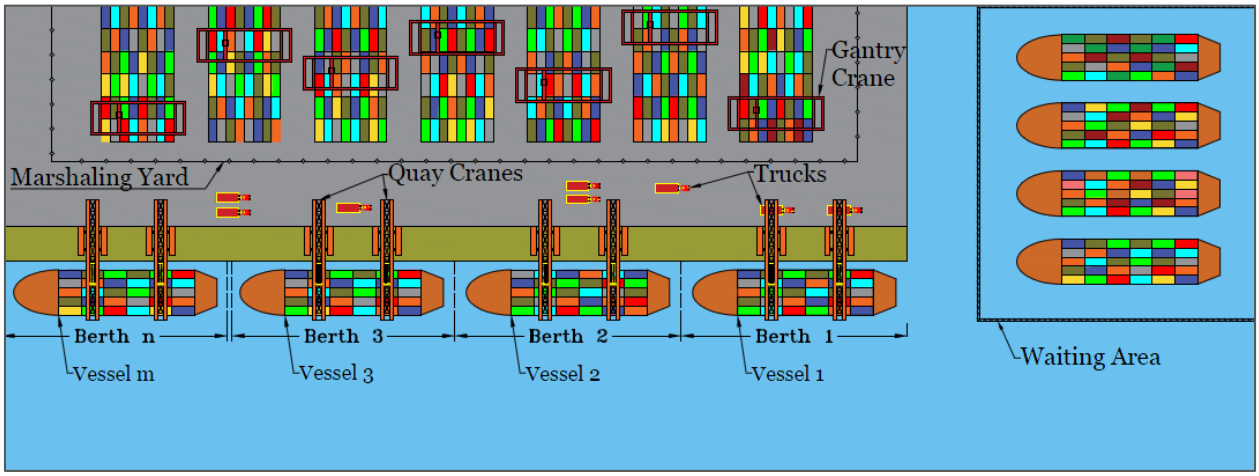

Figure 25 The MCT layout and basic MCT seaside operations.

In the process of assignment of vessels to the berthing positions, the spatial requirements are considered in this chapter of the dissertation. Specifically, the length and depth of the berthing position should be greater than the length and the draft of the vessel, considering a horizontal and

a vertical clearance, respectively. A given vessel can receive the highest handling rate at the preferred berthing position, which is the closest berthing position to the storage area determined at the marshaling yard for the vessel. The liner shipping companies and the MCT operator compromise on a departure time for each vessel, and the MCT operator will incur a penalty, if the vessel leaves the MCT later than the determined departure time.

In this chapter of the dissertation, an EA, which relies on the augmented self-adaptive parameter control strategy, is developed to solve the BSP. In this methodology, in addition to encoding the crossover and mutation probabilities in the solutions and evolution of them throughout the search process, the crossover and mutation probabilities will be updated based on the feedback from the search as well. Comprehensive numerical experiments will be conducted to evaluate performance of the designed algorithm against several commonly used algorithms in the BSP literature.

6.1. Model Formulation

A detailed description of the nomenclature adopted throughout this chapter of the dissertation is provided in this section. Moreover, the mixed-integer linear mathematical formulation developed for the discrete berth scheduling problem with dynamic vessel arrival pattern and spatial restrictions (DBSP) is presented in this section of the dissertation. The DBSP mathematical model components are described in Table 12.

The objective function (20) of the DBSP mathematical model (denoted as $\boldsymbol{TC}$) minimizes the total cost of serving the vessels arriving at the MCT. The total vessel service cost includes a total of three components: (1) total vessel waiting cost; (2) total vessel handling cost; and (3) total vessel late departure cost.

$$min\, \boldsymbol{TC} = [\sum_{v\in V}(\boldsymbol{\tau}_v^{wt} c_v^{wt}) + \sum_{v\in V}\sum_{b\in B}(t_{vb}^{ht}\boldsymbol{x}_{vb} c_v^{ht}) + \sum_{v\in V}(\boldsymbol{\tau}_v^{lt} c_v^{lt})] \quad (20)$$

Table 12 The main components of the proposed mathematical model.

Model Component		Description
Type	**Nomenclature**	
Sets	$V = \{1, \dots, m\}$	set of vessels arriving for service at the MCT (vessels)
	$B = \{1, \dots, n\}$	set of the MCT berthing positions available for vessel service (berthing positions)
Decision Variables	$\boldsymbol{x}_{vb}, v \in V, b \in B$	=1 if vessel v is assigned for service at berthing position b (=0 otherwise)
	$\boldsymbol{y}_{\underline{v}\overline{v}}, \underline{v}, \overline{v} \in V, \underline{v} \neq \overline{v}$	=1 if vessel $\overline{v}$ is assigned for service immediately after vessel $\underline{v}$ at the same berthing position (=0 otherwise)
	$\boldsymbol{f}_v, v \in V$	=1 if vessel v is assigned for service as the first vessel at a given berthing position (=0 otherwise)
	$\boldsymbol{l}_v, v \in V$	=1 if vessel v is assigned for service as the last vessel at a given berthing position (=0 otherwise)
Auxiliary Variables	$\boldsymbol{\tau}_v^{st}, v \in V$	service start time for vessel v (hours)
	$\boldsymbol{\tau}_v^{ft}, v \in V$	service finish time for vessel v (hours)
	$\boldsymbol{\tau}_v^{wt}, v \in V$	waiting time of vessel v (hours)
	$\boldsymbol{\tau}_v^{lt}, v \in V$	late departure time of vessel v (hours)
Parameters	m	number of vessels arriving for service at the MCT (vessels)
	n	number of the MCT berthing positions available for vessel service (berthing positions)
	$t_v^a, v \in V$	arrival time of vessel v at the MCT (hours)
	$t_b^{ber}, b \in B$	time when berthing position b is available at the first time in the considered planning horizon (hours)
	$q_v, v \in V$	number of containers to be handled for vessel v (TEUs)
	$p_{vb}, v \in V, b \in B$	handling productivity that can be provided by the MCT operator for vessel v at berthing position b (TEUs/hour)
	$t_{vb}^{ht}, v \in V, b$	handling time of vessel v at berthing position b (hours)
	$t_v^d, v \in V$	requested departure time for vessel v (hours)
	$h_v^{ves}, v \in V$	draft of vessel v (feet)
	$h_b^{ber}, b \in B$	depth of berthing position b (feet)
	$d_v^{ver}, v \in V$	minimum vertical clearance for vessel v (feet)
	$l_v^{ves}, v \in V$	length of vessel v (feet)
	$l_b^{ber}, b \in B$	length of berthing position b (feet)
	$d_v^{hor}, v \in V$	minimum horizontal clearance for vessel v (feet)
	$c_v^{wt}, v \in V$	waiting cost for vessel v (USD/hour)
	$c_v^{ht}, v \in V$	handling cost for vessel v (USD/hour)
	$c_v^{lt}, v \in V$	late departure cost for vessel v (USD/hour)
	K	large positive number

Constraint set (21) guarantees that each vessel, which arrives for service at the MCT, has to be assigned to one of the available MCT berthing positions.

$$\sum_{b \in B} \boldsymbol{x}_{vb} = 1 \; \forall v \in V \tag{21}$$

Constraint set (22) indicates that each vessel has to be moored at the assigned MCT berthing position without violation of the minimum vertical clearance requirement.

$$(h_v^{ves} + d_v^{ver})\boldsymbol{x}_{vb} \leq h_v^{ber} \; \forall v \in V, b \in B \tag{22}$$

Constraint set (23) ensures that each vessel has to be moored at the assigned MCT berthing position without violation of the minimum horizontal clearance requirement.

$$(l_v^{ves} + d_v^{hor})\boldsymbol{x}_{vb} \leq l_b^{ber} \; \forall v \in V, b \in B \tag{23}$$

Constraint set (24) guarantees that only one vessel can be assigned as the first vessel for service at each MCT berthing position.

$$\boldsymbol{f}_{\underline{v}} + \boldsymbol{f}_{\overline{v}} \leq 3 - \boldsymbol{x}_{\underline{v}b} - \boldsymbol{x}_{\overline{v}b} \; \forall \underline{v}, \overline{v} \in V, \underline{v} \neq \overline{v}, b \in B \tag{24}$$

Constraint set (25) indicates that only one vessel can be assigned as the last vessel for service at each MCT berthing position.

$$\boldsymbol{l}_{\underline{v}} + \boldsymbol{l}_{\overline{v}} \leq 3 - \boldsymbol{x}_{\underline{v}b} - \boldsymbol{x}_{\overline{v}b} \; \forall \underline{v}, \overline{v} \in V, \underline{v} \neq \overline{v}, b \in B \tag{25}$$

Constraint set (26) ensures that each vessel can be either assigned as the first vessel for service at a given MCT berthing position or be served after another vessel.

$$\boldsymbol{f}_{\overline{v}} + \sum_{\underline{v} \in V : \underline{v} \neq \overline{v}} \boldsymbol{y}_{\underline{v}\overline{v}} = 1 \; \forall \overline{v} \in V \tag{26}$$

Constraint set (27) guarantees that each vessel can be either assigned as the last vessel for service at a given MCT berthing position or be served before another vessel.

$$\boldsymbol{l}_{\underline{v}} + \sum_{\overline{v} \in V : \overline{v} \neq \underline{v}} \boldsymbol{y}_{\underline{v}\overline{v}} = 1 \; \forall \underline{v} \in V \tag{27}$$

Constraint set (28) indicates that service of each vessel at the assigned MCT berthing position can begin only when that berthing position becomes available for service.

$$\boldsymbol{\tau}_v^{st} \geq \sum_{b \in B} t_b^{ber} \boldsymbol{x}_{vb} \;\; \forall v \in V \tag{28}$$

Constraint set (29) ensures that service of each vessel at the assigned MCT berthing position can begin only after its arrival at the MCT.

$$\boldsymbol{\tau}_v^{st} \geq t_v^a \; \forall v \in V \tag{29}$$

Constraint set (30) calculates the handling time of the arriving vessels at the available MCT berthing positions.

$$t_{vb}^{ht} = \frac{q_v}{p_{vb}} \; \forall v \in V, b \in B \tag{30}$$

Constraint set (31) estimates the service start time for each vessel taking into account the service start and handling times of the vessels, which are assigned to be served before that vessel at the same MCT berthing position.

$$\boldsymbol{\tau}_{\overline{v}}^{st} \geq \boldsymbol{\tau}_{\underline{v}}^{st} + \sum_{b \in B} \left(t_{\underline{v}b}^{ht} \boldsymbol{x}_{\underline{v}b} \right) - K\left(1 - \boldsymbol{y}_{\underline{v}\overline{v}}\right) \forall \underline{v}, \overline{v} \in V, \underline{v} \neq \overline{v} \tag{31}$$

Constraint set (32) calculates the service finish time of the arriving vessels at the assigned MCT berthing positions.

$$\boldsymbol{\tau}_v^{ft} \geq \boldsymbol{\tau}_v^{st} + \sum_{b \in B} \left(t_{vb}^{ht} \boldsymbol{x}_{vb} \right) \; \forall v \in V, b \in B \tag{32}$$

Constraint set (33) computes the waiting time of the arriving vessels before their service start time at the assigned MCT berthing positions.

$$\boldsymbol{\tau}_v^{wt} \geq \boldsymbol{\tau}_{\overline{v}}^{st} - t_v^a \; \forall v \in V \tag{33}$$

Constraint set (34) estimates the late departure times of the arriving vessels assigned for service at the MCT berthing positions.

$$\boldsymbol{\tau}_v^{lt} \geq \boldsymbol{\tau}_v^{ft} - t_v^d \; \forall v \in V \tag{34}$$

Constraint sets (35)-(37) present the nature of decision variables, auxiliary variables, and parameters, which are adopted in the developed DBSP mathematical model.

$$\boldsymbol{x}_{vb}, \boldsymbol{y}_{\underline{v}\overline{v}}, \boldsymbol{f}_v, \boldsymbol{l}_v \in \{0,1\} \; \forall v \in V, \underline{v}, \overline{v} \in V, \underline{v} \neq \overline{v}, b \in B \tag{35}$$

$$m, n, q_v \in N \; \forall v \in V \tag{36}$$

$$\boldsymbol{\tau}_v^{st}, \boldsymbol{\tau}_v^{ft}, \boldsymbol{\tau}_v^{wt}, \boldsymbol{\tau}_v^{lt}, t_v^a, t_b^{ber}, p_{vb}, t_{vb}^{ht}, t_v^d, h_v^{ves}, h_b^{ber}, d_v^{ver}, l_v^{ves}, l_b^{ber}, d_v^{hor}, c_v^{wt}, c_v^{ht}, c_v^{lt}, K \; R^+ \; \forall v \in V, b \in B \tag{37}$$

6.2. Solution Algorithm Description

Performance of a given EA is remarkably affected by values assigned to the algorithmic parameters. In this chapter of the dissertation, an Augmented Self-Adaptive EA (ASAEA) is proposed to solve the issues of the existing strategies that are regularly used for setting the major algorithmic parameters in the process of evolution of the algorithm (crossover and mutation probabilities are considered as major algorithmic parameter in this chapter of the dissertation, as they enable the algorithm with explorative and exploitative capabilities – Eiben and Smith, 2003). Specifically, ASAEA is based on the augmented self-adaptive parameter control strategy, where in addition to encoding the crossover and mutation probabilities in the solutions and evolving throughout the search, they are periodically updated based on the feedback from the search. Each step of the ASAEA algorithm design is exhaustively described in the following sections.

6.2.1. The main algorithmic steps

Figure 26 illustrates the main algorithmic steps of the ASAEA algorithm. As it can be observed, ASAEA has a more complex algorithmic process as compared to a typical EA.

In the first step, the input data required for the DBSP mathematical model and the ASAEA algorithm are provided. In the second step, the first population is initiated, and a separate data structure (denoted as “AlgPa” in Figure 26) is generated for storing the most promising crossover and mutation probability values. Next, the possibly existing infeasible individuals in the initial population are repaired by ASAEA. After evaluating fitness of the initial population, the stopping criterion is checked, and in case of satisfaction, the algorithm returns the best solution, and the search process is terminated; otherwise, the algorithm enters the main loop. The next step is allocated to the execution of the parent selection operator. Then, the criterion, defined for updating the algorithmic parameters, is checked. The algorithmic parameters will be updated, if the criterion is met (more details are provided in section 6.2.6). After that, the algorithmic operators (crossover and mutation) are executed to generate the offspring chromosomes.

In the next step, the newly generated offspring are checked for any infeasibility to be repaired by the algorithm. After that, the offspring are evaluated, and then a group of the offspring chromosomes are selected via the offspring selection operator. Once the stopping criterion is met, the chromosome with the best fitness value in the population is returned, which corresponds to the berth schedule with the lowest possible total vessel service cost, including the total vessel waiting cost, the total vessel handling cost, and the total vessel late departure cost.

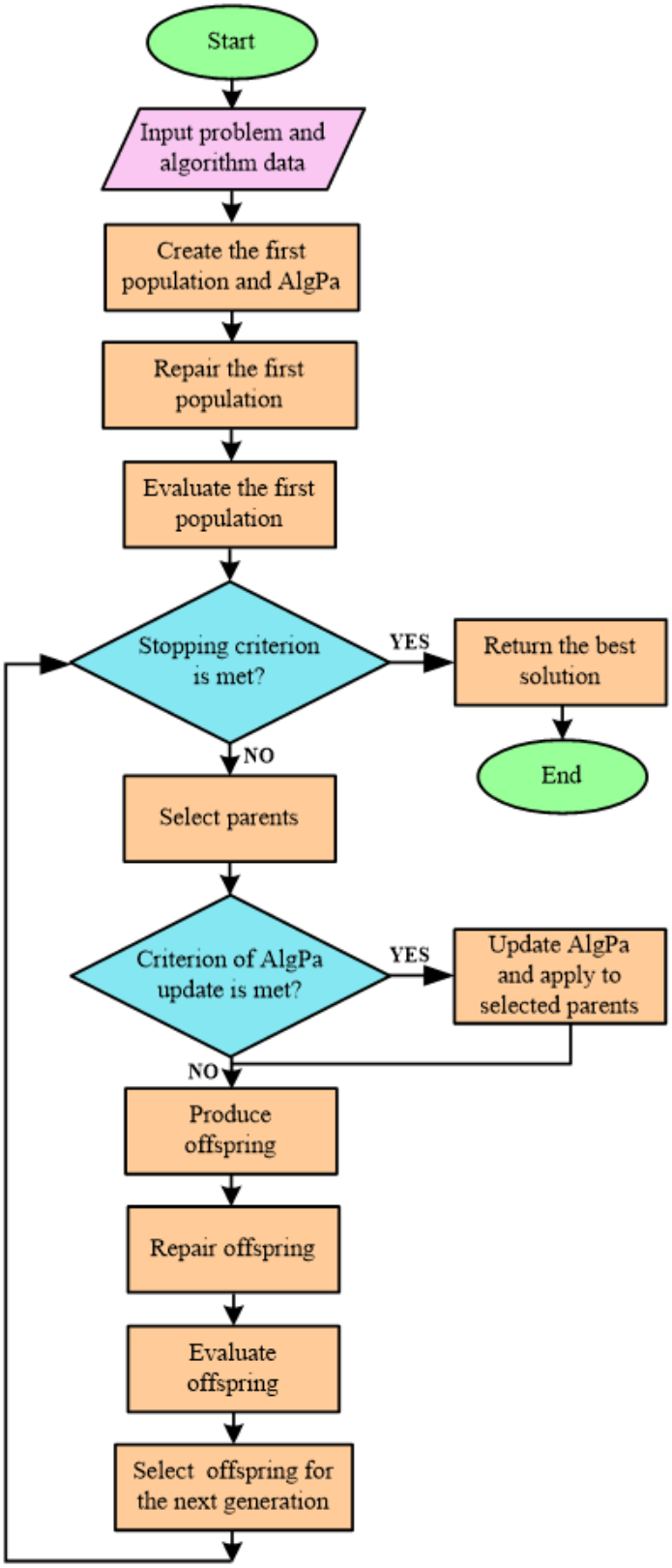

Figure 26 Main steps of ASAEA.

6.2.2. Solution encoding

As it is illustrated in Figure 27, a two-dimensional hybrid chromosome was adopted to present the solutions of the DBSP mathematical model. The proposed chromosome representation is classified as hybrid, as it comprises a combination of real values and integer values. The integer-coded portion is considered to represent the berthing position for each arriving vessel. On the other hand, the crossover and mutation probabilities for the ASAEA algorithm are stored in the real-coded portion of the chromosomes. A total of 7 vessels are assigned to 2 berthing positions in Figure 27. Vessels "5", "1", "3" and "6" have been assigned to berthing position "1"; while berthing position "2" is allocated to serve vessels "4", "7", and "2". Besides, the crossover probability (r^c) and the mutation probability (r^m) have been set equal to 0.35 and 0.02, respectively. Individual components of each ASAEA chromosome will be referred to as "genes" (Eiben and Smith, 2003). The term "locus" (plural: "loci") and "allele" will be utilized to denote the location and the value of a gene along each ASAEA chromosome, respectively (Eiben and Smith, 2003). Since the adopted chromosome representation is two-dimensional, two genes are accommodated in each locus of a given ASAEA chromosome. In the provided example, genes with alleles "3" (for a vessel identifier) and "1" (for a berthing position) are placed in locus "3".

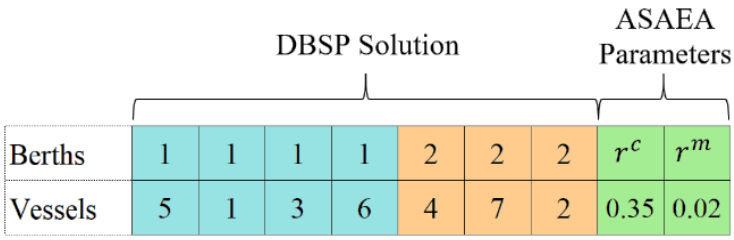

Figure 27 A chromosome representation model.

6.2.3. Initialization of the chromosomes and population

The First Come First Served with Spatial Requirements (FCFS-SR) heuristic was adopted in the developed ASAEA algorithm to generate the integer portion of the initial population chromosomes. Specifically, the FCFS-SR heuristic considers the geometric characteristics of arriving vessels and the MCT berthing positions. Application of the FCFS-SR heuristic prevents generation of the solutions that do not meet spatial constraints (i.e., the minimum vertical and horizontal clearance requirements should be satisfied for each vessel to be served at the MCT). Algorithm 6 outlines the main steps of FCFS-SR.

Algorithm 6: First Come First Served with Spatial Requirements (FCFS-SR) Heuristic

$\boldsymbol{FCFS\text{-}SR}(V, B, t^a, t^{ber}, t^{ht}, h^{ves}, h^{ber}, d^{ver}, l^{ves}, l^{ber}, d^{hor})$
in: $V = \{1, \dots, m\}$ – set of vessels; $B = \{1, \dots, n\}$ – set of berths; t^a – vessel arrival times; t^{ber} – initial berth availability; t^{ht} – vessel handling times; h^{ves} – draft of vessels; h^{ber}– depth of berths; d^{ver} – vertical clearance requirements; l^{ves} – length of vessels; l^{ber} – length of berths; d^{hor} – horizontal clearance requirements
out: $\boldsymbol{x}$ – initial vessel to berthing position assignment

0: $|\overline{\boldsymbol{t}^{ber}}| \leftarrow n$; $|\boldsymbol{x}| \leftarrow m \cdot n$; $|\boldsymbol{\tau}^{st}| \leftarrow m$; $|\boldsymbol{\tau}^{ft}| \leftarrow m$ ◁ Initialization
1: $\overline{\boldsymbol{t}^{ber}} \leftarrow t^{ber}$; $\bar{V} \leftarrow \boldsymbol{Sort}(V, t^a)$; $\bar{B} \leftarrow \boldsymbol{Sort}(B, \overline{\boldsymbol{t}^{ber}})$ ◁ Initialization
2: $v \leftarrow 1$
3: **for all** $v \in \bar{V}$ **do**
4: $cntr \leftarrow 1$
5: **while** $\left(h^{ber}_{\bar{B}_{cntr}} < h^{ves}_v + d^{ver}_v\right)$ **or** $\left(l^{ber}_{\bar{B}_{cntr}} < l^{ves}_v + d^{hor}_v\right)$ **do** ◁ Check the spatial requirements
6: $cntr \leftarrow cntr + 1$
7: **end while**
8: $b \leftarrow \bar{B}_{cntr}$
9: $\boldsymbol{x}_{vb} \leftarrow 1$ ◁ Assign a vessel for service at the appropriate berthing position
10: $\boldsymbol{\tau}^{st}_v \leftarrow max(\overline{\boldsymbol{t}^{ber}}, t^a_v)$ ◁ Calculate the vessel service start time
11: $\boldsymbol{\tau}^{ft}_v \leftarrow \boldsymbol{\tau}^{st}_v + t^{ht}_{vb}$ ◁ Calculate the vessel service finish time
12: $\overline{\boldsymbol{t}^{ber}_b} \leftarrow \boldsymbol{\tau}^{ft}_v$ ◁ Update availability of the selected berthing position
13: $\bar{B} \leftarrow \boldsymbol{Sort}(B, \overline{\boldsymbol{t}^{ber}})$ ◁ Sort berthing positions based on their availability
14: $v \leftarrow v + 1$
15: **end for**
16: **return** $\boldsymbol{x}$

The data structures, required for the FCFS-SR heuristic, are initiated in step "0" (note that $\overline{\boldsymbol{t}^{ber}}$ is the data structure, which is used in Algorithm 6 for storing the updated availability of the berthing positions). In step "1", the updated availability of the berthing positions is considered equal to the initial availability of the berthing positions. Moreover, vessels are sorted based on their arrival time at the MCT in step "1", while the berthing positions are sorted based on their availability. After that, Algorithm 6 enters the main loop (steps "3"–"15"). In steps "4"–"8", the heuristic determines the first available berthing position, which satisfies the spatial requirements considering the minimum vertical and horizontal clearance for a given vessel. In step "9", the first available berthing position with sufficient depth and length is allocated to serve the vessel. FCFS-SR calculates the vessel service start and finish times in steps "10" and "11", respectively. The availability of berthing positions is updated in step "12". Then, Algorithm 6 sorts the berthing positions based on their availability in step "13". The main loop keeps running until all the vessels have been assigned to the MCT berthing positions.

If the FCFS-SR heuristic runs several times, its deterministic nature leads to generation of exactly the same vessel to berthing position assignments in all the algorithmic runs. Hence, initializing the population by adopting only FCFS-SR is not desirable, as the ASAEA algorithm will start the search from just one point in the search space. In order to avoid the latter drawback, the first half of the initial ASAEA population will be initialized using FCFS-SR, while the second half will be generated randomly. Besides, the real-coded portion of the chromosomes will be generated randomly for the whole population, assuming the ranges [0.30, 0.80] and [0.01, 0.06] for the crossover and mutation probabilities, respectively. The latter ranges were adopted based on the available literature (Srinivas and Patnaik, 1994; Dulebenets et al., 2018a). The number of chromosomes in the ASAEA population (i.e., population size – $PopSize$) will be set based on the parameter tuning analysis (more details are provided in section 6.3.2).

6.2.4. Fitness function

All the chromosomes generated in the initial population are evaluated by the ASAEA algorithm (i.e., their fitness values will be estimated). This chapter of the dissertation assumes that the fitness function will be equal to the objective function (20) of the DBSP mathematical model (i.e., the fitness function will be equal to the total cost of serving the vessels arriving at the MCT).

6.2.5. Selection of the parent chromosomes

The parent selection is the first step of the main loop of the ASAEA algorithm. The ASAEA algorithm will further use the parent chromosomes to produce and mutate the offspring chromosomes via the crossover and mutation operations. The Boltzmann selection mechanism was adopted to select the parent chromosomes, which is based on the following equation (Eiben and Smith, 2003):

$$P(X_j) = \frac{e^{-u(X_j)/T_g}}{\sum_i e^{-u(X_i)/T_g}} \tag{38}$$

where: X_j is a solution; $u(X_j)$ is the fitness function value of solution X_j; T_g is the temperature in generation g; $P(X_j)$ is the selection probability assigned to a given solution.

The temperature value is updated from one generation to another within the proposed ASAEA algorithm as follows: $T_g = T_0 - \Delta T \cdot g$, where T_0 is the initial temperature; and ΔT is the temperature interval.

Algorithm 7: Boltzmann Selection

$\boldsymbol{BoltzSelection}(CurrentPop, PopFit, T_g, \omega)$
in: $CurrentPop$ – the population of the current generation; $PopFit$ – the fitness values of the population chromosomes; T_g – Boltzmann temperature in the current generation; ω – normalizing coefficient
out: $Parents$ – the selected parent chromosomes

0: $|BoltzPro| \leftarrow |CurrentPop|$; $|Parents| \leftarrow \emptyset$ ◁ Initialization
1: $i \leftarrow 1; j \leftarrow 1$
2: **while** $i \leq |CurrentPop|$ **do**
3: $BoltzPro_i \leftarrow \frac{e^{\frac{-PopFit_i}{\omega}/T_g}}{\sum_k e^{\frac{-PopFit_k}{\omega}/T_g}}$ ◁ Calculate the Boltzmann probability
4: $i \leftarrow i + 1$
5: **end while**
6: **while** $|Parents| \neq |CurrentPop|$ **do**
7: $a \leftarrow \boldsymbol{Rand}(0,1)$ ◁ Generate a random number
8: **while** $j \leq |CurrentPop|$ **do**
9: **if** $BoltzPro_j \geq a$ **then**
10: $Parents \leftarrow Parents \cup \{CurrentPop_j\}$ ◁ Append the selected parent chromosome
11: **break**
12: **end if**
13: $j \leftarrow j + 1$
14: **end while**
15: **end while**
16: **return** $Parents$

Algorithm 7 represents the main steps of the Boltzmann Selection mechanism. In step "0", two data structures named as $BoltzPro$ and $Parents$ are initialized to store the Boltzmann probability and the selected parent chromosomes, respectively. The Boltzmann probability is estimated for each one of the population chromosomes in steps "2"–"5". Unlike the canonical Boltzmann probability formula (see equation (38)), the proposed Boltzmann probability formula (outlined in step "3" of Algorithm 7) includes the normalizing coefficient (ω), in order to avoid abnormal temperatures due to fitness function fluctuations in the ASAEA population. In step "7" a random number "a" is generated during the execution of the "while" loop (steps "6"–"15"). Then, the Boltzmann selection mechanism searches for the chromosome, whose Boltzmann probability is greater than "a" (steps "8"–"14"). Once the latter requirement is met, the found

parent chromosome is appended to the generated $Parents$ data structure. Upon the selection of adequate number of parent chromosomes, the Boltzmann selection mechanism terminates the loop and returns the $Parents$ data structure in step "16".

6.2.6. ASAEA algorithmic parameters update

The most promising values of the crossover and mutation probabilities (i.e., the crossover and mutation probabilities that yield the largest relative improvements in terms of the fitness values) are stored in an external archive in each generation throughout the algorithmic search. The maximum of γ pairs of the crossover and mutation probabilities are stored in the archive. The effect of each set of the encoded crossover and mutation probabilities on each chromosome is tracked by estimating the relative fitness value improvement for a given chromosome as compared to the previous generation. In detail, the augmentation concept is implemented by executing two updating processes: (1) updating the crossover and mutation probabilities stored in the archive in each generation; and (2) updating the crossover and mutation probabilities encoded in the population every α generations.

In order to update the archive, a total of δ most promising crossover and mutation probability sets in the population replace δ least promising crossover and mutation probability sets in the archive in each generation. Then, every α generations, one set of algorithmic parameters out of γ sets, which were stored in the archive, is selected randomly. The selected set of parameters, which essentially belongs to one of the previous generations, is used to replace the algorithmic parameters encoded in β randomly selected individuals in the current population. Note that the values of α, β, γ and δ will be set during the process of parameter tuning (more details are provided in section 6.3.2 of the dissertation). The sets of the algorithmic parameters, which resulted in the best solutions, have a higher probability to be encoded within the chromosomes throughout the search process.

6.2.7. ASAEA operations

In order to produce and mutate the offspring chromosomes, the algorithmic operations are applied to the selected parents. According to Figure 27, each one of the chromosomes in the population includes an integer-coded portion and a real-coded portion. Thus, customized crossover

and mutation operators should be developed, which can be applied to the integer- and real-coded portions, separately. Moreover, considering the defined problem, a vessel should not be assigned to more than one berthing position in a feasible solution. The latter means that any repetitive number in the second row of the integer portion of the chromosomes makes the solution infeasible (see Figure 27). Hence, the algorithmic operators should be designed considering the latter aspect for the integer portion of the chromosomes. The developed crossover and mutation operators are comprehensively described in the two following sections.

6.2.7.1. Crossover operator

The crossover operator gives the capability of exploring different domains of the search space to the EA algorithms; hence, EAs can explore the search space aiming to find the best possible solutions (Eiben and Smith, 2003). In this chapter of the dissertation, two different crossover operators are combined together in order to apply the crossover operation to the hybrid chromosomes. The cycle crossover was encoded for the integer-coded portion, whereas the Laplace crossover was developed for the real-coded portion of the chromosomes. The integer-coded portions of chromosomes are mated utilizing the cycle crossover, which prevents generating infeasible solutions. On the other hand, the canonical crossover types (e.g., one-point crossover) might generate infeasible solutions. Furthermore, the Laplace crossover, introduced by Deep and Thakur (2007), was adopted in this chapter of the dissertation for the real-coded portion of chromosomes. The operation of cycle crossover is illustrated in Figure 28.

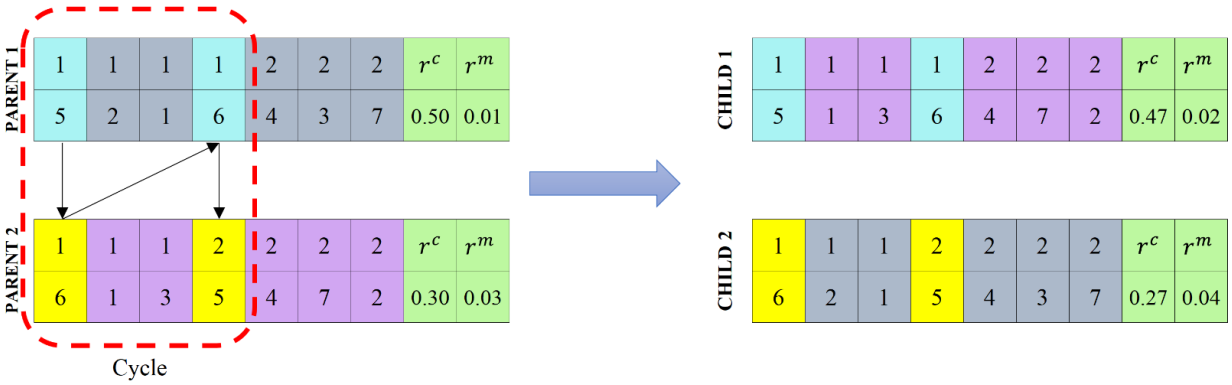

Figure 28 Cycle and Laplace crossover operations.

Based on the example provided in Figure 28, the following steps should be taken to establish a cycle: (1) select the first allele of the first parent (the vessel-related alleles are considered); (2) consider the same locus in the second parent; (3) look for the same allele (as the allele in the second parent) in the first parent and select that allele; and (4) do steps "2" and "3" until the first allele, identified in step "1", is found in the second parent. Then, the selected genes in the cycle from parent "1" are copied into child "1" (Eiben and Smith, 2003). In the provided example, the gene arrays, selected from parent "1" during the cycle (including vessels "5" and "6"), are copied into child "1". The missing vessels in child "1", which have not been included in the copied genes from parent "1", are copied from parent "2" into child "1". In the provided example, the missing vessels (including gene arrays "1", "3", "4", "7", and "2") in child "1" are copied from parent "2". The same procedure should be repeated to produce child "2".

The adopted Laplace crossover for the real-coded portion of the chromosomes operates according to the Laplace distribution. Firstly, the developed equation (39) is utilized to generate a random coefficient (denoted as μ):

$$\mu = \begin{cases} \phi - \psi \cdot [\ln(\theta)], \theta \leq 0.5 \\ \phi + \psi \cdot [\ln(\theta)], \theta > 0.5 \end{cases} \quad (39)$$

where: θ is a normally distributed random number in the range [0,1]; $\phi \in R^+$ and $\psi > R^+$ are the coefficients of the Laplace density function (in this chapter of the dissertation, it was assumed that $\phi = 0$, and $\psi = 0.75$). The smaller the value of ψ, the closer the genotype of children to the genotype of parents.

Suppose that the Laplace crossover is adopted to generate children using the following two parents:

$$\begin{cases} Y_1 = (y_{11}, y_{12}, \dots, y_{1n}) \\ Y_2 = (y_{21}, y_{22}, \dots, y_{2n}) \end{cases} \quad (40)$$

Then, the following equation can be utilized to calculate each one of the alleles of the children:

$$\begin{cases} \sigma_{1i} = y_{1i} + \mu|y_{1i} - y_{2i}| \\ \sigma_{2i} = y_{2i} + \mu|y_{1i} - y_{2i}| \end{cases} \quad (41)$$

where: y_{1i} and y_{2i} are the alleles of the parents; σ_{1i} and σ_{2i} are the alleles of the generated children.

Figure 28 illustrates an example of the Laplace crossover operation. Two parent chromosomes accommodate the crossover probabilities of 0.50 and 0.30 and the mutation probabilities of 0.01 and 0.03. Application of the Laplace crossover leads to the crossover and mutation probabilities equal to 0.47 and 0.02 for child "1", respectively. Besides, the crossover and mutation probabilities for child "2" were set by the Laplace crossover to 0.27 and 0.04, respectively.

6.2.7.2. Mutation operator

Domains of the search space are represented using the offspring chromosomes, generated by the designed crossover operator. The EA algorithms exploit the domains using the mutation operator to improve the quality of discovered solutions. In particular, a solution is changed slightly using a mutation operator to exploit a domain; otherwise, it might discover a new area in the search space instead of exploiting a specific domain. The integer-coded portions of the chromosomes undergo a customized mutation operator, which is a combination of the swap and insert mutation operators (Eiben and Smith, 2003). Figure 29 represents an example of a mutation operation. For swap mutation, alleles of two randomly selected genes are exchanged among each other. In the considered example, loci "2" and "7" exchange their alleles, which contain berthing positions "1" and "2", respectively; while vessel "5" of locus "1" is swapped with vessel "4" of locus "5" (see Figure 29). On the other hand, the insert mutation moves the second gene (i.e., the outmost right gene, selected for mutation) to the right locus of the first gene (i.e., the outmost left gene, selected for mutation), and shifts the rest of the genes to the right. In the considered example, berthing position "2" of locus "7" is inserted in locus "3", while vessel "4" of locus "5" is inserted in locus "2" (see Figure 29).

The floating point mutation was adopted for the real-coded portion of the chromosomes. The floating point mutation considers a range and generates a random value in the given range and

replaces the old allele with the newly generated random value (Figure 29). In the example provided in Figure 29, the crossover and mutation probability values were updated from 0.40 and 0.020 to 0.50 and 0.035, respectively, for the swap mutation example. On the other hand, the crossover and mutation probability values were updated from 0.40 and 0.020 to 0.48 and 0.018 for the insert mutation example, respectively (Figure 29). Algorithm 8 outlines how three aforementioned mutation operators are applied to the chromosomes within the developed ASAEA algorithm.

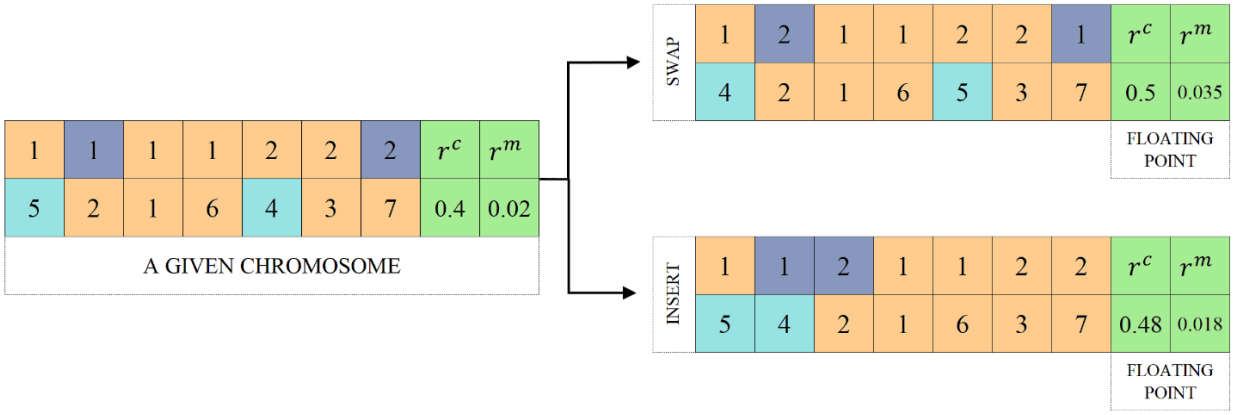

Figure 29 Swap, insert, and floating point mutation operations.

A new data structure ($MutOffspring$) is initialized for storing the mutated offspring chromosomes in step "0". Then, the main loop of the customized mutation operator starts (steps "2"–"17"). The random value ($RandNum1$) in the range [0, 1] is generated in step "3". Algorithm 8 checks the mutation probability in step "4". If the defined condition is met, the mutation operation will be executed for the crossover and mutation probability values. Considering the application of the self-adaptive parameter control strategy in ASAEA, r_i^m is the mutation probability encoded in each one of the ASAEA chromosomes. In steps "5" and "6", the floating point mutation is applied to the real-coded portion of the given chromosome (i.e., the crossover and mutation probability values). Another random number ($RandNum2$), chosen from the range [0, 1], is generated in step "8" to determine whether the swap or insert mutation should be implemented. If $RandNum2$ falls into the range [0.0, 0.25] or the range [0.51, 0.75], the swap mutation operator will be executed for the integer-coded portion of the chromosomes (steps "10" and "11"); otherwise, the insert mutation operator will be implemented for the integer-coded

portion of the chromosomes (steps "13" and "14"). The mutated offspring chromosomes (i.e., $MutOffspring$) is returned by Algorithm 8 in step "18".

Algorithm 8: Customized Mutation

$\boldsymbol{CustMutation}(Offspring)$
in: $Offspring$ – the offspring chromosomes generated by the crossover operator
out: $MutOffspring$ – the mutated offspring chromosomes

0: $MutOffspring \leftarrow Offspring$ ◁ Initialization
1: $i \leftarrow 1$
2: **while** $i \leq |Offspring|$ **do**
3: $RandNum1 \leftarrow \boldsymbol{Rand}(0,1)$ ◁ Generate a random number
4: **if** $r_i^m > RandNum1$ **then**
5: $MutOffspring_i^{r^c} \leftarrow \boldsymbol{Float}(Offspring_i^{r^c})$ ◁ Mutate the crossover probability value
6: $MutOffspring_i^{r^m} \leftarrow \boldsymbol{Float}(Offspring_i^{r^m})$ ◁ Mutate the mutation probability value
7: **end if**
8: $RandNum2 \leftarrow \boldsymbol{Rand}(0,1)$ ◁ Generate a random number
9: **if** $0 \leq RandNum2 \leq 0.25$ **or** $0.51 \leq RandNum2 \leq 0.75$ **then**
10: $MutOffspring_i^{ber} \leftarrow \boldsymbol{Swap}(Offspring_i^{ber}, r^m)$ ◁ Apply swap to berthing positions
11: $MutOffspring_i^{ves} \leftarrow \boldsymbol{Swap}(Offspring_i^{ves}, r^m)$ ◁ Apply swap to vessel identifiers
12: **else**
13: $MutOffspring_i^{ber} \leftarrow \boldsymbol{Insert}(Offspring_i^{ber}, r^m)$ ◁ Apply insert to berthing positions
14: $MutOffspring_i^{ves} \leftarrow \boldsymbol{Insert}(Offspring_i^{ves}, r^m)$ ◁ Apply insert to vessel identifiers
15: **end if**
16: $i \leftarrow i + 1$
17: **end while**
18: **return** $MutOffspring$

6.2.8. Chromosome repairing

The initial population generation and deployment of the algorithmic operators (i.e., crossover and mutation) may create infeasible chromosomes. A repairing operator was developed to be applied to infeasible chromosomes. Two factors can independently make a chromosome an infeasible one: (1) disruption of the vessel service order in a given chromosome; and (2) assignment of a given vessel to the berthing position, which does not meet the spatial requirements. To clarify factor (1), which causes the infeasibility of a chromosome, Figure 30 is presented. Vessels "6" and "4", which are assigned to berthing position "1", are placed in the loci between vessels "1", "3", and "7", which are assigned to berthing position "2". The latter causes disruption

in the vessel service order at berthing positions "1" and "2". The repairing operator, developed in the ASAEA algorithm, adjusts the service order of vessels (see Figure 30).

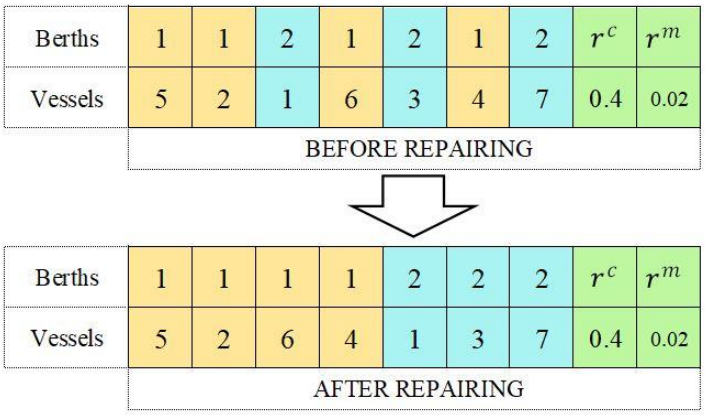

Figure 30 Repairing an infeasible chromosome.

Algorithm 9: Chromosome Repairing

$\boldsymbol{ChromRepair}(MutChrom, V, B, h^{ves}, h^{ber}, d^{ver}, l^{ves}, l^{ber}, d^{hor})$
in: $MutChrom$ – the mutated offspring chromosome in the current generation; $V = \{1, \dots, m\}$ – set of vessels; $B = \{1, \dots, n\}$ – set of berths; h^{ves} – draft of vessels; h^{ber}– depth of berths; d^{ver} – vertical clearance requirements; l^{ves} – length of vessels; l^{ber} – length of berths; d^{hor} – horizontal clearance requirements
out: $ChromRep$ – the repaired chromosome

0: $ChromRep \leftarrow MutChrom$ ◁ Initialization
1: $cntr \leftarrow 1; v \leftarrow 1; b \leftarrow 1$ ◁ Initialization
2: **while** $cntr \leq m$ **do**
3: $v \leftarrow ChromRep^{ves}_{cntr}; b \leftarrow ChromRep^{ber}_{cntr}$ ◁ Determine the vessel and berth identifiers
4: **if** $\left(h^{ber}_b < h^{ves}_v + d^{ver}_v\right)$ **or** $\left(l^{ber}_b < l^{ves}_v + d^{hor}_v\right)$ **do** ◁ Check the spatial requirements
5: $FitBerths \leftarrow \boldsymbol{Find}(h^{ber} \geq h^{ves}_v + d^{ver}_v$ **and** $l^{ber} \geq l^{ves}_v + d^{hor}_v)$ ◁ Find the appropriate berths
6: $ChromRep^{ber}_{cntr} \leftarrow \boldsymbol{Rand}(FitBerths)$ ◁ Select one of the appropriate berths
7: **end if**
8: $cntr \leftarrow cntr + 1$
9: **end while**
10: $ChromRep \leftarrow \boldsymbol{Sort}(ChromRep, B)$ ◁ Sort the gene arrays by berthing positions
11: **return** $ChromRep$

Algorithm 9 outlines the main steps of the repairing operator. A data structure (i.e., $ChromRep$) is initialized to store the repaired chromosome (step "0"). Then, Algorithm 9 checks the satisfaction of the spatial requirements for each vessel encoded in the given chromosome (steps

"2"–"9"). In case of violation of spatial requirements for a given vessel, all the berthing positions, which have adequate depth and length to serve the vessel, are temporarily stored in data structure denoted as $FitBerths$ (step "5"). In order to avoid any bias in the repairing process, one of the appropriate berthing positions, which satisfies the spatial requirements for a given vessel, is selected randomly (step "6"). After accomplishing the process of spatial requirement control for each vessel, Algorithm 9 sorts the gene arrays of the mutated offspring chromosome according to berthing positions to avoid any disruptions in the vessel service order (step "10"). The algorithm returns the repaired offspring chromosome in step "11".

6.2.9. Selection of the offspring chromosomes

The offspring chromosomes were selected by adoption of a Stochastic Universal Sampling (SUS) selection operator within the developed ASAEA algorithm. Although the process of chromosome selection by canonical selection mechanisms (e.g., ranking selection) is remarkably influenced by the fitness value of chromosomes, SUS is not substantially biased by fitness of the chromosomes. The SUS mechanism is schematically presented in Figure 31.

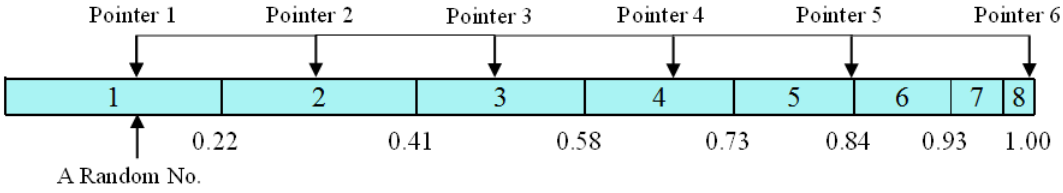

Figure 31 Stochastic universal sampling selection.

According to Figure 31, the fitness values of 8 given chromosomes are normalized in the range [0, 1], and suppose that $Nchrom = 6$ chromosomes should be selected. Then, the start point of the pointers is determined by generating a random number in the range [0, 1/6]. The distance between the pointers is equal to $1/Nchrom$. The locations of the pointers determine the chosen offspring chromosomes. As it is illustrated in Figure 31, the pointers are located on the portions that belong to individuals "1", "2", "3", "4", "5", and "8", which means that the aforementioned individuals will be selected for the next generation. This operation should be repeated until the adequate number of the offspring chromosomes has been selected (Eiben and Smith, 2003).

6.2.10. Elitism strategy

The algorithmic operators generally cause a lot of changes in the population in a given generation; hence, there is no guarantee that better solutions would be generated as compared to the previous generation. Elitism is a mechanism that is deployed to keep the best solution unchanged in each generation and transfers it to the next generation. Specifically, before executing the algorithmic operators (i.e., parent selection, crossover, mutation, and offspring selection), the best solution is recorded by the elitism operator; then, the best solution will be appended to the next generation offspring chromosomes after the offspring selection. The elitism strategy stores the best solution within the developed ASAEA algorithm before executing the Boltzmann selection mechanism adopted for the parent selection.

6.2.11. Convergence criterion

In order to have a balanced tradeoff between the final solution quality and the computational time, a convergence or stopping criterion is defined in Evolutionary Algorithms, as the evolution process will never stop in nature (Eiben and Smith, 2003). Considering the changes in the quality of solutions discovered from one generation to another, a certain number of generations ($gens$) is considered as the stopping criterion for the ASAEA algorithm, developed in this chapter of the dissertation.

6.3. Computational Experiments

Performance of the developed ASAEA algorithm, which relies on the augmented self-adaptive parameter control strategy, was evaluated by implementing a set of comprehensive computational experiments. The conducted computational experiments are exhaustively described in this section of the dissertation. The computational experiments are classified in the following five major steps: (1) input data generation; (2) algorithmic parameter setting; (3) performance evaluation against the benchmark objective function values; (4) comprehensive comparative analysis against the alternative EA-based and state-of-the-art metaheuristics; and (5) algorithmic parameter evolution. The input data required for the DBSP mathematical model were generated in the first step. Then, the algorithmic parameters that belong to the ASAEA algorithm and other alternative algorithms were set by implementing a parameter tuning analysis. The alternative solution algorithms, which have been considered in this chapter, can be divided into two groups.

The EA-based algorithms with different strategies for setting the crossover and mutation probabilities establish the first group of the alternative algorithms, including: (1) a typical EA with constant values for the crossover and mutation probabilities that remain the same throughout the search process; (2) an EA with a deterministic parameter control strategy (DPCEA), where the algorithm updates the crossover and mutation probabilities based on a defined counter (e.g., a specific number of generations) throughout the evolution of the algorithm; (3) an EA with an adaptive parameter control strategy (AEA), where the crossover and mutation probabilities are controlled and updated considering the feedback from the search throughout the evolution of the algorithm; and (4) an EA with a self-adaptive parameter control strategy (SAEA), where the crossover and mutation probabilities are encoded in the population chromosomes and evolve along with the algorithm itself. For more details about the EA, DPCEA, AEA, and SAEA algorithms refer to chapter 5 of this dissertation. Other than the EA-based algorithms, a lot of different metaheuristic algorithms have been introduced in the literature, which have been inspired by different phenomena (Cordeau et al., 2005; Hansen et al., 2008; Arabani et al., 2011; Ting et al., 2014). The second group of the alternative algorithms includes the advanced metaheuristic algorithms, which have been commonly used in the MCT and freight terminal operations literature, including the following: (1) Tabu Search (TS); (2) Simulated Annealing (SA); (3) Variable Neighborhood Search (VNS); (4) Ant Colony Optimization (ACO); and (5) Particle Swarm Optimization (PSO). The five aforementioned algorithms are comprehensively described in Cordeau et al. (2005), Emde and Boysen (2016), Hansen et al. (2008), Arabani et al. (2011), and Ting et al. (2014), respectively.

The third step of the computational experiments includes evaluation of the objective function values returned by the adopted solution algorithms against the benchmark objective function values for the small-size problem instances. The developed mathematical model (i.e., DBSP) will be encoded in General Algebraic Modeling System (GAMS, 2016) environment to determine the benchmark objective function values. GAMS solves the mixed-integer linear programming models utilizing CPLEX. Note that the large-size problem instances will not be solved by CPLEX, as CPLEX could not return the global optimum for the large-size problem instances within 2 hours based on preliminary computational experiments. In the fourth step, the

alternative solution algorithms will be exhaustively evaluated adopting the large-size problem instances in terms of the following performance indicators: (a) objective function and computational time values; (b) convergence patterns; and (c) objective function stochastic variations. In the fifth step, the SAEA and ASAEA algorithms will be compared against each other considering evolution of the algorithmic parameters.

In this chapter of the dissertation, all the solution algorithms were encoded in MATLAB 2016a (Mathworks, 2019). An Alienware CPU with an Intel® Core™ i7-7700K processor and 32.0 GB of RAM was used to conduct the computational experiments. All the steps of the computational experiments are provided in sections 6.3.1-6.3.5 of this dissertation.

6.3.1. Input data generation

The input data for the DBSP mathematical model is generated in the first step of the computational experiments. The available BSP literature and appropriate online resources were the references, considered for adopting the input data that were required to conduct the computational experiments (Imai et al., 2008; Golias and Haralambides, 2011; Imai et al., 2014; Dulebenets et al., 2018a,b; Eniram, 2019). Table 13 represents how the values of the DBSP mathematical model parameters were generated. Considering the common assumptions used in the BSP literature, it was assumed that vessels arrive at the MCT according to an exponential distribution (Imai et al., 2008; Dulebenets et al., 2018a,b). An average inter-arrival time of vessels was set equal to 2 hours as follows: $t_{v+1}^{a} = t_{v}^{a} + \Delta t \; \forall v \in V, \Delta t = exp[2]$ (hours), where term "$exp[num_1]$" denotes the exponential distribution with a mean of num_1.

The beginning of the planning horizon was assumed as the first availability of each MCT berthing position (i.e., $t_{b}^{ber} = 0 \; \forall b \in B$). The number of containers that should be handled for each vessel was generated randomly between 1000 TEUs and 3000 TEUs based on the uniform distribution as follows: $q_v = U[1000; 3000] \; \forall v \in V$ (TEUs), where term "$U[num_1; num_2]$" denotes the uniform distribution with lower bound num_1 and upper bound num_2. Each vessel at a given berthing position was assigned the handling productivity as follows: $p_{vb} = U[130; 160] \; \forall v \in V, b \in B$ (TEUs/hour). Different factors were considered to estimate the departure time requested for a given vessel: (1) the arrival time of that vessel at the MCT; (2)

vessel handling time at the berthing position, which has the highest handling productivity; and (3) a tolerance factor, varying between 20% to 50% and capturing a potential increase in the vessel handling time (e.g., due to diversion of the vessel from the berthing position with the highest handling productivity to the alternative berthing position) as follows: $t_v^d = t_v^a + min_b(t_{vb}^{ht}) \cdot U[1.20; 1.50]\ \forall v \in V$ (hours).

Table 13 Data generation for the DBSP mathematical model.

Parameter	**Selected Value**
Number of arriving vessels: m (vessels)	The values differ based on the problem instance
Number of available berthing positions: n (berthing positions)	The values differ based on the problem instance
Arrival time of vessel v at the MCT: $t_v^a, v \in V$ (hours)	$t_{v+1}^a = t_v^a + \Delta t\ \forall v \in V$
Average inter-arrival time of vessels: Δt (hours)	$\Delta t = exp[2]$
First time availability of berthing position b in the planning horizon: $t_b^{ber}, b \in B$ (hours)	$t_b^{ber} = 0\ \forall b \in B$
Number of containers to be handled for vessel v: $q_v, v \in V$ (TEUs)	$q_v = U[1000; 3000]\ \forall v \in V$
Handling productivity for vessel v at berthing position b: $p_{vb}, v \in V, b \in B$ (TEUs/hour)	$p_{vb} = U[130; 160]\ \forall v \in V, b \in B$
Requested departure time for vessel v: $t_v^d, v \in V$ (hours)	$t_v^d = t_v^a + min_b(t_{vb}^{ht}) \cdot U[1.20; 1.50]\ \forall v \in V$
Draft of vessel v: $h_v^{ves}, v \in V$ (feet)	$[41.0; 41.0; 42.7; 47.6; 49.9; 50.9]$
Depth of berthing position b: $h_b^{ber}, b \in B$ (feet)	$h_b^{ber} = 1.20 \cdot min_v(h_v^{ves}) + U[0; 1.20 \cdot max_v(h_v^{ves}) - min_v(h_v^{ves})]\ \forall b \in B$
Minimum vertical clearance for vessel v (feet): $d_v^{ver}, v \in V$ (feet)	$d_v^{ver} = U[4; 8]\ \forall v \in V$
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)	$l_b^{ber} = 1.20 \cdot min_v(l_v^{ves}) + U[0; 1.20 \cdot max_v(l_v^{ves}) - min_v(l_v^{ves})]\ \forall b \in B$
Minimum horizontal clearance for vessel v: $d_v^{hor}, v \in V$ (feet)	$d_v^{hor} = U[50; 100]\ \forall v \in V$
Waiting cost for vessel v: $c_v^{wt}, v \in V$ (USD/hour)	$c_v^{wt} = U[1000; 5000]\ \forall v \in V$
Handling cost for vessel v: $c_v^{ht}, v \in V$ (USD/hour)	$c_v^{ht} = U[50000; 70000]\ \forall v \in V$
Late departure cost for vessel v: $c_v^{lt}, v \in V$ USD/hour)	$c_v^{lt} = U[5000; 10000]\ \forall v \in V$
Large positive number: K	10000

According to Eniram (2019), a group of well-known vessel types in the maritime transportation industry were modeled in this chapter of the dissertation: (1) Panamax (length: 820.2 feet, draft: 41.0 feet); (2) Panamax Max (length: 951.4 feet, draft: 41.0 feet); (3) Post Panamax

(length: 935.0 feet, draft: 42.7 feet); (4) Post Panamax Plus (length: 984.3 feet, draft: 47.6 feet); (5) New Panamax (length: 1200.8 feet, draft: 49.9 feet); and (6) Triple E (length: 1312.3 feet, draft: 50.9 feet). The dimensional specifications of aforementioned vessel types were considered to generate the dimensional specifications of the MCT berthing positions. Specifically, the depth of a given berthing position was randomly generated using the equation: $h_b^{ber} = 1.20 \cdot min_v(h_v^{ves}) + U[0; 1.20 \cdot max_v(h_v^{ves}) - min_v(h_v^{ves})]\ \forall b \in B$ (feet). Besides, the length of a given berthing position was set as follows: $l_b^{ber} = 1.20 \cdot min_v(l_v^{ves}) + U[0; 1.20 \cdot max_v(l_v^{ves}) - min_v(l_v^{ves})]\ \forall b \in B$ (feet). The minimum vertical and horizontal clearance values for each vessel were estimated using the following equations, respectively: $d_v^{ver} = U[4; 8]\ \forall v \in V$ (feet), $d_v^{hor} = U[50; 100]\ \forall v \in V$ (feet). The waiting cost for each vessel was randomly estimated between 1000 USD/hour and 5000 USD/hour considering the following uniform distribution as follows: $c_v^{wt} = U[1000; 5000]\ \forall v \in V$ (USD/hour). On the other hand, the handling cost for each vessel was assumed to vary between 50000 USD/hour and 70000 USD/hour and was estimated utilizing this equation: $c_v^{ht} = U[50000; 70000]\ \forall v \in V$ (USD/hour). Moreover, the late departure cost for each vessel was randomly generated between 5000 USD/hour and 10000 USD/hour: $c_v^{lt} = U[5000; 10000]\ \forall v \in V$ (USD/hour). The value of a large positive number was set to $K = 10000$.

Table 14 The developed problem instances.

Instance	Category	No. of Vessels	No. of Berths	Instance	Category	No. of Vessels	No. of Berths	Instance	Category	No. of Vessels	No. of Berths
PI-1	Small-Size	8	2	PI-21	Small-Size	14	4	PI-41	Large-Size	90	8
PI-2	Small-Size	8	3	PI-22	Small-Size	15	2	PI-42	Large-Size	90	10
PI-3	Small-Size	8	4	PI-23	Small-Size	15	3	PI-43	Large-Size	95	6
PI-4	Small-Size	9	2	PI-24	Small-Size	15	4	PI-44	Large-Size	95	8
PI-5	Small-Size	9	3	PI-25	Small-Size	16	2	PI-45	Large-Size	95	10
PI-6	Small-Size	9	4	PI-26	Small-Size	16	3	PI-46	Large-Size	100	6
PI-7	Small-Size	10	2	PI-27	Small-Size	16	4	PI-47	Large-Size	100	8
PI-8	Small-Size	10	3	PI-28	Small-Size	17	2	PI-48	Large-Size	100	10
PI-9	Small-Size	10	4	PI-29	Small-Size	17	3	PI-49	Large-Size	105	6
PI-10	Small-Size	11	2	PI-30	Small-Size	17	4	PI-50	Large-Size	105	8
PI-11	Small-Size	11	3	PI-31	Large-Size	75	6	PI-51	Large-Size	105	10
PI-12	Small-Size	11	4	PI-32	Large-Size	75	8	PI-52	Large-Size	110	6
PI-13	Small-Size	12	2	PI-33	Large-Size	75	10	PI-53	Large-Size	110	8
PI-14	Small-Size	12	3	PI-34	Large-Size	80	6	PI-54	Large-Size	110	10
PI-15	Small-Size	12	4	PI-35	Large-Size	80	8	PI-55	Large-Size	115	6
PI-16	Small-Size	13	2	PI-36	Large-Size	80	10	PI-56	Large-Size	115	8
PI-17	Small-Size	13	3	PI-37	Large-Size	85	6	PI-57	Large-Size	115	10
PI-18	Small-Size	13	4	PI-38	Large-Size	85	8	PI-58	Large-Size	120	6
PI-19	Small-Size	14	2	PI-39	Large-Size	85	10	PI-59	Large-Size	120	8
PI-20	Small-Size	14	3	PI-40	Large-Size	90	6	PI-60	Large-Size	120	10

The considered problem instances defined the number of arriving vessels (m) and the number of available berthing positions (n) independently. Considering the generated input data and by changing the number of arriving vessels and the number of available berthing positions, a total of 60 problem instances were developed. The developed problem instances were classified as the small-size and large-size problem instances, which are listed in Table 14.

6.3.2. Algorithmic parameter setting

The developed ASAEA algorithm will be evaluated against nine alternative state-of-the-art metaheuristic algorithms, which have been commonly adopted for berth scheduling in the MCT operations literature. There are a set of parameters in each one of the considered algorithms, which remain constant throughout the algorithmic run (while some of the considered algorithms have certain parameters that change throughout the algorithmic run – i.e., the DPCEA, AEA, SAEA, and ASAEA algorithms). The appropriate values for the aforementioned constant algorithmic parameters were set in the second step of the computational experiments.

The Taguchi's scheme was adopted to conduct the parameter selection analysis in a timely manner for all the developed algorithms in this chapter of the dissertation. Based on the Taguchi's scheme, only "the most favorable" parameter values are considered throughout the parameter tuning analysis (Arabani et al., 2011). Specifically, in order to tune one of the algorithmic parameters, a series of computational experiments is conducted, where the value of the selected parameter is changed. On the other hand, the other parameters are assigned certain constant values. A tradeoff between the solution quality at convergence and the required computational time is considered to determine the most promising values of a given algorithmic parameter. The same procedure is applied for the rest of the algorithmic parameters once the most promising values are determined for a given parameter.

In order to conduct the parameter selection analysis, a total of 3 problem instances were selected at random from the generated large-size problem instances (which are described in section 6.3.1 of the dissertation). Each algorithm was executed 10 times for each most favorable algorithmic parameter combination, aiming to obtain the average objective function and computational time values. Table 15 and Table 16 show the results from the performed parameter

selection analysis, including the algorithm, the parameter description, the candidate values considered for a given parameter, and the best value identified for a given parameter.

Table 15 The parameter selection analysis results for the considered EA-based metaheuristic algorithms.

Algorithm	Parameter Description	Candidate Values	Best Value	Algorithm	Parameter Description	Candidate Values	Best Value
EA	Population size ($PopSize$)	[40; 50; 60]	50	AEA	Crossover probability values (r^{cv})[2]	[0.80÷0.40; 0.80÷0.30; 0.30÷0.80]	[0.80÷0.30]
EA	Initial temperature (T_0)	[1200; 1500; 1800]	1500	AEA	Mutation probability values (r^{mv})[3]	[0.06÷0.01; 0.06÷0.02; 0.01÷0.06]	[0.06÷0.02]
EA	Temperature interval (ΔT)	[0.10; 0.15; 0.20]	0.20	AEA	Adaptive criterion ($gens^{max}$)	[100; 200; 300]	300
EA	Normalizing coefficient (ω)[1]	[0.10; 0.15; 0.20]	0.10	AEA	Maximum number of generations ($gens$)	[4000; 5000; 6000]	6000
EA	Crossover probability (r^c)	[0.30; 0.50; 0.80]	0.80	SAEA	Population size ($PopSize$)	[40; 50; 60]	50
EA	Mutation probability (r^m)	[0.01; 0.03; 0.06]	0.03	SAEA	Initial temperature (T_0)	[1200; 1500; 1800]	1200
EA	Maximum number of generations ($gens$)	[4000; 5000; 6000]	6000	SAEA	Temperature interval (ΔT)	[0.10; 0.15; 0.20]	0.20
DPCEA	Population size ($PopSize$)	[40; 50; 60]	50	SAEA	Normalizing coefficient (ω)[1]	[0.10; 0.15; 0.20]	0.10
DPCEA	Initial temperature (T_0)	[1200; 1500; 1800]	1500	SAEA	Maximum number of generations ($gens$)	[4000; 5000; 6000]	6000
DPCEA	Temperature interval (ΔT)	[0.10; 0.15; 0.20]	0.20	ASAEA	Population size ($PopSize$)	[40; 50; 60]	50
DPCEA	Normalizing coefficient (ω)[1]	[0.10; 0.15; 0.20]	0.10	ASAEA	Initial temperature (T_0)	[1200; 1500; 1800]	1200
DPCEA	Crossover probability values ($\widetilde{r^{cv}}$)	[0.80, 0.40; 0.80, 0.30; 0.30, 0.80]	[0.80, 0.30]	ASAEA	Temperature interval (ΔT)	[0.10; 0.15; 0.20]	0.20
DPCEA	Mutation probability values ($\widetilde{r^{mv}}$)	[0.06, 0.01; 0.06, 0.02; 0.01, 0.06]	[0.06, 0.02]	ASAEA	Normalizing coefficient (ω)[1]	[0.10; 0.15; 0.20]	0.10
DPCEA	Number of segments in the piecewise function (n^{seg})	[15; 20; 25]	20	ASAEA	Generational interval between the parameter updates (α)	[1; 10; 20]	1
DPCEA	Maximum number of generations ($gens$)	[4000; 5000; 6000]	6000	ASAEA	The external archive size (γ)[4]	[0.20; 0.25; 0.30]	0.30
AEA	Population size ($PopSize$)	[40; 50; 60]	50	ASAEA	Number of individuals for the parameter update (β)[4]	[0.20; 0.25; 0.30]	0.20
AEA	Initial temperature (T_0)	[1200; 1500; 1800]	1200	ASAEA	Percentage of parameter values to be updated in the external archive (δ)	[0.80; 0.90; 1.00]	1.00
AEA	Temperature interval (ΔT)	[0.10; 0.15; 0.20]	0.20	ASAEA	Maximum number of generations ($gens$)	[4000; 5000; 6000]	6000
AEA	Normalizing coefficient (ω)[1]	[0.10; 0.15; 0.20]	0.10				

Notes: 1 – the normalizing coefficient for the Boltzmann selection mechanism is set as a percentage of the average population fitness; 2 – the crossover probability values were altered with an increment of 0.10 for the AEA algorithm; 3 – the mutation probability values were altered with an increment of 0.01 for the AEA algorithm; 4 – the external archive size and the number of individuals for the parameter update were set as a percentage of the ASAEA population.

Table 16 The parameter selection analysis results for the considered state-of-the-art metaheuristic algorithms.

Algorithm	Parameter Description	Candidate Values	Best Value	Algorithm	Parameter Description	Candidate Values	Best Value
TS	Number of solutions evaluated during the local search (Ψ^{TS})	[5; 7; 10]	10	ACO	Population size ($PopSize$)	[40; 50; 60]	50
TS	Exchange rate (r^{TS})[1]	[2; 4; 6]	2	ACO	Evaporation rate (ρ)	[0.20; 0.30; 0.40]	0.20
TS	Tabu List size (Γ)	[10; 15; 20]	10	ACO	Initial pheromone amount ($\tau^0_{\underline{v}\overline{v}}, \underline{v}, \overline{v} \in V$)	[0.30; 0.40; 0.50]	0.30
TS	Maximum number of iterations ($iters$)	[4000; 5000; 6000]	6000	ACO	Pheromone trail parameter (α^{ACO})	[1.00; 1.50; 2.00]	1.00
SA	Initial temperature (T_0)	[1200; 1500; 1800]	1200	ACO	Heuristic parameter (β^{ACO})	[1.00; 1.50; 2.00]	1.50
SA	Temperature interval (ΔT)	[0.10; 0.15; 0.20]	0.20	ACO	Maximum number of iterations ($iters$)	[4000; 5000; 6000]	6000
SA	Normalizing coefficient (ω)[2]	[0.10; 0.15; 0.20]	0.10	PSO	Population size ($PopSize$)	[40; 50; 60]	50
SA	Exchange rate (r^{SA})[1]	[2; 4; 6]	2	PSO	Cognition component (c_1)	[1.50; 2.00; 3.00]	2.00
SA	Maximum number of iterations ($iters$)	[4000; 5000; 6000]	6000	PSO	Social component (c_2)	[1.50; 2.00; 3.00]	2.00
VNS	Number of neighborhood structures (Ψ^{VNS})	[5; 7; 10]	10	PSO	Maximum velocity increase (V^{PSO}_{max})[3]	[0.10; 0.20; 0.30]	0.20
VNS	Exchange rate (r^{VNS})[1]	[2; 4; 6]	2	PSO	Maximum number of iterations ($iters$)	[4000; 5000; 6000]	6000
VNS	Maximum number of iterations ($iters$)	[4000; 5000; 6000]	6000				

Notes: 1 – the exchange rate parameter corresponds to the number of vessel to berth assignments to be changed in a given solution in order to create a new solution during the local search; 2 – the normalizing coefficient for the Boltzmann selection mechanism is set as a percentage of the average population fitness; 3 – the inertia weight for PSO was set as follows: $W^{PSO} = 0.50 + U[0.00; 0.50]$ (Ting et al., 2014).

6.3.3. Performance evaluation against the benchmark objective function values

Performance of the developed algorithms against the benchmark objective function values is evaluated in the third step of the computational experiments. The MCT and operations research literature commonly adopt CPLEX as an exact optimization algorithm to obtain the global optimum for mixed-integer linear programming mathematical models (Hansen et al., 2008). The benchmark objective function values were estimated utilizing CPLEX in this chapter of the dissertation (i.e., the optimal objective function values), and the objective function values, returned by the developed algorithms (i.e., the EA, DPCEA, AEA, SAEA, ASAEA, TS, SA, VNS, ACO, and PSO algorithms) were compared against the benchmark values using the small-size problem instances [PI-1 through PI-30]. The computational time and the relative optimality gap of CPLEX were limited to 7,200 seconds and 0.01%, respectively. All the algorithms, adopted in this chapter of the dissertation, are stochastic in nature; therefore, a total of 10 replications were performed for each algorithm in order to obtain the average objective function and computational time values for the small-size problem instances.

Table 17 Comparison of the considered algorithms against the exact optimization (CPLEX) for problem instances [PI-1 through PI-30].

Instance	No. of Vessels	No. of Berths	CPLEX		EA		DPCEA		AEA		SAEA		ASAEA	
			Objective, 10^6 USD	CPU, sec.	Objective, 10^6 USD	CPU, sec.	Objective, 10^6 USD	CPU, sec.	Objective, 10^6 USD	CPU, sec.	Objective, 10^6 USD	CPU, sec.	Objective, 10^6 USD	CPU, sec.
PI-1	8	2	7.9382	0.31	7.9382	6.63	7.9382	6.86	7.9382	7.90	7.9382	9.58	7.9382	10.59
PI-2	8	3	7.1958	0.81	7.1958	6.65	7.1958	6.85	7.1958	7.93	7.1958	9.61	7.1958	10.64
PI-3	8	4	6.8256	0.22	6.8256	6.62	6.8256	6.81	6.8256	7.88	6.8256	9.55	6.8256	10.57
PI-4	9	2	9.0248	0.59	9.0248	6.79	9.0248	6.96	9.0248	8.04	9.0248	9.73	9.0248	10.84
PI-5	9	3	8.0258	2.89	8.0258	6.78	8.0258	6.92	8.0258	8.02	8.0258	9.72	8.0258	10.82
PI-6	9	4	7.5863	0.72	7.5863	6.78	7.5863	6.94	7.5863	8.03	7.5863	9.77	7.5863	10.85
PI-7	10	2	10.2635	0.78	10.2635	6.97	10.2635	7.09	10.2635	8.20	10.2635	10.02	10.2635	11.16
PI-8	10	3	9.0151	12.09	9.0151	7.00	9.0151	7.11	9.0151	8.23	9.0151	10.07	9.0151	11.21
PI-9	10	4	8.5098	1.69	8.5098	6.97	8.5098	7.09	8.5098	8.21	8.5098	10.02	8.5098	11.16
PI-10	11	2	11.4298	1.15	11.4298	7.10	11.4298	7.19	11.4298	8.32	11.4298	10.21	11.4298	11.43
PI-11	11	3	9.9455	59.43	9.9455	7.13	9.9455	7.21	9.9455	8.35	9.9455	10.26	9.9455	11.45
PI-12	11	4	9.3916	18.44	9.3916	7.15	9.3916	7.22	9.3916	8.36	9.3916	10.31	9.3916	11.49
PI-13	12	2	12.9442	2.03	12.9442	7.35	12.9442	7.39	12.9442	8.64	12.9442	10.63	12.9442	11.92
PI-14	12	3	11.2608	209.34	11.2608	7.28	11.2608	7.34	11.2608	8.52	11.2608	10.52	11.2608	11.76
PI-15	12	4	10.4824	47.50	10.4824	7.30	10.4824	7.37	10.4824	8.55	10.4824	10.53	10.4824	11.80
PI-16	13	2	14.4643	5.19	14.5251	7.52	14.4643	7.53	14.4643	8.71	14.4643	10.81	14.4643	12.11
PI-17	13	3	12.4245	975.45	12.5388	7.49	12.4941	7.52	12.4941	8.72	12.4245	10.82	12.4245	12.13
PI-18	13	4	11.5453	655.94	11.7222	7.51	11.6954	7.54	11.6954	8.74	11.5453	10.87	11.5453	12.17
PI-19	14	2	15.8371	8.70	16.2517	7.67	16.1942	7.68	16.1570	8.90	15.8971	11.11	15.8371	12.45
PI-20	14	3	13.4706	5453.10	13.8648	7.69	13.8244	7.71	13.8168	8.90	13.5906	11.16	13.4706	12.50
PI-21	14	4	12.4624	4674.83	12.8537	7.68	12.8128	7.70	12.7864	8.91	12.6624	11.12	12.5624	12.48
PI-22	15	2	17.1866	14.59	17.7417	7.86	17.7264	7.89	17.6901	9.14	17.4966	11.41	17.3966	12.84
PI-23	15	3	14.4428	7203.65	14.0908	7.87	14.0395	7.88	13.7550	9.13	13.7275	11.39	13.5847	12.84
PI-24	15	4	13.2278	7203.71	13.1679	7.91	13.0275	7.91	12.7758	9.19	12.7540	11.43	12.5721	12.88
PI-25	16	2	18.7701	28.98	19.4133	7.99	19.3787	8.05	19.3294	9.28	19.1830	11.69	19.0101	13.12
PI-26	16	3	15.5692	7202.86	15.0605	8.00	14.9772	8.04	14.6616	9.28	14.6100	11.68	14.5859	13.17
PI-27	16	4	14.1527	7202.95	14.0320	8.01	13.9020	8.08	13.6510	9.33	13.6272	11.71	13.4999	13.20
PI-28	17	2	20.3710	72.00	21.0907	8.13	21.0606	8.23	20.9897	9.45	20.8502	11.94	20.6358	13.45
PI-29	17	3	16.7674	7202.66	16.1970	8.16	16.1116	8.27	15.7584	9.47	15.7084	12.00	15.7014	13.50
PI-30	17	4	15.1667	7202.42	15.0326	8.17	14.8568	8.27	14.6034	9.50	14.5534	11.99	14.4534	13.52
Average:			**12.1899**	**1848.83**	**12.2474**	**7.41**	**12.2135**	**7.49**	**12.1489**	**8.66**	**12.0978**	**10.72**	**12.0528**	**12.00**

Table 18 Comparison of the considered algorithms against the exact optimization (CPLEX) for problem instances [PI-1 through PI-30].

Instance	No. of Vessels	No. of Berths	CPLEX		TS		SA		VNS		ACO		PSO	
			Objective, 10^6 USD	CPU, sec.	Objective, 10^6 USD	CPU, sec.	Objective, 10^6 USD	CPU, sec.	Objective, 10^6 USD	CPU, sec.	Objective, 10^6 USD	CPU, sec.	Objective, 10^6 USD	CPU, sec.
PI-1	8	2	7.9382	0.31	7.9382	0.17	7.9382	0.13	7.9382	0.11	7.9382	10.44	7.9382	6.79
PI-2	8	3	7.1958	0.81	7.1958	0.17	7.1958	0.13	7.1958	0.11	7.1958	10.44	7.1958	6.78
PI-3	8	4	6.8256	0.22	6.8256	0.17	6.8256	0.13	6.8256	0.11	6.8256	10.43	6.8256	6.91
PI-4	9	2	9.0248	0.59	9.0248	0.18	9.0248	0.14	9.0248	0.11	9.0248	10.87	9.0248	7.20
PI-5	9	3	8.0258	2.89	8.0258	0.18	8.0258	0.14	8.0258	0.11	8.0258	10.89	8.0258	7.22
PI-6	9	4	7.5863	0.72	7.5863	0.18	7.5863	0.14	7.5863	0.11	7.5863	10.91	7.5863	7.33
PI-7	10	2	10.2635	0.78	10.2635	0.18	10.2635	0.14	10.2635	0.12	10.2635	11.30	10.2635	7.63
PI-8	10	3	9.0151	12.09	9.0151	0.18	9.0151	0.14	9.0151	0.12	9.0151	11.07	9.0151	7.46
PI-9	10	4	8.5098	1.69	8.5098	0.18	8.5098	0.14	8.5098	0.12	8.5098	11.06	8.5098	7.59
PI-10	11	2	11.4298	1.15	11.4298	0.18	11.4298	0.14	11.4298	0.12	11.4298	11.49	11.4298	7.84
PI-11	11	3	9.9455	59.43	9.9455	0.18	9.9455	0.14	9.9455	0.12	9.9455	11.49	9.9455	7.85
PI-12	11	4	9.3916	18.44	9.3916	0.19	9.3916	0.15	9.3916	0.12	9.3916	11.50	9.3916	8.00
PI-13	12	2	12.9442	2.03	12.9442	0.19	12.9442	0.15	12.9442	0.13	12.9442	11.92	12.9442	8.24
PI-14	12	3	11.2608	209.34	11.2608	0.19	11.2608	0.15	11.2608	0.13	11.2608	11.95	11.2608	8.27
PI-15	12	4	10.4824	47.50	10.4824	0.19	10.4824	0.15	10.4824	0.13	10.4824	11.93	10.4824	8.36
PI-16	13	2	14.4643	5.19	14.5251	0.20	14.5251	0.16	14.5251	0.13	14.5251	12.37	14.4643	8.71
PI-17	13	3	12.4245	975.45	12.5488	0.20	12.5488	0.16	12.5488	0.13	12.5388	12.41	12.5388	8.70
PI-18	13	4	11.5453	655.94	11.7422	0.20	11.7322	0.16	11.7322	0.13	11.7222	12.41	11.7054	8.83
PI-19	14	2	15.8371	8.70	16.2817	0.20	16.2747	0.16	16.2617	0.14	16.2517	12.95	16.2042	9.09
PI-20	14	3	13.4706	5453.10	13.8948	0.20	13.8878	0.16	13.8748	0.14	13.8648	12.99	13.8344	9.09
PI-21	14	4	12.4624	4674.83	12.8737	0.21	12.8657	0.16	12.8637	0.14	12.8437	13.01	12.8228	9.23
PI-22	15	2	17.1866	14.59	17.7664	0.21	17.7584	0.17	17.7464	0.15	17.7364	13.53	17.7344	9.51
PI-23	15	3	14.4428	7203.65	14.1108	0.21	14.1028	0.17	14.0908	0.15	14.0808	13.57	14.0495	9.54
PI-24	15	4	13.2278	7203.71	13.1879	0.21	13.1799	0.17	13.1779	0.15	13.1579	13.60	13.0375	9.70
PI-25	16	2	18.7701	28.98	19.4333	0.21	19.4273	0.17	19.4233	0.15	19.4033	13.98	19.3887	9.92
PI-26	16	3	15.5692	7202.86	15.0805	0.21	15.0725	0.17	15.0705	0.15	15.0505	13.98	14.9872	9.92
PI-27	16	4	14.1527	7202.95	14.0520	0.22	14.0440	0.17	14.0420	0.15	14.0220	13.98	13.9120	10.04
PI-28	17	2	20.3710	72.00	21.1207	0.22	21.1077	0.18	21.1007	0.16	21.0807	14.46	21.0706	10.36
PI-29	17	3	16.7674	7202.66	16.2270	0.22	16.2070	0.18	16.2070	0.16	16.1870	14.46	16.1216	10.33
PI-30	17	4	15.1667	7202.42	15.0626	0.23	15.0426	0.18	15.0426	0.16	15.0226	14.48	14.8668	10.46
Average:			**12.1899**	**1848.83**	**12.2582**	**0.20**	**12.2539**	**0.15**	**12.2516**	**0.13**	**12.2442**	**12.33**	**12.2193**	**8.56**

The results of the conducted analysis are summarized in Table 17 and Table 18, including the instance number, the number of arriving vessels at the MCT, the number of the MCT berthing positions available to serve the vessels, the obtained objective function values (i.e., the benchmark objective function values that were obtained by CPLEX, and the average objective function values that were returned by the considered algorithms), and the computational time values required.

Based on the analysis results provided in Table 17 and Table 18, it was found that the CPLEX computational time was substantially influenced with the problem size. The latter finding highlights the high computational complexity of the DBSP mathematical model. Therefore, the DBSP mathematical model needs to be solved by heuristic and/or metaheuristic algorithms for the large-size problem instances in order to obtain solutions within a reasonable computational time. Note that the problem instances PI-23, PI-24, PI-26, PI-27, PI-29, and PI-30 could not be solved to the global optimality during the limited computational time defined for CPLEX (i.e., 7,200 sec). On the other hand, the considered metaheuristic algorithms solved the small-size problem instances in less than 14 sec. Moreover, the adopted algorithms were able to return the global optimal solutions (i.e., solutions with the same objective values as compared to the benchmarks provided by CPLEX) for many of the generated small-size problem instances (i.e., the optimality gaps were found to be equal to zero). However, for certain small-size problem instances, the considered algorithms could not find the global optimal solutions. For instance, the average objective function value that was returned by the EA algorithm for problem instance PI-16 was 0.42% worse than the optimal one.

The maximum optimality gaps for the developed algorithms were estimated over the generated small-size problem instances. By changing the vessel arrival times ($t^a_{v+1} = t^a_v + \Delta t \; \forall v \in V, \Delta t = exp[2]$) and the handling productivities ($p_{vb} = U[130; 160] \; \forall v \in V, b \in B$), a total of 50 scenarios were developed for each small-size problem instance. A total of 10 replications were executed for each developed algorithm, aiming to estimate the average objective function values for each scenario and each small-size problem instance. The optimality gaps were computed using on the following relationship:

$$\boldsymbol{OG}_{Alg} = \frac{\boldsymbol{TC}_{Alg} - \boldsymbol{TC}_{CPLEX}}{\boldsymbol{TC}_{CPLEX}} \quad (42)$$

where $\boldsymbol{OG}_{Alg}$ is the optimality gap of algorithm Alg; $\boldsymbol{TC}_{CPLEX}$ is the optimal objective function value (i.e., the total vessel service cost), obtained by CPLEX; and $\boldsymbol{TC}_{alg}$ is the average objective function value over 10 replications, returned by algorithm Alg.

Figure 32 illustrates the optimality gaps over the scenarios, generated for the considered algorithms and all the developed small-size problem instances, using a boxplot format. Each boxplot is represented by a rectangle. The boxplot rectangle covers the range, where the bottom of a rectangle represents the 25th optimality gap percentile, while the top of a rectangle corresponds to the 75th optimality gap percentile. The median optimality gap is shown using an additional line within each rectangle. The dashed lines (or "whiskers") extend from the bottom and the top of each rectangle, covering 99.3% of the optimality gap values. The outliers, which are shown using symbol "+", are the optimality gap values falling outside the range covered by whiskers. The conducted optimality gap analysis proves the high accuracy of the developed algorithms. The maximum optimality gaps were found to be lower than ≈4.5% for the generated scenarios and considered small-size problem instances. It can be observed that smaller optimality gaps were generally recorded for the SAEA and ASAEA algorithms, where the maximum ASAEA optimality gap did not exceed 1.59%. As CPLEX was not able to find the global optimal solutions for problem instances PI-23, PI-24, PI-26, PI-27, PI-29, and PI-30 within the imposed computational time limit, the optimality gaps are not shown in boxplots for the aforementioned problem instances (see Figure 32).

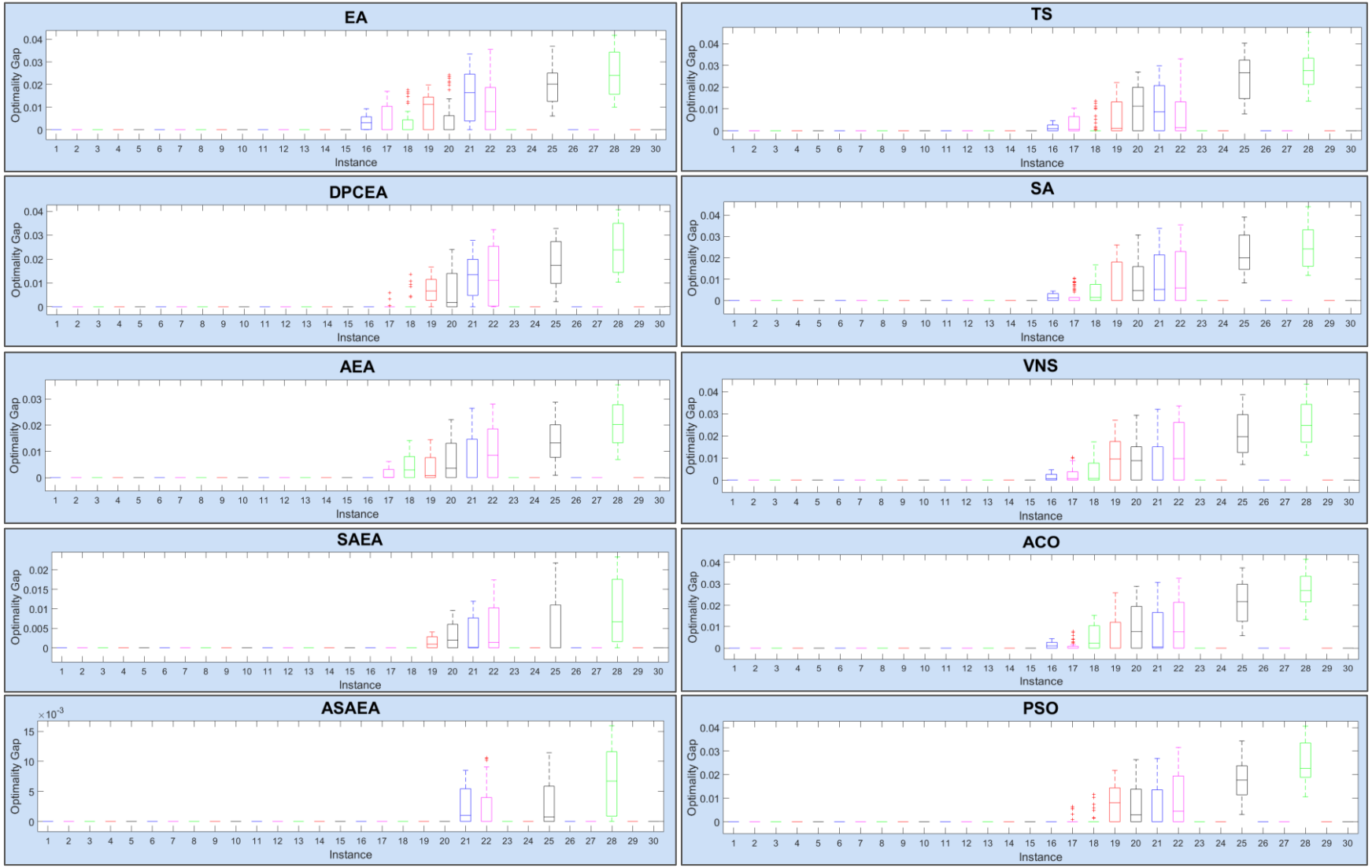

Figure 32 The optimality gaps over the generated scenarios for the considered algorithms and problem instances [PI-1 through PI-30].

6.3.4. Comprehensive comparative analysis against the alternative metaheuristics

The fourth step of the computational experiments was undertaken for the large-size problem instances [PI-31 through PI-60] to conduct a comprehensive comparative analysis of the developed ASAEA algorithm against the alternative EA-based and state-of-the-art metaheuristics in terms of the following performance indicators: (a) objective function and computational time values; (b) convergence patterns; and (c) objective function stochastic variations. In order to obtain the average objective function and computational time values for the large-size problem instances, a total of 10 replications were performed for each algorithm. More information regarding the analysis results is provided in sections 6.3.4.1-6.3.4.3 of the dissertation.

6.3.4.1. Algorithmic performance evaluation for the large-size problem instances

The instance number, the number of vessels arriving for service at the MCT, the number of the MCT berthing positions available for vessel service, the average objective function values that were returned by the considered algorithms, and the required computational time values are presented in Table 19 and Table 20. Considering the provided information in Table 19 and Table 20, it can be observed that the ASAEA algorithm clearly outperforms the alternative EA-based and state-of-the-art metaheuristics in terms of the objective function values at convergence. The average objective function value improvements that were achieved by ASAEA over the alternative EA-based metaheuristics are provided in Figure 33, while Figure 34 demonstrates the average objective function value improvements that were achieved by ASAEA over the alternative state-of-the-art metaheuristics. The numerical experiments, conducted using the generated large-size problem instances, showed that the ASAEA algorithm outperformed the EA, DPCEA, AEA, and SAEA algorithms on average by 16.41%, 15.03%, 12.50%, and 8.29%, respectively, in terms of the objective function values. Moreover, for the generated large-size problem instances, the ASAEA algorithm was superior to the TS, SA, VNS, ACO, and PSO algorithms on average by 19.25%, 19.20%, 18.88%, 16.30%, and 16.08%, respectively, in terms of the objective function values.

Table 19 Objective function and computational time values obtained by the considered algorithms for problem instances [PI-31 through PI-60].

Instance	No. of Vessels	No. of Berths	EA		DPCEA		AEA		SAEA		ASAEA	
			Objective, 10^6 USD	CPU, sec.	Objective, 10^6 USD	CPU, sec.	Objective, 10^6 USD	CPU, sec.	Objective, 10^6 USD	CPU, sec.	Objective, 10^6 USD	CPU, sec.
PI-31	75	6	60.877	55.29	60.354	55.66	59.336	98.25	57.077	98.29	52.086	142.88
PI-32	75	8	57.821	55.23	56.585	55.58	55.410	98.97	53.444	99.31	49.843	146.88
PI-33	75	10	56.294	55.46	55.693	56.07	54.596	99.86	53.018	99.91	48.380	145.21
PI-34	80	6	65.827	58.72	64.967	59.11	63.632	105.19	62.017	105.30	57.017	152.98
PI-35	80	8	61.976	57.83	61.664	58.28	59.583	104.91	57.682	104.96	54.168	153.04
PI-36	80	10	61.920	58.23	60.648	58.60	59.384	105.12	57.058	105.18	53.270	153.28
PI-37	85	6	71.701	60.96	70.843	61.63	69.835	110.95	66.792	111.65	61.688	164.79
PI-38	85	8	67.176	61.58	66.822	61.98	65.221	111.28	61.948	111.69	57.048	165.27
PI-39	85	10	65.764	61.00	65.712	61.76	64.430	110.50	61.978	110.68	56.478	163.23
PI-40	90	6	76.910	64.37	76.537	65.11	73.823	116.66	71.594	116.95	66.843	172.19
PI-41	90	8	70.656	64.90	70.080	65.06	69.390	117.65	66.484	117.74	60.813	175.49
PI-42	90	10	71.143	65.08	70.207	65.24	67.971	117.69	65.637	119.11	60.319	173.14
PI-43	95	6	82.389	68.87	81.070	69.38	79.836	122.88	76.118	123.34	70.788	182.74
PI-44	95	8	76.146	68.97	75.318	69.58	72.512	125.38	70.659	125.42	64.761	184.10
PI-45	95	10	75.491	67.73	73.968	68.57	72.788	123.20	69.799	123.31	65.230	182.12
PI-46	100	6	86.876	71.55	86.681	72.12	85.159	131.35	81.433	131.64	75.549	192.83
PI-47	100	8	79.747	72.69	78.781	72.79	77.777	131.93	74.292	131.97	68.251	193.00
PI-48	100	10	79.136	71.39	78.070	72.98	76.524	129.53	73.625	130.38	67.892	192.10
PI-49	105	6	93.265	74.40	91.115	75.07	89.702	135.36	85.991	135.61	79.010	200.57
PI-50	105	8	83.025	74.65	82.853	75.41	81.266	136.47	78.121	136.57	71.926	201.19
PI-51	105	10	82.580	74.39	82.513	76.23	79.537	135.62	76.909	135.79	72.214	200.00
PI-52	110	6	97.995	77.39	97.218	77.96	94.429	141.03	90.845	141.22	83.603	208.26
PI-53	110	8	87.435	78.94	85.777	79.40	84.606	143.78	81.786	144.20	74.692	212.45
PI-54	110	10	86.655	77.58	86.199	78.61	82.986	140.72	80.906	141.99	74.317	208.53
PI-55	115	6	104.241	79.90	101.522	80.49	100.522	146.28	96.251	146.31	88.620	213.36
PI-56	115	8	92.329	80.74	90.786	81.50	88.430	147.56	85.099	148.29	78.791	214.20
PI-57	115	10	91.314	79.60	88.696	80.15	87.251	145.71	84.549	145.81	78.404	212.61
PI-58	120	6	109.817	82.73	108.911	83.28	105.920	151.30	101.529	151.50	93.805	222.81
PI-59	120	8	96.543	83.86	95.123	84.65	92.848	153.40	89.727	154.14	83.543	223.54
PI-60	120	10	94.593	82.71	94.216	83.51	92.290	151.70	87.718	151.91	81.097	220.97
Average:			**79.588**	**69.56**	**78.631**	**70.19**	**76.900**	**126.34**	**74.003**	**126.67**	**68.348**	**185.79**

Table 20 Objective function and computational time values obtained by the considered algorithms for problem instances [PI-31 through PI-60].

Instance	No. of Vessels	No. of Berths	TS		SA		VNS		ACO		PSO	
			Objective, 10^6 USD	CPU, sec.	Objective, 10^6 USD	CPU, sec.	Objective, 10^6 USD	CPU, sec.	Objective, 10^6 USD	CPU, sec.	Objective, 10^6 USD	CPU, sec.
PI-31	75	6	62.703	1.17	62.452	1.06	62.485	0.99	61.524	58.67	61.208	49.53
PI-32	75	8	58.549	1.18	58.734	1.07	58.464	1.01	56.740	59.36	56.930	49.26
PI-33	75	10	58.392	1.18	57.945	1.07	58.135	1.00	56.091	59.44	56.992	50.29
PI-34	80	6	67.715	1.23	67.864	1.11	67.185	1.05	66.283	61.50	66.010	52.07
PI-35	80	8	63.631	1.23	63.517	1.11	63.854	1.05	62.160	61.81	62.560	52.50
PI-36	80	10	63.006	1.23	62.746	1.12	63.030	1.05	61.593	62.40	61.553	53.05
PI-37	85	6	73.374	1.30	73.558	1.17	73.552	1.13	70.947	64.93	71.361	54.72
PI-38	85	8	68.782	1.29	68.457	1.17	68.629	1.14	67.022	65.59	66.444	55.37
PI-39	85	10	67.952	1.28	68.134	1.16	67.665	1.12	66.724	65.04	66.766	55.13
PI-40	90	6	78.446	1.34	79.200	1.22	78.721	1.18	76.346	68.06	76.105	58.10
PI-41	90	8	73.112	1.36	72.964	1.24	72.137	1.21	71.437	69.44	70.858	59.19
PI-42	90	10	72.514	1.35	72.530	1.23	72.262	1.18	71.054	69.09	70.690	58.79
PI-43	95	6	84.371	1.41	84.157	1.28	84.252	1.25	82.148	71.58	82.092	61.12
PI-44	95	8	77.579	1.43	77.249	1.30	76.529	1.27	76.224	72.91	75.666	62.29
PI-45	95	10	76.791	1.40	76.802	1.27	75.738	1.23	75.498	71.63	75.223	61.80
PI-46	100	6	89.494	1.49	89.551	1.36	89.311	1.33	86.547	76.25	87.344	65.39
PI-47	100	8	82.071	1.50	81.681	1.37	82.216	1.35	79.114	76.78	79.863	65.61
PI-48	100	10	81.415	1.48	81.050	1.36	81.291	1.32	79.094	76.21	78.690	64.86
PI-49	105	6	94.534	1.53	94.934	1.39	94.700	1.36	93.431	78.58	91.998	67.00
PI-50	105	8	86.255	1.53	86.127	1.40	85.240	1.38	84.114	78.63	83.392	66.93
PI-51	105	10	84.567	1.56	85.129	1.42	84.881	1.39	82.981	79.90	82.937	68.09
PI-52	110	6	99.835	1.60	99.862	1.46	99.534	1.43	97.028	81.96	97.048	70.72
PI-53	110	8	89.488	1.61	90.194	1.47	88.911	1.44	88.351	82.93	87.399	71.02
PI-54	110	10	89.117	1.59	88.761	1.45	88.803	1.42	87.308	81.72	86.748	69.85
PI-55	115	6	106.296	1.65	106.399	1.52	106.326	1.46	104.038	83.71	102.364	72.98
PI-56	115	8	94.402	1.65	93.780	1.52	94.153	1.46	91.538	83.93	91.100	72.88
PI-57	115	10	92.808	1.65	92.777	1.51	92.848	1.44	90.153	83.51	90.500	72.51
PI-58	120	6	112.379	1.72	111.961	1.58	111.562	1.52	108.751	87.22	108.809	76.18
PI-59	120	8	98.321	1.71	98.574	1.59	98.023	1.51	96.117	87.37	95.842	76.04
PI-60	120	10	97.304	1.71	97.351	1.58	97.240	1.51	94.335	86.89	95.009	75.90
Average:			**81.507**	**1.45**	**81.481**	**1.32**	**81.256**	**1.27**	**79.490**	**73.57**	**79.317**	**62.97**

The efficiency of the augmented self-adaptive strategy, proposed for parameter control, can be confirmed due to superiority of the ASAEA algorithm over the alternative EA-based and state-of-the-art metaheuristics, where not only the algorithmic parameter values are encoded in the solutions, but they are also updated based on the feedback from the search. Other metaheuristic algorithms that rely on parameter control strategy (i.e., DPCEA, AEA, and SAEA) returned solutions with higher quality (i.e., lower objective function values) as compared to the metaheuristic algorithms that solely rely on parameter tuning (i.e., EA, TS, SA, VNS, ACO, and PSO). The latter finding can be justified by the fact that the parameter tuning strategy does not evolve the parameter values throughout the algorithmic run, which may not be favorable for the search process (as it was demonstrated throughout the conducted analysis). Furthermore, single-solution-based algorithms (i.e., TS, SA, and VNS) were generally outperformed by population-based algorithms (i.e., EA, DPCEA, AEA, SAEA, ASAEA, ACO, and PSO), as they work with one solution at a time, which negatively affects the explorative capabilities of such algorithms.

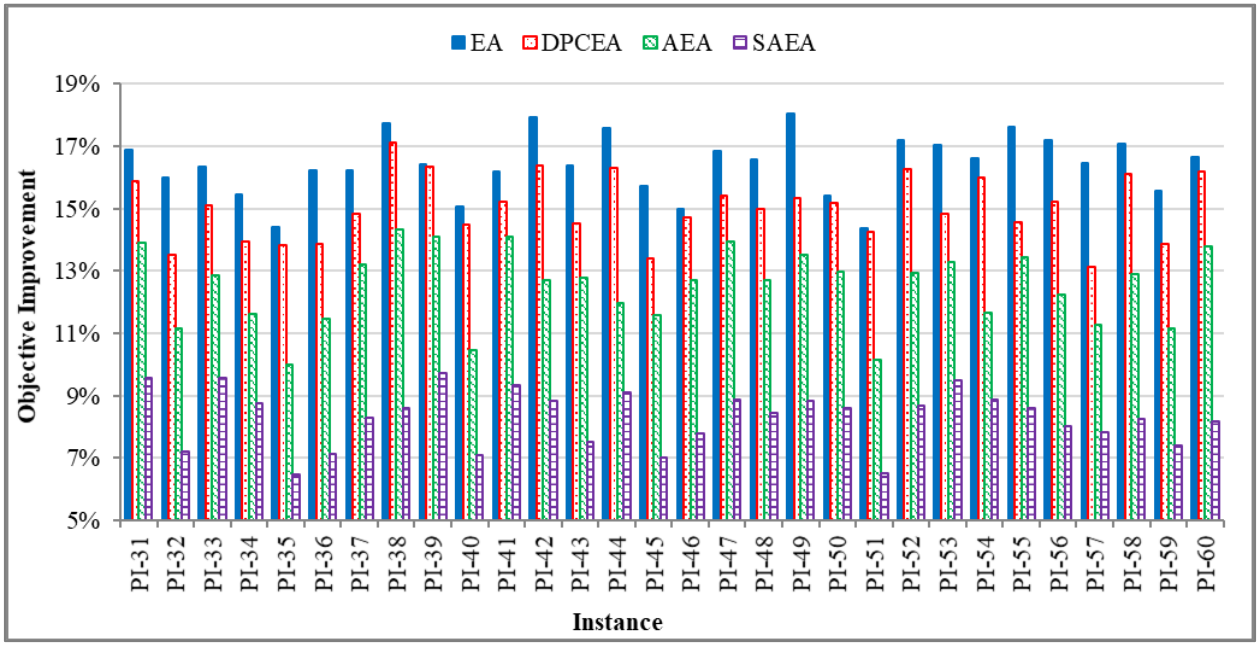

Figure 33 The average objective function value improvements that were achieved by ASAEA over the alternative EA-based metaheuristics for problem instances [PI-31 through PI-60].

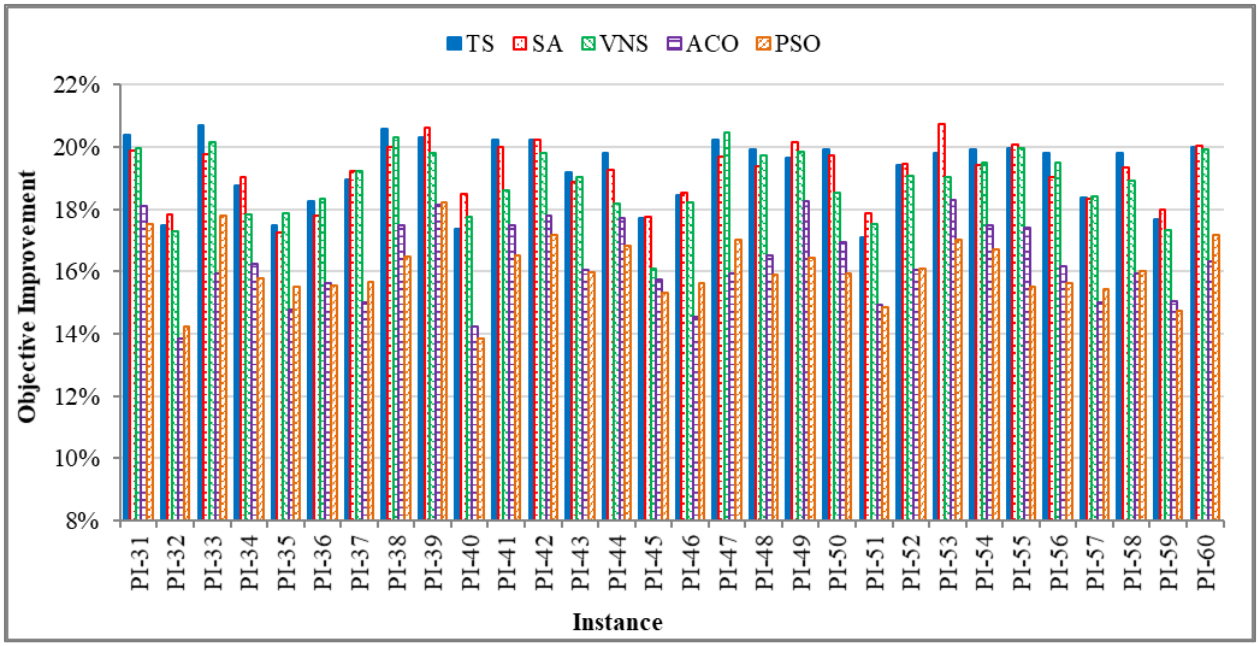

Figure 34 The average objective function value improvements achieved by ASAEA over the alternative state-of-the-art metaheuristics for problem instances [PI-31 through PI-60].

The numerical experiments, conducted for the large-size problem instances, also included a statistical analysis. Specifically, the difference between the average objective function values returned by ASAEA and other alternative algorithms (i.e., EA, DPCEA, AEA, SAEA, TS, SA, VNS, ACO, and PSO) was analyzed to identify whether they are statistically different. The latter task was accomplished by conducting a set of paired z-tests. The null hypothesis (H_0) was defined as follows: the average objective function values, returned by the ASAEA algorithm is equal to the average objective function values, returned by alternative algorithm Alg, which can be denoted as – H_0: $\boldsymbol{TC}_{ASAEA} = \boldsymbol{TC}_{Alg}$. Conversely, the alternative hypothesis (H_a) assumed that the average objective function values, returned by the ASAEA algorithm, were different from the average objective function values, returned by alternative algorithm Alg – H_a: $\boldsymbol{TC}_{ASAEA} \neq \boldsymbol{TC}_{Alg}$. The ASAEA objective function values were compared against the objective function values returned by 9 alternative metaheuristic algorithms for a total of 9 paired z-tests. Table 21 summarizes the results from the conducted paired z-tests, including the test number, the candidate algorithms compared, the number of observations considered (which is equal to 30, as all the generated large-size problem instances were considered), the average objective function values ($\boldsymbol{TC}$) obtained, the objective function value standard deviations ($\boldsymbol{STD}$), and the test statistic values (i.e., z-values).

Table 21 The results obtained from the conducted paired z-tests.

Test	Algorithms Compared	Number of Observations	TC, 10^6 USD	STD, 10^6 USD	Test Statistic (z)
1	ASAEA	30	68.348	12.023	3.312
	EA	30	79.588	14.174	
2	ASAEA	30	68.348	12.023	3.063
	DPCEA	30	78.631	13.910	
3	ASAEA	30	68.348	12.023	2.579
	AEA	30	76.900	13.616	
4	ASAEA	30	68.348	12.023	1.750
	SAEA	30	74.003	12.989	
5	ASAEA	30	68.348	12.023	3.844
	TS	30	81.507	14.387	
6	ASAEA	30	68.348	12.023	3.832
	SA	30	81.481	14.418	
7	ASAEA	30	68.348	12.023	3.774
	VNS	30	81.256	14.366	
8	ASAEA	30	68.348	12.023	3.305
	ACO	30	79.490	14.012	
9	ASAEA	30	68.348	12.023	3.277
	PSO	30	79.317	13.842	

The null hypothesis (with a 5.00% significance level, which corresponds to a critical z-value of 1.96) was rejected for all the conducted paired z-tests, except test "4". Therefore, the average objective function values, which were returned by the ASAEA algorithm for the large-size problem instances, were statistically different (i.e., statistically lower) as compared to the average objective function values, which were returned by the EA, DPCEA, AEA, TS, SA, VNS, ACO, and PSO algorithms. The null hypothesis was not rejected for test "4" at a 5.00% significance level; however, ASAEA outperformed SAEA on average by 8.29% in terms of the objective function values for the considered large-size problem instances, which can be considered a remarkable difference from the practical standpoint.

The highest computational time values were generally recorded for ASAEA in the conducted numerical experiments, which can be explained by two major factors. First, encoding the crossover and mutation probabilities within the ASAEA chromosomes resulted in the additional operations in order to alter the crossover and mutation probabilities throughout the search process. Second, storing the promising crossover and mutation probability values in an external archive during each generation along with periodic updates of the crossover and mutation probability values in the current population chromosomes impose additional computational time. However, the maximum ASAEA computational time did not exceed 3.73 min. Hence, the MCT

operators will be able to fairly quickly develop and revise berthing plans as needed. In the meantime, other algorithms that adopt other parameter control strategies (i.e., DPCEA, AEA, and SAEA) typically required more time to converge in comparison with the algorithms that solely rely on parameter tuning (i.e., EA, TS, SA, VNS, ACO, and PSO). This pattern can be supported by the fact that the additional operations, which are required for updating the algorithmic parameter values, will increase the computational time. Besides, single-solution-based algorithms (i.e., TS, SA, and VNS) converged in less computational time as compared to population-based algorithms (i.e., EA, DPCEA, AEA, SAEA, ASAEA, ACO, and PSO), as they work with just one solution at a time.

6.3.4.2. Convergence pattern analysis

The objective function value improvements from one generation to another (in case of the considered EA-based metaheuristics – EA, DPCEA, AEA, SAEA, and ASAEA) or from one iteration to another (in case of the considered state-of-the-art metaheuristics – TS, SA, VNS, ACO, and PSO) can be followed through the analysis of convergence patterns. The convergence patterns were recorded for all the metaheuristic algorithms developed for each replication and each one of the generated large-size problem instances. Figure 35 illustrates the convergence patterns for the EA, DPCEA, AEA, SAEA, and ASAEA algorithms, while the convergence patterns of the TS, SA, VNS, ACO, and PSO algorithms are depicted in Figure 36. The convergence patterns are shown only for the last replication and problem instances [PI-52 through PI-60], where similar tendencies were observed for other replications and large-size problem instances. Note that the ASAEA convergence patterns are presented in Figure 36 as well for comparison purposes.

As the initial solutions of all the algorithms were generated using the same heuristic (i.e., the FCFS-SR heuristic – see section 6.2.3 of the dissertation), the convergence patterns indicate that all the considered metaheuristic algorithms start the search process with the same solution. According to the performed convergence pattern analysis, it can be observed that ASAEA was able to discover the promising solutions of the search space significantly faster as compared to the alternative EA-based and state-of-the-art metaheuristics. Hence, the deployed augmented self-adaptive parameter control strategy within the developed ASAEA algorithm allows effective exploration of the search space as well as effective identification of the domains with high-quality

solutions. Furthermore, the ASAEA exploitative capabilities were improved after application of the proposed augmented self-adaptive parameter control strategy, since the ASAEA algorithm was able to find the solutions with superior objective function values within the identified domains of the search space before convergence (i.e., substantial objective function improvements were recorded even after generation ≈5000).

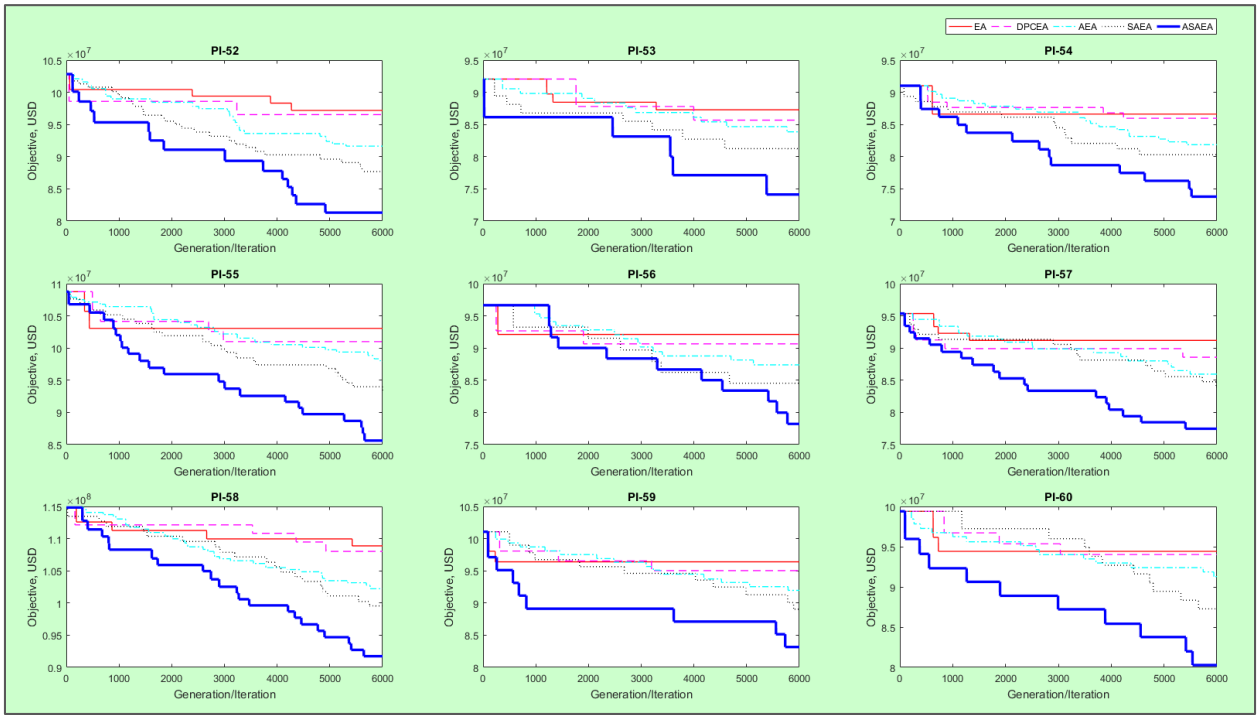

Figure 35 The EA, DPCEA, AEA, SAEA, and ASAEA convergence patterns for problem instances [PI-52 through PI-60].

The promising solutions were discovered fairly quickly by the SAEA and AEA algorithms, which rely on the self-adaptive and adaptive parameter control strategies, respectively. Moreover, it was found from the convergence diagrams that the explorative capabilities declined for DPCEA as compared to AEA, SAEA, and ASAEA. The latter finding highlights that the augmented self-adaptive, self-adaptive, and adaptive parameter control strategies are more promising for the search process than the deterministic parameter control strategy, which was adopted within the DPCEA solution algorithm. The algorithms that solely rely on parameter tuning (i.e., EA, TS, SA, VNS, ACO, and PSO) demonstrated lack of the explorative and exploitative capabilities. The latter tendency can be justified by the fact that the parameter tuning strategy keeps the parameter values

constant throughout the algorithmic run, which may negatively influence the overall search process. Moreover, the single-solution-based algorithms (i.e., TS, SA, and VNS) generally showed a premature convergence. As the single-solution-based algorithms execute the search operations by utilizing just one solution at a time, they cannot effectively explore various areas of the search space, which further leads to a premature convergence.

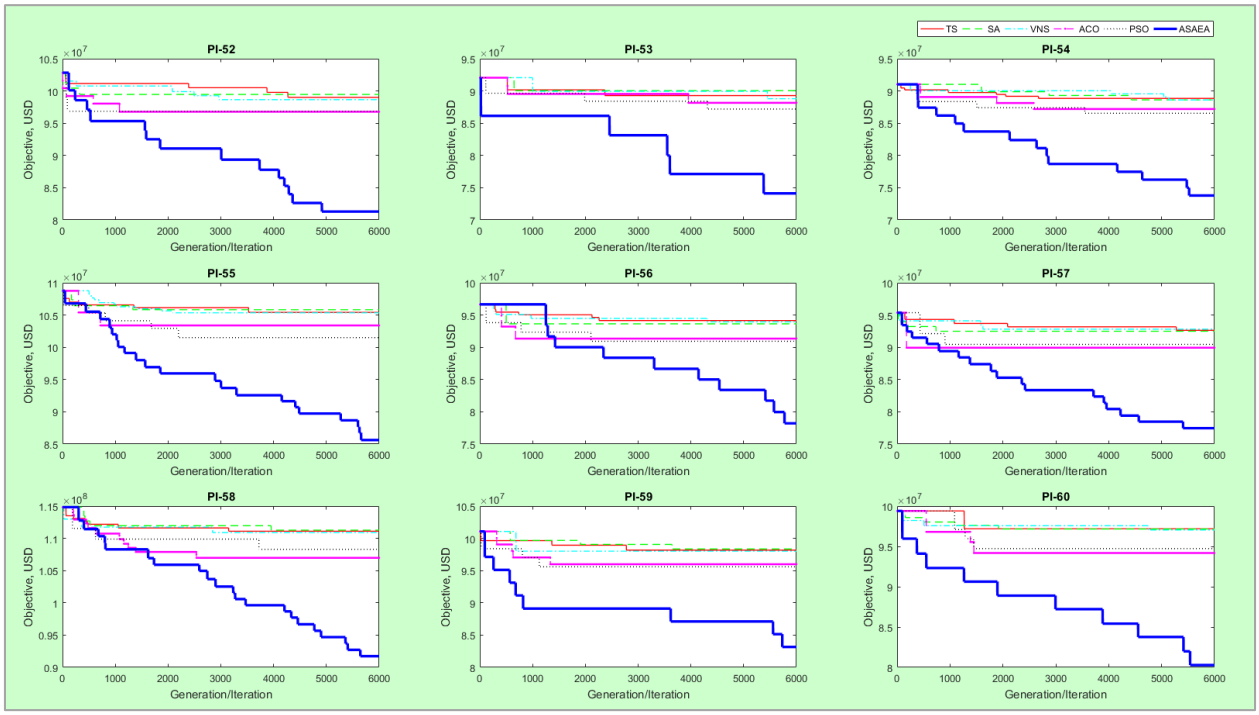

Figure 36 The TS, SA, VNS, ACO, PSO, and ASAEA convergence patterns for problem instances [PI-52 through PI-60].

6.3.4.3. Analysis of the objective function stochastic variations

Although the EA-based and state-of-the-art metaheuristic algorithms deploy stochastic operators (e.g., crossover, mutation, selection), they were found to be efficient for solving complex decision problems (Eiben and Smith, 2003). Stochastic nature of the algorithms further leads to variations in the objective function values, obtained by such algorithms at termination, from one algorithmic run to another. If the objective function values, which were returned by a given algorithm, show significant stochastic variations, such algorithm cannot be considered as a reliable decision support tool in practice (e.g., the MCT operator will not be able to develop a berth schedule using a given algorithm, if the total vessel service cost substantially fluctuates from one

algorithmic replication to another). Hence, estimation of the objective function stochastic variations for the candidate metaheuristic algorithms was included in the scope of the computational experiments in this chapter of the dissertation. The coefficient of variation was estimated for the objective function values, returned by each algorithm, over 10 replications and for each large-size problem instance. A boxplot format was adopted to illustrate the results of the conducted analysis for each algorithm over all the generated large-size problem instances (Figure 37). The boxplot components have been explained comprehensively in section 6.3.3 of this dissertation.

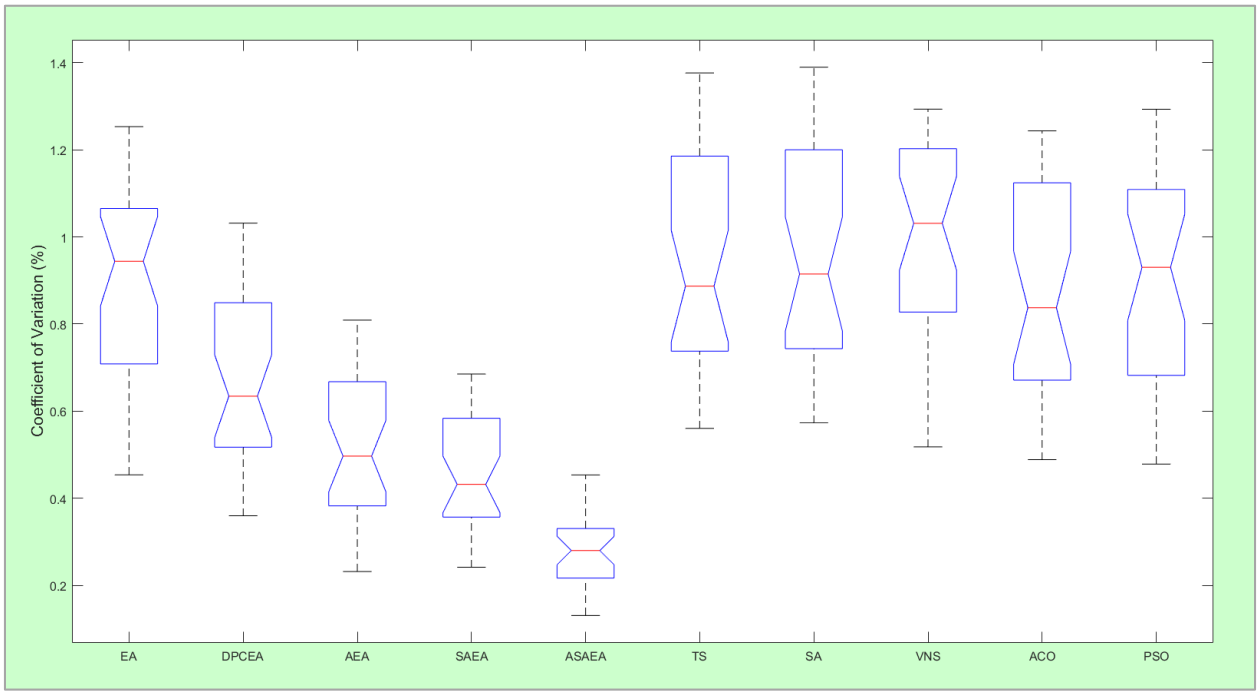

Figure 37 The objective function coefficient of variation values for the developed algorithms and problem instances [PI-31 through PI-60].

It can be observed in Figure 37 that the coefficient of variation over the objective function values did not exceed 1.4% for all the metaheuristic algorithms, which can be viewed as acceptable from the practical standpoint. The EA-based algorithms that rely on parameter control (i.e., DPCEA, AEA, SAEA, and ASAEA) demonstrated lower coefficient of variation values when comparing to the other considered state-of-the-art algorithms. Furthermore, the objective function values, which were returned by the ASAEA algorithm, showed the least coefficient of variation values. Specifically, the ASAEA objective function coefficient of variation values did not exceed

0.5% for the considered large-size problem instances. The latter finding underlines that the developed augmented self-adaptive parameter control strategy can strengthen the algorithmic performance and enhance the algorithmic stability by reducing the stochastic variations of the objective function values, returned by the algorithm at convergence.

6.3.5. Algorithmic parameter evolution

As it was mentioned earlier, the SAEA and ASAEA algorithms have the crossover and mutation probabilities encoded in the chromosomes, and evolve them in each generation throughout the algorithmic run. The scope of the fourth step of the computational experiments included the analysis of evolution of the crossover and mutation probability values, encoded within the SAEA and ASAEA chromosomes, from the beginning of the search process up to its convergence. The average crossover and mutation probability values (over all the individuals within the populations of SAEA and ASAEA, separately) were recorded in each generation and for each one of the developed large-size problem instances. Figure 38 and Figure 39 illustrate the evolution of the crossover and mutation probability values for the SAEA and ASAEA algorithms. Evolution of the crossover and mutation probability values is shown only for the last replication and problem instances [PI-52 through PI-60], whereas similar tendencies were observed for other replications and large-size problem instances.

According to the results obtained from the conducted analysis, it was found that fairly high values for the crossover probabilities (about 0.70-0.80) and for the mutation probabilities (about 0.05-0.06) were similarly adopted by both SAEA and ASAEA at the beginning of the search process. High crossover and mutation probability values made the SAEA and ASAEA algorithms capable of fast identification of the search space domains, which contain good-quality solutions. However, later in the search process, the crossover and mutation probability values started gradually decreasing for SAEA and ASAEA, aiming to effectively exploit the discovered search space domains for more promising solutions.

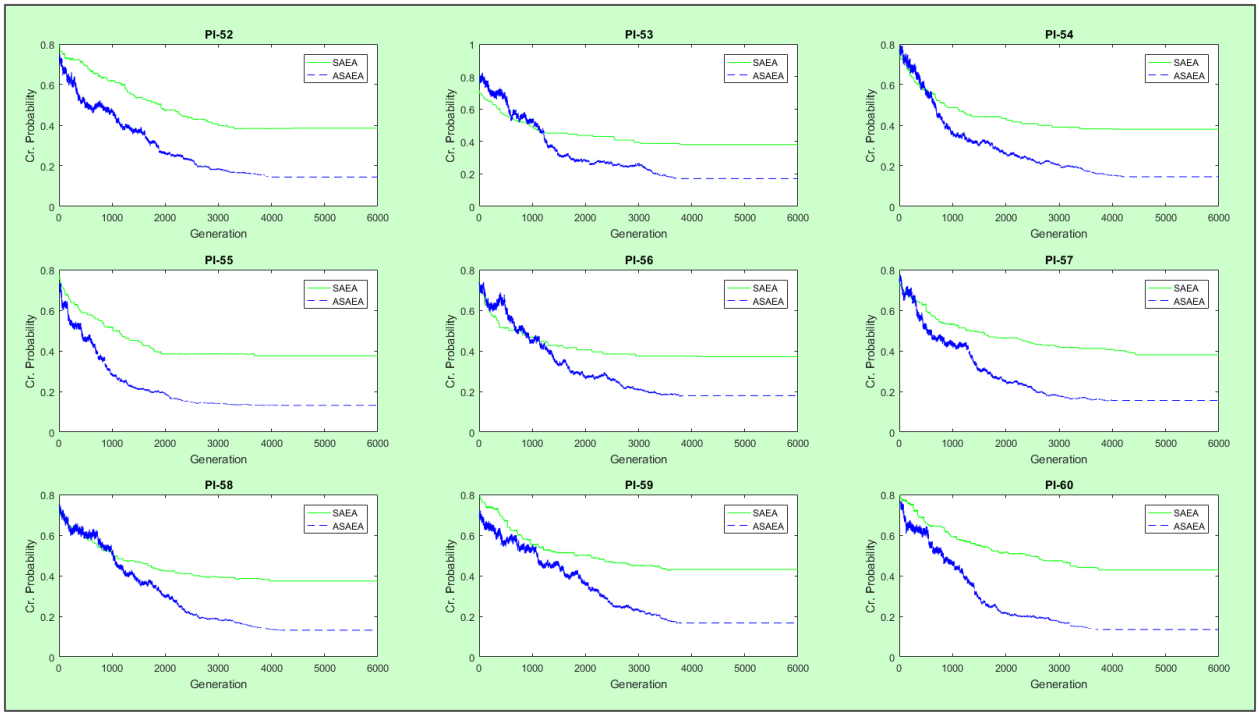

Figure 38 Evolution of the crossover probability values for the SAEA and ASAEA algorithms and problem instances [PI-52 through PI-60].

As it is observed in Figure 38 and Figure 39, the ASAEA algorithm adopted lower crossover and mutation probability values as compared to the SAEA algorithm (especially, later in the search process). Since the ASAEA algorithm outperformed the SAEA algorithm in terms of the objective function values, lower crossover and mutation probability values were more favorable for the search process towards the convergence (more details are provided in sections 6.3.4.1 and 6.3.4.2 of the dissertation). As the ASAEA algorithm deploys the proposed augmented self-adaptive parameter control strategy, it discovers the most promising crossover and mutation probability values periodically from the external archive. On the other hand, SAEA does not adjust the crossover and mutation probability values as effectively as ASAEA throughout the search process, since SAEA does not rely on the external archive.

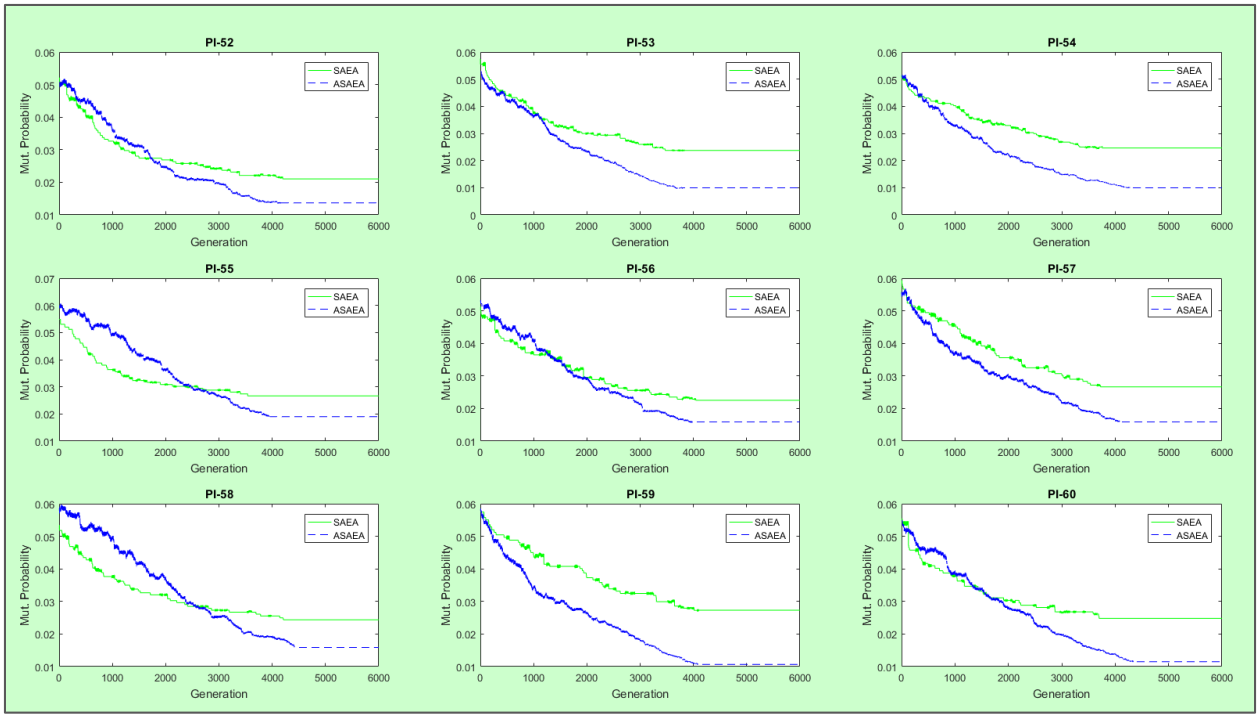

Figure 39 Evolution of the mutation probability values for the SAEA and ASAEA algorithms and problem instances [PI-52 through PI-60].

6.4. Conclusions for Chapter 6

This chapter of the dissertation proposed an augmented self-adaptive EA for the berth scheduling problem (BSP), where not only the crossover and mutation probabilities (which are considered as the major parameters of EAs) are encoded in the chromosomes and evolve throughout the search process, but also some of the pairs of crossover and mutation probabilities, which already showed enhancing impact on the corresponding solutions, would be propagated in the population. In particular, augmented self-adaptive strategy addresses the main drawback of the self-adaptive approach, which does not consider any feedback from the search throughout the evolution of the algorithmic parameters. Extensive computational experiments were implemented to evaluate performance of the developed algorithms against the exact optimization algorithm. The maximum optimality gap, calculated over the obtained benchmark objective function values by CPLEX, was equal to ≈4.5% for the single-solution-based algorithms, whereas it did not exceed more than 1.59% for the developed augmented self-adaptive EA. Based on the numerical experiments, conducted over the large-size problem instances, the augmented self-adaptive parameter control strategy outperformed the parameter tuning strategy and the deterministic,

adaptive, and self-adaptive parameter control strategies by 16.41%, 15.03%, 12.50%, and 8.29%, respectively, in terms of the objective function values. Besides, the Augmented Self-Adaptive Evolutionary Algorithm (ASAEA) improved the objective function values, recorded for Tabu Search, Simulated Annealing, Variable Neighborhood Search, Ant Colony Optimization, and Particle Swarm Optimization, on average by 19.25%, 19.20%, 18.88%, 16.30%, and 16.08%, respectively, over the generated large-size problem instances. Moreover, the computational time, recorded for the proposed ASAEA did not exceed 3.73 min, which will allow the marine container terminal operators to develop and revise berthing plans fairly quickly.

The statistical superiority of the proposed ASAEA against other developed algorithms was evaluated by conducting a set of paired z-tests for the considered large-size problem instances. The results showed that the objective function values, which were returned by ASAEA, were statistically different as compared to all the alternative metaheuristic-based algorithms considered, except SAEA. However, the augmented self-adaptive parameter control strategy outperformed the self-adaptive parameter control strategy on average by 8.29% over the large-size problem instances (which can be viewed as significant). The coefficient of variation in the objective function values was calculated in order to assess the reliability of the algorithms, developed in this chapter of the dissertation. The results demonstrated that the maximum coefficient of variation did not exceed 1.4% over all the developed algorithms, which can be considered as acceptable from the practical standpoint. Note that the coefficient of variation was recorded to be equal to 0.5% for the ASAEA. To summarize, the developed ASAEA algorithm was proved to be a promising solution methodology, considering the results that were provided by the conducted numerical experiments, which can be helpful in terms of berth scheduling for the MCT operators.

CHAPTER 7

A UNIVERSAL ISLAND EA-PSO-EDA-DE METAHEURISTIC FOR THE BERTH SCHEDULING PROBLEM

This chapter of the dissertation provides a solution methodology, which relies on the island-based concept to solve a berth scheduling problem (BSP). As it is illustrated in Figure 40, a discrete berthing layout is adopted to serve a set of arriving vessels ($V = \{1, \dots, m\}$) at the available berthing positions ($B = \{1, \dots, n\}$) at the MCT. The vessel arrival time pattern is dynamic, and the scope of this chapter of the dissertation does not cover any uncertainties in the vessel arrival times or handling times because of unpredictable issues (e.g., inclement weather conditions, vessel technical issues, human errors, etc.). Once a given vessel arrives at the MCT, a certain number of push boats will tow the vessel to the pre-assigned berthing position, and if the berthing position cannot serve the vessel at the moment, the vessel will be towed to the considered waiting area at the MCT (upper left corner in Figure 40).

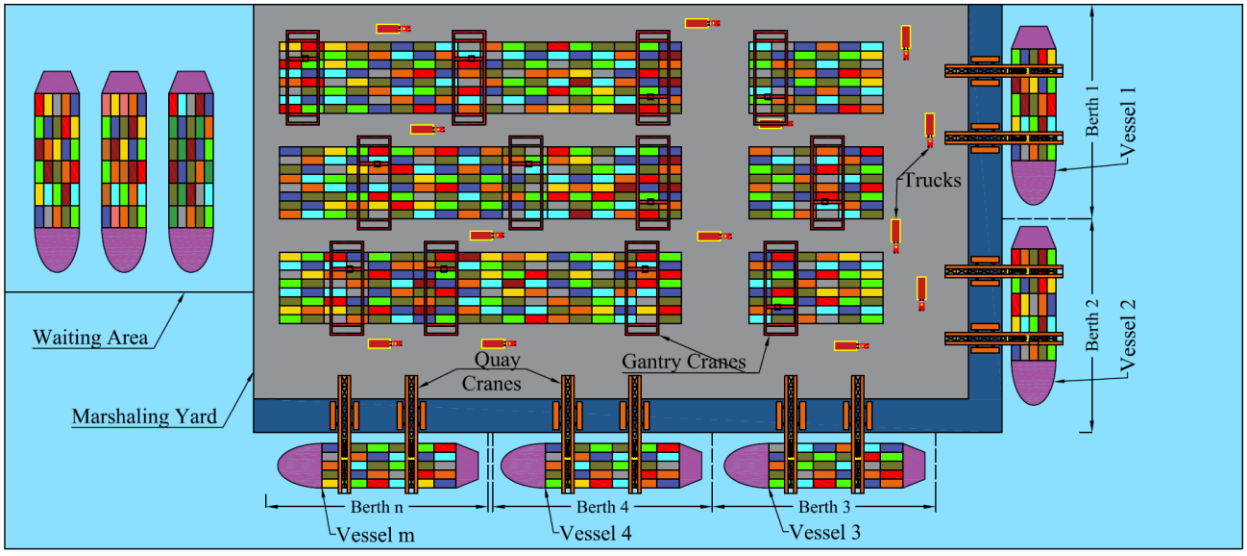

Figure 40 The general overview of the MCT.

A preferred berthing position is assigned to each arriving vessel, where the vessel receives the highest handling rate (i.e., the vessel is served in the shortest possible handling time).

Moreover, the spatial requirements are controlled in the assignment of vessels to the berthing positions in this chapter of the dissertation. In particular, the length and depth of the berthing position should be greater than the length and the draft of the vessel considering a horizontal clearance and a vertical clearance, respectively. A target departure time is imposed for all the arriving vessels, which is determined in the contracts signed between the MCT operators and the liner shipping companies. Note that the MCT operator will incur a penalty, if the vessel leaves the MCT after the negotiated departure time.

A metaheuristic algorithm, which relies on the island-based concept, is developed to solve the BSP in this chapter of the dissertation. The population is divided to four sub-populations or islands in the island-based metaheuristic algorithm, and each sub-population is covered with a population-based metaheuristic algorithm commonly used for the scheduling problems. Performance of the island-based metaheuristic algorithm is evaluated against four population-based and three single-solution-based metaheuristic algorithms.

7.1. Mathematical Structure of Optimization Model

The mathematical model, developed for the Spatially Constrained Berth Scheduling Problem (SCBSP), is comprehensively described in this section of the dissertation. Table 22 presents the nomenclature adopted in the mixed-integer linear mathematical formulation, developed for the proposed optimization model.

The objective function of SCBSP is formulated in equation (43). In this equation, the total waiting cost, the total handling cost, and the total late departure cost of the arriving vessels at the MCT are minimized.

$$min\, \boldsymbol{Z} = [\sum_{v\in V}(\varphi_v^{wt}\boldsymbol{\theta}_v^{wt}) + \sum_{v\in V}\sum_{b\in B}\sum_{t\in T}\left(\varphi_v^{ht}\theta_{vb}^{ht}\boldsymbol{x}_{vbt}\right) + \sum_{v\in V}(\varphi_v^{lt}\boldsymbol{\theta}_v^{lt})] \qquad (43)$$

A group of constraints were defined in SCBSP to model the real-world operational features of the problem studied herein. Constraint set (44) ensures that each vessel will be assigned to a berthing position just once during the considered planning horizon.

$$\sum_{b\in B}\sum_{t\in T}\boldsymbol{x}_{vbt} = 1\ \forall v \in V \quad (44)$$

Table 22 Nomenclature adopted for the mathematical model.

Model Component		Description
Type	**Nomenclature**	
Sets	$V = \{1, \dots, m\}$	set of the arriving vessels at the MCT (vessels)
	$B = \{1, \dots, n\}$	set of the available berthing positions at the MCT (berthing positions)
	$T = \{1, \dots, p\}$	set of discrete time periods (time 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 time period t (=0 otherwise)
Auxiliary Variables	$\boldsymbol{\theta}_{vbt}^{it}, v \in V, b \in B, t \in T$	the number of idling time periods for berthing position b between service of vessel v and its immediate predecessor served during time period $(t-1)$ (time periods)
	$\boldsymbol{q}_{vbt}, v \in V, b \in B, t \in T$	= 1 if vessel v leaves berthing position b at time period t (= 0 otherwise)
	$\boldsymbol{\theta}_{v}^{st}, v \in V$	start service time period of vessel v (time period)
	$\boldsymbol{\theta}_{v}^{ft}, v \in V$	finish service time period of vessel v (time period)
	$\boldsymbol{\theta}_{v}^{wt}, v \in V$	the number of time periods that vessel v should wait for service (time periods)
	$\boldsymbol{\theta}_{v}^{lt}, v \in V$	the number of time periods that vessel v leaves later than the negotiated departure time period (time periods)
Parameters	m	the total number of the arriving vessels at the MCT to be served (vessels)
	n	the total number of the available berthing positions at the MCT (berthing positions)
	p	the total number of time periods in the planning horizon (time periods)
	$\theta_{v}^{a}, v \in V$	the time period when vessel v arrives at the MCT (time period)
	$\theta_{b}^{ber}, b \in B$	the time period when berthing position b is available for the first time in the considered planning horizon (time period)
	$\widetilde{\theta_{v}^{ht}}, v \in V, b \in B$	the required time periods for handling vessel v at its preferred berthing position (time periods)
	$\theta_{vb}^{ht}, v \in V, b \in B$	the required time periods for handling vessel v at assigned berthing position b (time periods)
	$\theta_{v}^{d}, v \in V$	the negotiated departure time period for vessel v (time period)
	$D_{v}^{ves}, v \in V$	draft of vessel v (feet)
	$D_{b}^{ber}, b \in B$	depth of berthing position b (feet)
	$L_{v}^{ves}, v \in V$	length of vessel v (feet)
	$L_{b}^{ber}, b \in B$	length of berthing position b (feet)
	$P_{v}^{ver}, v \in V$	minimum vertical clearance for vessel v (feet)
	$P_{v}^{hor}, v \in V$	minimum horizontal clearance for vessel v (feet)
	$\varphi_{v}^{wt}, v \in V$	unit waiting cost component for vessel v (U.S. $/time period)
	$\varphi_{v}^{ht}, v \in V$	unit handling cost component for vessel v (U.S. $/time period)
	$\varphi_{v}^{lt}, v \in V$	unit late departure cost component for vessel v (U.S. $/time period)
	Υ	large positive number

Constraint set (45) avoids assignment of more than one vessel to the same berthing position at the same time period.

$$\sum_{v \in V} \boldsymbol{x}_{vbt} \leq 1 \; \forall b \in B, t \in T \tag{45}$$

Constraint set (46) ensures that the start service time period for a vessel should be greater than the time period when the assigned berthing position is available for service.

$$\sum_{b \in B} \sum_{t \in T} (\theta_b^{ber} \boldsymbol{x}_{vbt}) \leq \boldsymbol{\theta}_v^{st} \; \forall v \in V \tag{46}$$

Constraint set (47) ensures to set the start service time period for a vessel after its arrival at the MCT.

$$\theta_b^{ber} \boldsymbol{x}_{vbt} + \sum_{v' \in V: v' \neq v} \sum_{t' \in T: t' < t} \left(\theta_{v'b}^{ht} \boldsymbol{x}_{v'bt'} + \boldsymbol{\theta}_{v'bt'}^{it} \right) + \boldsymbol{\theta}_{vbt}^{it} \geq \theta_v^a \boldsymbol{x}_{vbt} \; \forall v \in V, b \in B, t \in T \tag{47}$$

Constraint set (48) guarantees that if a vessel is assigned to start its service at the berthing position at time period $\boldsymbol{\theta}_v^{st}$, the vessel must leave the berthing position at time period $\boldsymbol{\theta}_v^{st} + \theta_{vb}^{ht}$.

$$\boldsymbol{x}_{vbt} = \boldsymbol{q}_{vb(\boldsymbol{\theta}_v^{st} + \theta_{vb}^{ht})} \; \forall v \in V, b \in B, t \in T \tag{48}$$

The horizontal and vertical spatial requirements are checked by constraint sets (49) and (50), respectively. Specifically, the length and depth of the berthing position should be greater than the length and the draft of the vessel considering the horizontal and the vertical clearance due to safety issues, respectively.

$$(L_v^{ves} + P_v^{hor}) \boldsymbol{x}_{vbt} \leq L_b^{ber} \; \forall v \in V, b \in B, t \in T \tag{49}$$

$$(D_v^{ves} + P_v^{ver}) \boldsymbol{x}_{vbt} \leq D_b^{ber} \; \forall v \in V, b \in B, t \in T \tag{50}$$

Constraint set (51) estimates the start service time period for each vessel served at the MCT.

$$\boldsymbol{\theta}_v^{st} \geq \theta_b^{ber}\boldsymbol{x}_{vbt} + \sum_{v' \in V: v' \neq v} \sum_{t' \in T: t' < t} \left(\theta_{v'b}^{ht}\boldsymbol{x}_{v'bt'} + \boldsymbol{\theta}_{v'bt'}^{it}\right) + \boldsymbol{\theta}_{vbt}^{it} - \Upsilon(1 - \boldsymbol{x}_{vbt})\ \forall v \in V, b \in B, t \in T \quad (51)$$

Constraint set (52) computes the number of waiting time periods for each vessel served at the MCT.

$$\boldsymbol{\theta}_v^{wt} \geq \boldsymbol{\theta}_v^{st} - \theta_v^a\ \forall\ v \in V \quad (52)$$

Constraint set (53) estimates the finish service time period for each arriving vessel served at the MCT.

$$\boldsymbol{\theta}_v^{ft} = \boldsymbol{\theta}_v^{st} + \sum_{b \in B} \sum_{t \in T} (\theta_{vb}^{ht}\boldsymbol{x}_{vbt})\ \forall v \in V \quad (53)$$

Constraint set (54) calculates the late departure time periods for each vessel served at the MCT.

$$\boldsymbol{\theta}_v^{ld} = \boldsymbol{\theta}_v^{ft} - \theta_v^d\ \forall v \in V \quad (54)$$

The nature of all the decision variables, auxiliary variables, and parameters of the SCBSP mathematical model are defined in constraint sets (55) to (57).

$$\boldsymbol{x}_{vbt}, \boldsymbol{q}_{vbt} \in \mathbb{B}\ \forall v \in V, b \in B, t \in T \quad (55)$$

$$\boldsymbol{\theta}_{vbt}^{it}, \boldsymbol{\theta}_v^{st}, \boldsymbol{\theta}_v^{ft}, \boldsymbol{\theta}_v^{wt}, \boldsymbol{\theta}_v^{lt}, m, n, p, \theta_v^a, \theta_b^{ber}, \theta_{vb}^{ht}, \widetilde{\theta_v^{ht}}, \theta_v^d\ \in \mathbb{N}\ \forall v \in V, b \in B \quad (56)$$

$$D_v^{ves}, D_v^{ber}, L_v^{ves}, L_v^{ber}, P_v^{ver}, P_v^{hor}, \varphi_v^{wt}, \varphi_v^{ht}, \varphi_v^{lt}, \Upsilon \in \mathbb{R}^+\ \forall v \in V \quad (57)$$

7.2. Solution Strategy

An island-based metaheuristic algorithm, called a Universal Island-based Metaheuristic Algorithm (UIMA), is developed in this chapter of the dissertation to solve the proposed SCBSP mathematical model. Based on this concept (i.e., island-based metaheuristic), the population is divided into several sub-populations (i.e., islands) and a separate metaheuristic algorithm is executed on each one of the islands (Eiben and Smith, 2003). Throughout the search process, the islands can be kept isolated, or a specific number of the solutions can be exchanged among the islands (Eiben and Smith, 2003; Meng et al., 2017). The latter approach was adopted in this dissertation. Specifically, a certain number of solutions from each island were selected (called immigrants) to migrate to the other islands throughout the search process. There are four important parameters/procedures during the immigration process in UIMA including: (1) frequency of the migration; (2) number of immigrants; (3) selection of the migrating individuals; and (4) removal of individuals, which should be determined properly. Figure 41 illustrates the general process of UIMA (Eiben and Smith, 2003).

Based on Figure 41, the data, required for the implementation of UIMA, are provided in the first step. Then, two strategies are adopted to generate the initial population: (1) a First Come First Served with Spatial Requirements (FCFS-SR) heuristic is considered to generate half of the whole population; and (2) the next half of the population is generated randomly (for more details, refer to Dulebenets et al., 2018b). The whole population is checked by a repairing operator to make necessary changes in the solutions in case of infeasibility (more details are provided in section 7.2.1.1). Then, the generated initial population is divided into four sub-populations, where the half of each sub-population is chosen from the solutions generated using FCFS-SR, and the randomly generated solutions are selected for the next half of each sub-population. Each sub-population is assigned a metaheuristic algorithm randomly. In the next step, all the four metaheuristics start executing in parallel. Then, the defined migration criterion is checked and in case of satisfaction, a specific number of solutions, selected from each island, are exchanged between all the possible pairs of the islands (i.e., sub-populations of the metaheuristics). Then, the stopping criterion is checked, and if it is met, the algorithm returns the best solution discovered by the adopted algorithms; otherwise, the four developed algorithms start searching on their specific islands for another iteration.

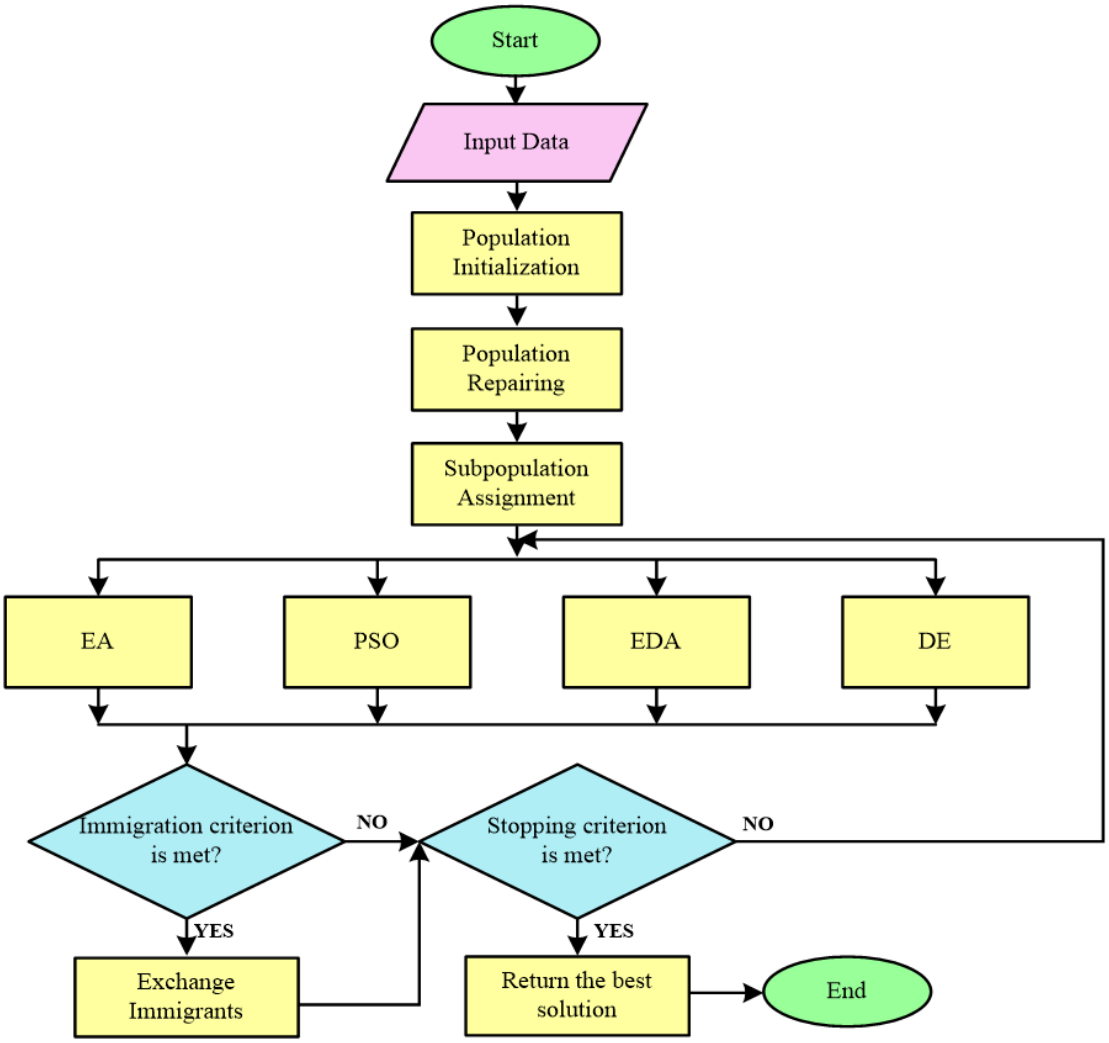

Figure 41 Basic UIMA idea.

A population in UIMA includes a deterministic number of solutions. A two-dimensional solution (as it consists of two rows), adopted in this chapter of the dissertation, is illustrated in Figure 42. The row at the top includes the indices of vessels, and the lower row accommodates the berthing positions that the vessels at the row above are assigned to. For example, vessels 4 and 6 are assigned to the berthing positions 2 and 1, respectively.

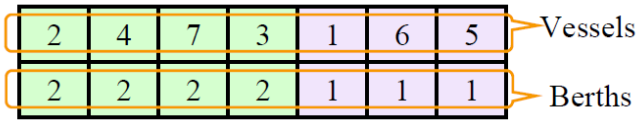

Figure 42 The solution skeleton.

7.2.1. The topology of UIMA

As mentioned earlier, four different population-based metaheuristic algorithms were adopted to be executed on the set of islands in this chapter of the dissertation, which include the following: (1) Evolutionary Algorithm (EA); (2) Particle Swarm Optimization (PSO); (3) Estimation of Distribution Algorithm (EDA); and (4) Differential Evolution (DE). Figure 43 presents a schematic illustration of the proposed UIMA. Each one of the islands covers a specific area in the search space, and a given algorithm keeps exploring and exploiting that area throughout the search process. All the islands are covered by the same type of algorithms in traditional island-based algorithms (Grefenstette, 1981; Alba and Tomassini, 2002), while, in UIMA, each island is assigned a specific population-based metaheuristic algorithm. The latter approach makes a unique exploration and exploitation of each island, which results in a diverse population, and also decreases the probability of the premature convergence (i.e., convergence to a local optimum) (Ammi and Chikhi, 2015; Magalhaes et al., 2015).

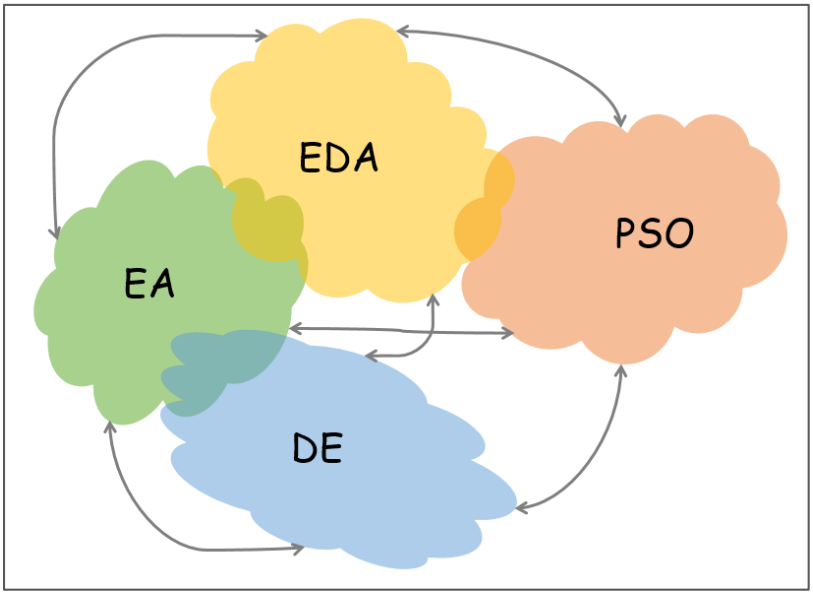

Figure 43 The schematic illustration of the proposed UIMA.

Figure 43 illustrates a different color and shape for each search area (i.e., island), allocated for each metaheuristic algorithm. The latter means that each metaheuristic algorithm relies on a unique search methodology, which leads to a discovery of diverse solutions in the search space; however, the islands are not fully isolated. Specifically, each metaheuristic algorithm plays a specific role throughout the search process based on its exclusive features. The EA and DE algorithms are evolutionary-based search tools, which work based on the survival of the fittest and are inspired by the process of natural selection. EA and DE select the solutions utilizing some specific selection operators. The selected solutions participate in the mating process and generation of new solutions. The EA and DE algorithms utilize the crossover and mutation operators for the exploration and exploitation of the search space (i.e., generation of the new solutions), respectively (Storn and Price, 1997; Eiben and Smith, 2003). EA and DE are controllable search tools in the UIMA algorithm, as they can discover new areas in the search space by adopting appropriate values for the crossover and mutation probabilities. Liang et al. (2009), Correcher and Valdez (2017), Dulebenets et al. (2018b) and many other researchers adopted an EA to solve a BSP, whereas DE was deployed by Şahin and Kuvvetli (2016), and Liu et al. (2016) to solve a BSP.

The PSO algorithm is recognized as a swarm-intelligence algorithm, which is a fast, local search operator (Kennedy and Eberhart, 1997; Bonabeau et al., 1999). PSO was inspired by flocking behavior of birds or schooling behavior of fish (Arabani et al., 2011; Ting et al., 2014). The population in PSO is composed of a group of particles. Each particle has a position and velocity, which give an opportunity to the particle to move across the search space and discover the new areas. The best position discovered by all the particles (representing the solution with the highest quality) is updated in each iteration. Moreover, the position and the velocity of each particle are updated in each iteration towards the convergence of the algorithm. Wang et al. (2012) and Ting et al. (2014) adopted a PSO to solve BSPs. The EDA algorithm utilizes the statistical tools throughout the search process, whereas the EA, PSO, and DE algorithms explore and exploit the search space stochastically (Larrañaga and Lozano, 2001; Izquierdo et al., 2012). EDA has been widely used for the scheduling problems in different fields of the literature (Izquierdo et al., 2012; Hao et al., 2013; Chen and Chen, 2013; Wang et al., 2018). In Figure 43, the double arrows simulate the exchanging immigrants among islands. The adopted metaheuristic algorithms are comprehensively described in sections 7.2.1.1 to 7.2.1.4.

7.2.1.1.Evolutionary Algorithm

The Evolutionary Algorithms (EAs) are naturally inspired and population-based metaheuristic algorithms, which work based on the survival of the fittest individual (a chromosome or an individual are two other names for a solution, which are used interchangeably in this dissertation) (Boussaïd et al., 2013). In EA, a group of solutions, which is called the population, can be generated randomly or considering a problem-specific approach. In order to evaluate the quality of each solution, the fitness value is calculated for all the generated solutions, based on the objective function defined for the problem. Considering the fitness values of individuals, some of them are selected as parents to mate together. The crossover and mutation, as two algorithmic operators, are adopted to generate the new offspring using the selected parents. The generated offspring are evaluated and a group of them are selected for the next generation (Boussaïd et al., 2013; Dulebenets et al., 2018b; Ghoumari et al., 2018). The main steps of EA are presented in Algorithm 10 (Bozorg-Haddad et al., 2017).

Algorithm 10: 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 attended the 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 chromosomes

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

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

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

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

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

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

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

9: **return** $EASubPop$

In general, the EA algorithm has an iterative loop, which checks the stopping criterion, and also EA returns the best solution discovered at the termination of the algorithm. However, the EA algorithm herein is operated as a search tool within UIMA; hence, the stopping criterion is controlled by the UIMA algorithm (see Figure 41). Due to the latter fact, there is no iterative loop

in Algorithm 10, and also UIMA provides the best solution discovered by one of the algorithms (i.e., EA, PSO, EDA, and DE) throughout the search process. The explanation above (regarding the iterative loop and returning the best solution at convergence) is applicable to the PSO, EDA, and DE algorithms as well (Algorithms 11, 12, and 13 in sections 7.2.1.2, 7.2.1.3, and 7.2.1.4).

Step 0 of Algorithm 10 includes the initialization of the data structure, required for storing fitness values. In step 1, the fitness values of chromosomes in the assigned sub-population are estimated (using equation (43) in section 7.1). In step 2, a parent selection operator (***SUS***) is applied to the sub-population to select a specific number of chromosomes (equal to the sub-population size) as parents. In order to produce the offspring, the crossover and mutation, as two main algorithmic operators, are applied to the parents selected in steps 3 and 4, respectively. In step 5, the newly generated offspring are investigated to be repaired in case of infeasibility. The fitness value of each offspring is calculated in step 6. Another selection operator is applied to the offspring to establish a new sub-population for the next generation (step 7). The elitism strategy implements the necessary changes to the sub-population in step 8. The newly discovered chromosomes are returned in step 9.

In step 2 of Algorithm 10, a Stochastic Universal Sampling (SUS) selection operator was adopted to select the parents. The SUS operator selects the chromosomes in a way, which gives the chromosomes with lower fitness values some chances to be selected as parents (Eiben and Smith, 2003). The operation of SUS is illustrated in Figure 44.

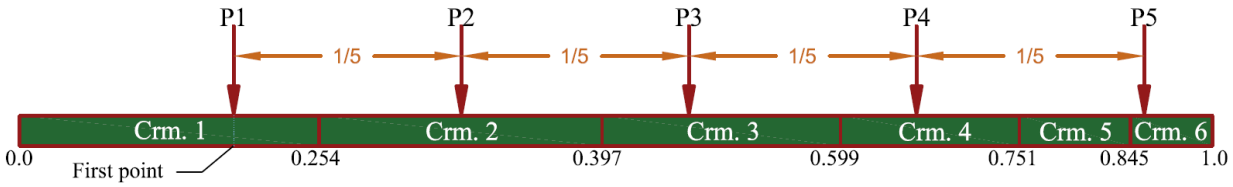

Figure 44 The stochastic universal sampling selection strategy.

Based on Figure 44, SUS aims to choose 5 chromosomes (i.e., $N^{SUS} = 5$) out of 6 randomly selected chromosomes. This relationship: $\boldsymbol{P}_i^{chr} = \frac{\boldsymbol{z}_i}{\sum_{i \in I} \boldsymbol{z}_i}$ was developed to estimate the

selection probability of each solution (i.e., the length assigned to the solution in the range [0,1]); where $\boldsymbol{P}_i^{chr}$ is the portion of the range [0,1] assigned to chromosome i (among the set of randomly selected chromosomes, i.e. I) with the fitness value $\boldsymbol{Z}_i$. The first point shown in Figure 44 is randomly chosen in the range $[0,1/5]$. Then, five pointers, which are located equal to $1/5$ far apart of each other, are placed on the drawn axes. Each chromosome, which has a pointer in its range, is selected as a parent (i.e., chromosomes 1, 2, 3, 4, and 6 in Figure 44). Note that, due to the specific selection strategy of SUS, chromosome 6 is selected as a parent, although it has a lower fitness value as compared to chromosome 5. In step 3, the order crossover is implemented using the parents selected by SUS. In the developed optimization model, each one of the vessels should be served just once, which means that a given vessel should not appear more than once in the chromosome. Hence, the order crossover is selected, as it prevents the generation of repetitive genes in the row of the vessels (i.e., the upper row of chromosomes).

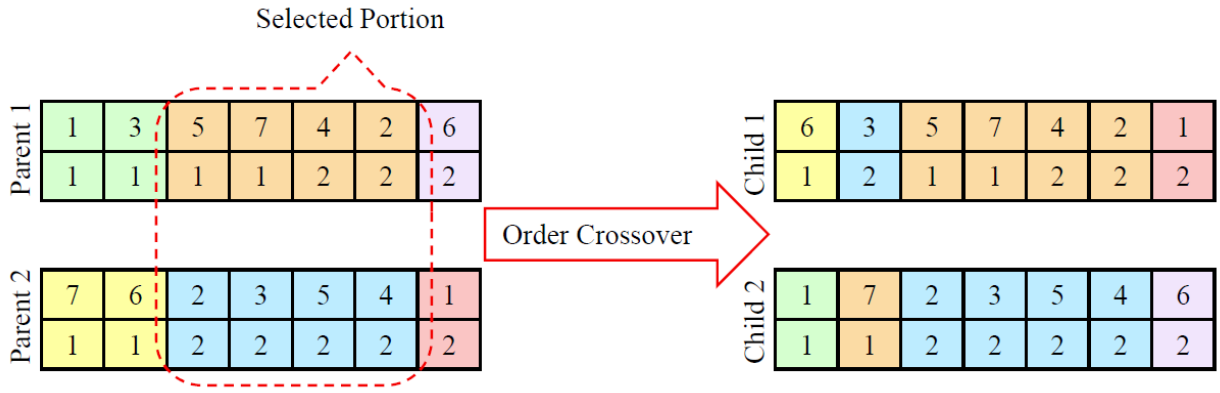

Figure 45 Application of the order crossover on two parents.

As it is shown in Figure 45, the same portion (in terms of location) of parents is selected randomly. The selected portion of Parent 1 is copied into Child 1, then, those vessels, which are not in the copied portion, are picked up from Parent 2 in a specific order. This order starts after the selected portion, followed towards the end of Parent 2 and continues from the beginning of Parent 2 from left to the right. In order to generate Child 2, the same process must be implemented (Eiben and Smith, 2003). A swap mutation is adopted in step 4 of Algorithm 10, which is applied to the rows of vessels and berths, separately. Swap mutation relies on random selection of two genes, and then exchanges their alleles between each other (Figure 46) (Eiben and Smith, 2003).

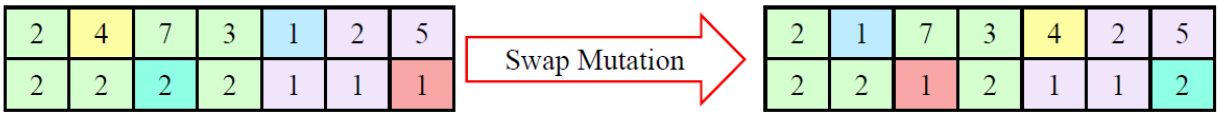

Figure 46 Application of the swap mutation on an offspring chromosome.

All the generated offspring are checked in step 5, and they will be repaired in case of infeasibility. The spatial requirements violation and the disruption of the vessel service order in a given solution are the two factors, which independently make a solution an infeasible one. Figure 47 clarifies how a service order disruption can create an infeasible solution.

Figure 47 The process of repairing an infeasible solution.

According to Figure 47, vessels 5 and 1 are placed among the vessels 2, 7, and 3, which are assigned to berthing position 2. The latter disrupts the vessel service order at berthing positions 1 and 2. After repairing the infeasible solution, none of the vessels interrupts the vessel service order at any other berthing position.

In step 7, the sub-population for the next generation is established utilizing a binary tournament selection. Specifically, two chromosomes are randomly selected from the sub-population and the one, which has a better quality (lower objective function value in this case), will be selected for the next generation. The latter process is iterated as long as the sufficient number of chromosomes (equal to the considered sub-population size for the algorithm) are selected for the next generation (Eiben and Smith, 2003). In order to prevent missing the best solution due to application of algorithmic operators throughout the search process, the elitism strategy is applied within the EA algorithm in step 8. Specifically, the elitism strategy saves the best solution from the previous generation and adds it to the newly discovered sub-population (Eiben and Smith, 2003).

7.2.1.2.Particle Swarm Optimization

The Particle Swarm Optimization (PSO), introduced by Kennedy and Eberhart (1997), is classified as a population-based metaheuristic algorithm. This algorithm was inspired by the flocking behavior of birds or schooling behavior of fish. In detail, when a given particle in a group of particles (a bird in a flock) is moving, its move is remarkably affected by the distance between that particle and the other ones. Note that the aforementioned distances are affected by the position and velocity of a given particle. In order to adjust the position of each particle, two factors are defined in PSO: (1) $pBest$ – the best position that a particle has been experienced so far throughout the search process; and (2) $gBest$ – the best position achieved so far by all the particles. $pBest$ and $gBest$ are updated throughout the search process. The main steps of PSO are presented in Algorithm 11 (Ting et al., 2014).

Algorithm 11: 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 particle
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** $PSOSubPop$

In step 0 of Algorithm 11, the necessary data structure for storing fitness values is initialized. In step 1, the solutions with integer-coded representation, adopted within UIMA, are mapped to the solutions with real-coded representation, which will be further used by PSO. The fitness value of each particle in the assigned sub-population is calculated in step 2. Two key

parameters (i.e., $pBest$ and $gBest$) are recorded in steps 3 and 4. The current velocity and position are retrieved in step 5 for each particle. A group of new particles are generated by updating the velocity and position of all the particles in steps 6 and 7, respectively. The PSO sub-population is updated in step 8. The newly generated particles are checked by ***Repair*** to be fixed in case of any infeasibility (step 9). The fitness values of new particles are calculated in step 10. The generated sub-population is mapped to the integer-coded representation, adopted within UIMA (see Figure 42) in step 11, and the algorithm returns the particles ($PSOSubPop$) discovered in step 12.

The velocity and position of each particle are updated (steps 5 and 6) utilizing two equations below (Arabani et al., 2011; Ting et al, 2014):

$$V_p^{new} = WV_p^{old} + C_1R_1\left(pBest_p - P_p^{old}\right) + C_2R_2\left(gBest - P_p^{old}\right) \quad (58)$$

$$P_p^{new} = P_p^{old} + V_p^{new} \quad (59)$$

where, V_p^{new} and V_p^{old} are the new and current velocities of particle p, respectively. P_p^{old} is the current position and P_p^{new} is the new position of particle p. W is the inertia weight, and C_1 and C_2 are the cognition and social components, respectively. In detail, the larger the inertia weight (W), the higher the exploration capability of the algorithm and the lower the chance of getting stuck in the local optima. Whereas, a smaller W would help the algorithm to exploit a discovered search area (Arabani et al., 2011). It is suggested by Eberhart and Shi (2001) to set $W = 0.5 + rand/2$ ($rand$ is a random number in the range [0,1]) and $C_1 = C_2 = 2$. Nonetheless, a parameter tuning was conducted to set the values of W, C_1, and C_2. R_1 and R_2 are two random numbers in the range [0,1]. $pBest_p$ is the current $pBest$ of particle p, where $gBest$ is the best position found so far by all the particles. Finally, equation (59) is utilized to estimate the new position of particle p (i.e., P_p^{new}).

The skeleton of a given particle in PSO is illustrated in Figure 48. The lower row in Figure 48 includes the specific format of encoding for berthing positions in a given particle, where similar to the general solution skeleton shown in Figure 42, the upper row accommodates the indices of arriving vessels. In Figure 48, there are 7 arriving vessels, which should be served in two berthing

positions. A group of real numbers are generated in the range [0,2] to encode the berthing positions in the lower row. In detail, the integer part of the real numbers in the lower row stands for the berthing position that the vessel in the upper row is assigned to, where the fractional part determines the service order of vessels at the berthing position. For instance, vessels 1, 6, and 5 should be served at the berthing position 1. The sorted real numbers correspond to the berthing positions in an ascending order, where vessel 6 should be moored first, followed by vessel 1, and then vessel 5 (Ting et al., 2014).

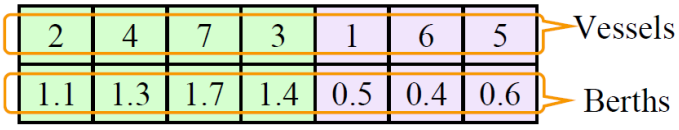

Figure 48 Particle representation in PSO.

7.2.1.3. Estimation of Distribution Algorithm

The Estimation of Distribution Algorithm (EDA) is a population-based metaheuristic algorithm developed by Larrañaga and Lozano (2001). As compared to the other three adopted algorithms in this chapter of the dissertation (i.e., EA, PSO, and DE), EDA utilizes statistical tools (i.e., a probability distribution) to generate new solutions, whereas EA, PSO, and DE generate new solutions stochastically. A group of promising solutions in the population are considered to collect the statistical information (Izquierdo et al., 2012). EDA establishes a probability matrix, which includes the probability of assigning a vessel to each one of the berthing positions in the developed SCBSP. Algorithm 12 presents the main steps of EDA (Izquierdo et al., 2012).

In step 0 of Algorithm 12, the data structure required for storing fitness values is initialized. In the canonical version of EDA, the probability set is generated, and then the initial population is sampled based on the probability set (Izquierdo et al., 2012). Nonetheless, the probability set is established according to the sub-population in this dissertation, due to the fact that the sub-population is already generated and assigned to EDA (i.e., $EDASubPop$). The fitness values of all the solutions in the assigned sub-population are estimated in step 1. The initial probability matrix is generated using the ***UpdatePrb*** function (step 2). In order to decrease the probability of premature convergence, the probability matrix, generated in the previous step, is shaken in step 3

(further explanation are provided later in this section). 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 sub-population is checked to repair any existing infeasible solutions in step 6. The fitness value of each solution is calculated in step 7. The new sub-population ($EDASubPop$), generated using algorithmic operators, is returned by Algorithm 12 in step 8.

Algorithm 12: 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** $EDASubPop$

The probability set is a matrix (equation [60]), where the number of its rows is equal to the number of arriving vessels (m), and the number of columns is equal to the number of berthing positions (n). In detail, $P_{mn}(i)$ is the probability of assigning vessel m to berthing position n in generation i. Therefore, a row comprises the probabilities of assigning a given vessel to different berthing positions, whose summation should be equal to 1 (Izquierdo et al., 2012).

$$P(i) = \begin{bmatrix} P_{11}(i) & \cdots & P_{1n}(i) \\ \vdots & \ddots & \vdots \\ P_{m1}(i) & \cdots & P_{mn}(i) \end{bmatrix} \tag{60}$$

The probability set is updated in each generation using the developed $\boldsymbol{UpdatePrb}$ function. First, best Υ $(= round(\psi \cdot popsize))$ solutions from the current sub-population are

selected. Then, considering the selected solutions, all the arrays of the probability set will be updated using equation (61):

$$P_{mn}(i) = \frac{Num_{mn}(i)}{\sum_{k \in B} Num_{mk}(i)} \quad (61)$$

where, $Num_{mn}(i)$ is the number of times vessel m has been scheduled for service at berthing position n among the solutions that were selected in generation i; and the denominator is the number of times vessel m has been scheduled for service at all the considered berthing positions in generation i.

In order to decrease the probability of the premature convergence, the shaking operator is implemented in EDA. Moreover, application of the shaking operator increases the population diversity, which makes a better exploration and exploitation in the search space accordingly. The shaking operator selects a specific number of vessels ($Num^{vs} < m$), then the corresponding rows 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 together randomly). In each generation of EDA, the elitism operator is implemented (step 4 of Algorithm 12) to generate a portion of the new sub-population. Specifically, the new sub-population is composed of the Num^{elt} $(= round(\psi \cdot popsize))$ number of best solutions from the previous generation. The remainder of the new sub-population is generated utilizing the $\boldsymbol{PrbExplore}$ function (step 5 in Algorithm 12). $\boldsymbol{PrbExplore}$ relies on the same strategy as the roulette wheel selection operator in EAs to generate a new solution (Izquierdo et al., 2012). The process of assigning a vessel to a berthing position by $\boldsymbol{PrbExplore}$ is illustrated in Figure 49. First of all, the corresponding row to a given vessel in the probability set is considered. In Figure 49, it is assumed that there are 7 berthing positions in the problem. Then, the berthing positions are sorted based on the probability of the assignment (in an ascending order). The probabilities are cumulatively summed up together. A random number is generated in the range [0,1] (e.g., 0.48). As the generated random number (i.e., 0.48) is less than 0.56 and greater than 0.32, the berthing position 1 is selected. The same approach is repeated for the other vessels as well.

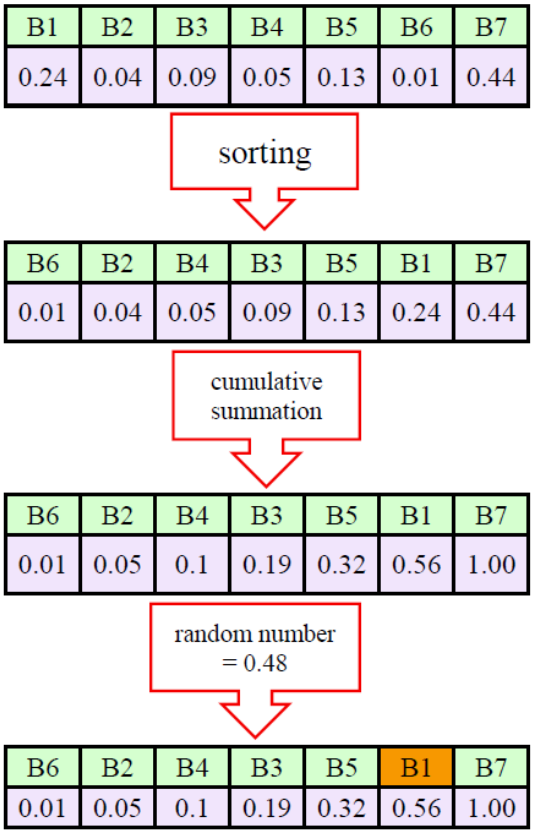

Figure 49 New solution generation by ***PrbExplore***.

7.2.1.4.Differential Evolution

The Differential Evolution (DE) algorithm is the last population-based metaheuristic algorithm adopted for UIMA, which was introduced by Storn and Price (1997) for the first time. As DE deploys the mutation and crossover operators to discover new areas in the search space, the procedure applied for the optimization by DE is very similar to EA. Specifically, DE is based on generating and manipulating vectors, which play the equivalent role of the chromosomes in EA. The main steps of DE are presented in Algorithm 4 (Arabani et al., 2011).

In step 0, the data structure, required for storing fitness values, is initialized. The generated solutions are called vectors in DE; nonetheless, they are exactly the same as what is presented in Figure 42. In step 1, the integer-coded vectors in sub-population are mapped to a new format named as $rpiPop$. In step 2, the fitness values of vectors ($FitVals$) are estimated. In step 3 and 4, the algorithmic operators (i.e., mutation and crossover operators) are applied to the sub-population,

respectively. In step 5, the newly generated vectors ($CrPop$) are investigated to be repaired in case of infeasibility. The vectors, checked and repaired in step 5, are evaluated (i.e., their fitness values are calculated) in step 6. In order to select a group of vectors for the next generation, a selection strategy is applied to the sub-population in step 7. The elitism strategy is implemented in DE in step 8. The sub-population, selected for the next generation ($rpiPop$), is mapped to an integer-coded sub-population ($DESubPop$). This mapping is conducted as some vectors may have to migrate to the other islands belong to the other three algorithms. Therefore, the migrating vectors should be in the same format with the solutions on the other islands (step 9). The newly found sub-population ($DESubPop$) is returned by DE in step 10. Further details are provided for some of the aforementioned steps later in this section.

Algorithm 13: Differential Evolution (DE)

$\boldsymbol{DE}(DESubPop, \alpha, C^{cr}, InData)$

in: $DESubPop$ – the assigned sub-population; α – the mutation coefficient; C^{cr} – the crossover probability; $InData$ – the SCBSP input data

out: $DESubPop$

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

1: $rpiPop \leftarrow \boldsymbol{RPIMap}(DESubPop)$ ◁ Maps the integer-coded sub-population to a real-coded one

2: $FitVals \leftarrow \boldsymbol{Fitness}(rpiPop, InData)$ ◁ Estimates the objective function values for all vectors

3: $MuPop \leftarrow \boldsymbol{Mutation}(rpiPop, \alpha)$ ◁ Applies mutation to the sub-population

4: $CrPop \leftarrow \boldsymbol{Crossover}(rpiPop, MuPop, C^{cr})$ ◁ Applies crossover to the sub-population

5: $CrPop \leftarrow \boldsymbol{PopRepair}(CrPop)$ ◁ Repairs the infeasible generated vectors

6: $FitCrPop \leftarrow \boldsymbol{Fitness}(CrPop, InData)$ ◁ Estimates the objective function values for all vectors

7: $rpiPop \leftarrow \boldsymbol{Selection}(FitVals, FitCrPop, rpiPop, CrPop)$ ◁ Selects the vectors for the next iteration

8: $rpiPop \leftarrow \boldsymbol{Elit}(rpiPop)$ ◁ Applies elitism strategy

9: $DESubPop \leftarrow \boldsymbol{POPMap}(rpiPop)$ ◁ Maps the real-coded sub-population into an integer-coded one

10: **return** $DESubPop$

The vectors in DE are in the format called Relative Position Indexing (RPI) (Lichtblau, 2002). The process of converting an integer-coded vector to an RPI-format one is illustrated in Figure 50.

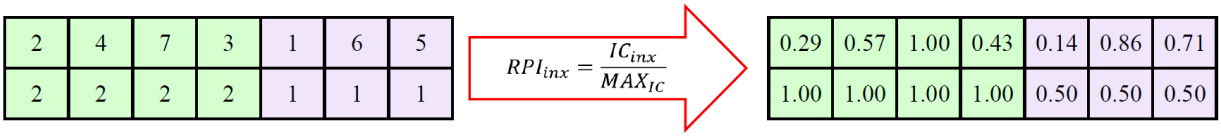

Figure 50 Mapping an integer coded vector to an RPI-format one.

As it is illustrated in Figure 50, an integer-coded vector is mapped to an RPI-format one utilizing the equation, shown in the arrow. In the latter equation, RPI_{inx} is the index of a vessel or a berthing position in the RPI-format, IC_{inx} is the integer-coded index of a vessel or a berthing position, and MAX_{IC} is the index with the maximum integer value of the corresponding vector (either the vessel vector or the berthing position vector). Note that the maximum value for each row of the integer-coded vector is selected independently (e.g., the maximum values for the rows of vessels and berths are 7 and 2, respectively).

One of the equations (62) to (65) is utilized to apply the mutation operator to the RPI-format vectors (step 3) (Arabani et al., 2011):

$$V_i^m = V_a^{org} + \alpha \cdot \left[V_b^{org} - V_c^{org}\right]; i \neq a, b, c \quad (62)$$

$$V_i^m = V_{best}^{org} + \alpha \cdot \left[V_a^{org} - V_b^{org}\right]; i \neq a, b \quad (63)$$

$$V_i^m = V_a^{org} + \alpha \cdot \left[V_b^{org} - V_c^{org} + V_d^{org} - V_e^{org}\right]; i \neq a, b, c, d, e \quad (64)$$

$$V_i^m = V_{best}^{org} + \alpha \cdot \left[V_a^{org} - V_b^{org} + V_c^{org} - V_d^{org}\right]; i \neq a, b, c, d \quad (65)$$

where, V_i^m is the i^{th} vector in the sub-population after implementing the mutation operator; V_a^{org}, V_b^{org}, V_c^{org}, V_d^{org}, V_e^{org} are five randomly selected vectors from the current (original) sub-population, which must differ from vector i; V_{best}^{org} is the best vector (considering the fitness value) in the current sub-population; and α is a coefficient randomly selected in the range [0,1). Note that all the equations (62) to (65) were encoded in the DE algorithm, and only one of them was selected randomly to apply the mutation operator to each vector in the DE sub-population.

After implementation of the mutation operator, the crossover operator is applied to the vectors using equation (66) (step 4):

$$V_i^{cr} = \begin{cases} V_i^m & if\ rand \le C_{cr} \\ V_i^{org} & otherwise \end{cases} \quad (66)$$

where V_i^{cr} is the i^{th} vector, which has been generated by the crossover operator; and C_{cr} is the constant coefficient of the crossover. In detail, equation (66) means that if a random number ($rand$) generated in the range [0,1) is smaller than or equal to C_{cr}, the vector will be set equal to the mutant vector; otherwise, the original vector will be returned by the crossover operator.

In order to establish the new sub-population for the next generation, the selection operator is applied to the current sub-population (step 7). The selection operator in DE is implemented based on equation (67):

$$V_i^{next} = \begin{cases} V_i^{cr} & if\ Fit(V_i^{cr}) \le Fit(V_i^{org}) \\ V_i^{org} & otherwise \end{cases} \quad (67)$$

where V_i^{next} is the i^{th} vector in the sub-population for the next generation. Based on equation (67), V_i^{cr} is selected for the next generation, if the fitness value of the vector after implementation of the crossover ($Fit(V_i^{cr})$) is smaller than the fitness value of the original vector ($Fit(V_i^{org})$) (as the developed model aims to minimize the objective function value); otherwise, the original vector (V_i^{org}) will be transferred to the next generation.

7.2.2. The strategic parameters of UIMA

There are several factors that can substantially affect the process of migration. Hence, the latter factors should be properly determined. The frequency of implementing the migration process is one of the most important factors (Alba and Tomassini, 2002). Furthermore, there are three other effective parameters/procedures, where a reasonable strategy should be developed for each one of them (Eiben and Smith, 2003): (1) number of immigrants; (2) selection of the individuals, which should migrate; and (3) selection of the individuals, which must be removed in the destination island to make enough room for the migrating individuals.

7.2.2.1. Frequency of migration

The frequency of migration can be conducted using the deterministic approach as a widely used approach. In the latter approach, the migration process is implemented based on a specific number of algorithmic iterations, and this process will be conducted recursively towards termination of the algorithmic search. Note that if the migration is implemented too frequently, the population diversity decreases. Whereas, if the migration occurs very few times, some sub-populations may converge to a local optimum, and all the conducted computational efforts in other sub-populations will not have any significant value. Besides, one of the major drawbacks with the deterministic strategy is that it does not consider any feedback from the search for implementation of the migration process (Eiben and Smith, 2003). Some studies adopted an adaptive strategy for determination of the migration frequency to address the two aforementioned drawbacks with the deterministic migration approach (Martin et al., 1997; Kurdi, 2015). An adaptive approach was developed in this chapter of the dissertation, where the migration process is conducted whenever the improvements in fitness values in two or more than two islands are less than $\lambda = 0.1\%$ (determined based on the conducted parameter tuning) as compared to the fitness values of the first iteration after the previous migration.

7.2.2.2. Number of immigrants

The number of immigrants (N^{Img}) is defined as the number of individuals that should migrate from one island to the other ones. A high number of immigrants may lead to a quick convergence to a similar solution on all the islands; hence, the number of immigrants should be properly considered. Therefore, Eiben and Smith (2003) recommended exchanging only a few solutions between the islands. The number of immigrants in this dissertation was set to $N^{Img} = 4$ individuals (according to the conducted parameter tuning, see section 7.3.2) at each migration process.

7.2.2.3.Selection of the migrating individuals

Different selection strategies can be adopted to select the migrating individuals. The immigrants could be selected randomly, or a selection operator could be adopted to implement this task (Eiben and Smith, 2003). A roulette wheel selection operator was adopted to select the migrating individuals in this chapter of the dissertation.

In the roulette wheel selection, a selection probability is assigned to each one of solutions in the sub-population (calculated using equation [68]), which has a reverse relationship (as the SCBSP mathematical model aims to minimize the objective function value) with its fitness value.

$$\boldsymbol{Pr}_i = \frac{1/\boldsymbol{Z}_i}{\sum_{j \in subpop} 1/\boldsymbol{Z}_j} \tag{68}$$

Where $\boldsymbol{Pr}_i$ and $\boldsymbol{Z}_i$ are the probability and the fitness value of individual i, respectively. The probabilities calculated using equation (68) are cumulatively summed up; therefore, each one of solutions is assigned a portion of the range [0,1]. A random number in the range [0,1] is generated, which corresponds to one of the candidate solutions. The latter solution (the one the selected random number corresponds to) is selected as an immigrant. The latter process should be iterated until a sufficient number of immigrants is selected (Eiben and Smith, 2003).

7.2.2.4. Removal of individuals

Similar to the immigrant selection, a strategy should be developed to select a specific number of individuals for removal in the destination island to make enough room for the newly migrating solutions. The solutions, which should be removed, can be selected randomly or based on a selection strategy (Eiben and Smith, 2003). If the solutions are randomly removed from the destination islands, some of the more promising solutions might be missed. To avoid the latter, the developed roulette wheel selection, described in section 7.2.2.3, was revised for this section of the dissertation. Equation (69) was developed as a revision of equation (68) to calculate the selection probability of solutions for removal.

$$\boldsymbol{Pr}_i = \frac{\boldsymbol{Z}_i}{\sum_{j \in subpop} \boldsymbol{Z}_j} \tag{69}$$

Note that, the application of equation (69) increases the selection probability of low-quality individuals for removal. The process of the selection of the solutions for removal is exactly the same as all the steps described for selecting migrating individuals in section 7.2.2.3, except for the application of probabilities calculated by equation (69).

7.3. Numerical Experiments

The comparative analysis, designed for evaluation of UIMA, is comprehensively described in this section of the dissertation, and the obtained results are presented. All the metaheuristic algorithms adopted for the islands (i.e., EA, PSO, EDA, and DE) were independently executed to evaluate performance of the developed UIMA algorithm against each one of them. Moreover, a group of single-solution-based metaheuristic algorithms were adopted to solve the developed SCBSP, including: (1) Tabu Search – TS; (2) Simulated Annealing – SA; and (3) Variable Neighborhood Search – VNS. CPLEX (embedded in GAMS environment) was used to solve the SCBSP mathematical model to the global optimality. The quality of the solutions, obtained by UIMA and other adopted algorithms (i.e., EA, PSO, EDA, DE, TS, SA, and VNS), was evaluated against the global optima generated by CPLEX.

The MATLAB 2016a environment was adopted to encode all the metaheuristic algorithms (Mathworks, 2019), and the numerical experiments were conducted utilizing an Alienware CPU with an Intel® Core™ i7-7700K processor, 32 GB of RAM, and Windows 10 Operating System.

7.3.1. Input data description

Table 23 presents how the values of parameters adopted in the developed SCBSP mathematical model were generated. Appropriate values for the parameters in Table 23 were adopted from several studies published in the BSP literature and online resources: Nishimura et al. (2001); Imai et al. (2008); Cheong et al. (2010); Lu et al. (2011); Imai et al. (2013); Rodriguez-Molins et al. (2014); Hu (2015); Dulebenets (2016); Dulebenets (2018a-g); Dulebenets et al. (2018a,b); Dulebenets et al. (2019a-c); and Eniram (2019).

The uniform generation of a random real number between two values $Val1$ and $Val2$ will be presented using the following notation: $U[Val1; Val2]$. The average inter-arrival time ($\Delta\theta$) defines the arrival pattern of vessels at the MCT. Based on the commonly used methodology in the BSP literature, the arrival pattern of vessels follows an exponential distribution with an average inter-arrival time of 2 time periods (Imai et al., 2008; Dulebenets et al., 2018a,b). Note that each time period was assumed to be equal to 1 hour. The adopted notation $abs(Val3)$ rounds $Val3$ to

the nearest integer value to $Val3$, where $abs^{+}(Val4)$ means to round $Val4$ to the nearest integer value greater than or equal to $Val4$.

Table 23 Data generation for the SCBSP mathematical model.

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 time periods: p (time periods)	Values are changed based on the problem instance
Arrival time of vessel v at the MCT: $\theta_v^a, v \in V$ (time periods)	$\theta_{v+1}^a = abs^{+}(\theta_v^a + \Delta\theta)\ \forall v \in V$
Average inter-arrival time of vessels: $\Delta\theta$ (time periods)	$\Delta\theta = \exp[2]$
First time availability of berthing position b in the planning horizon: $\theta_b^{ber}, b \in B$ (time period)	$\theta_b^{ber} = 0\ \forall b \in B$
The required time periods for handling vessel v at preferred berthing position b: $\widetilde{\theta_{vb}^{ht}}, v \in V, b \in B$ (time periods)	$\widetilde{\theta_{vb}^{ht}} = abs(U[10; 24])$
The required time periods for handling vessel v at planned berthing position b: $\theta_{vb}^{ht}, v \in V, b \in B$ (time periods)	$\theta_{vb}^{ht} = abs^{+}(\widetilde{\theta_{vb}^{ht}} \cdot \left(1 + 0.03 \cdot abs\left(B_v^{pr} - B_v^{as}\right)\right))$
Negotiated departure time period for vessel v: $\theta_v^d, v \in V$ (time periods)	$\theta_v^d = abs(\theta_v^a + \widetilde{\theta_{vb}^{ht}} \cdot U[1.20; 1.50])\ \forall v \in V, b \in B$
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)	$D_b^{ber} = 1.20 \cdot min_v(D_v^{ves}) + U[0; 1.20 \cdot max_v(D_v^{ves}) - min_v(D_v^{ves})]\ \forall b \in B$
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)	$L_b^{ber} = 1.20 \cdot min_v(L_v^{ves}) + U[0; 1.20 \cdot max_v(L_v^{ves}) - min_v(L_v^{ves})]\ \forall b \in B$
Minimum vertical clearance for requirement 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)	$P_v^{hor} = U[50; 100]\ \forall v \in V$
Waiting cost for vessel v: $\varphi_v^{wt}, v \in V$ (U.S. \$/ time periods)	$\varphi_v^{wt} = U[1000; 5000]\ \forall v \in V$
Handling cost for vessel v: $\varphi_v^{ht}, v \in V$ (U.S. \$/ time periods)	$\varphi_v^{ht} = U[50000; 70000]\ \forall v \in V$
Late departure cost for vessel v: $\varphi_v^{lt}, v \in V$ (U.S. \$/ time periods)	$\varphi_v^{lt} = U[5000; 10000]\ \forall v \in V$
Large positive number: Y	10000

Different types of vessels were modeled throughout the numerical experiments such as (Eniram, 2019): (1) Panamax (length: 820.2 ft., draft: 41.0 ft.); (2) Panamax Max (length: 951.4 ft., draft: 41.0 ft.); (3) Post Panamax (length: 935.0 ft., draft: 42.7 ft.); (4) Post Panamax Plus

(length: 984.3 ft., draft: 47.6 ft.); (5) New Panamax (length: 1200.8 ft., draft: 49.9 ft.); and (6) Triple E (length: 1312.3 ft., draft: 50.9 ft.). In this chapter of the dissertation, it is assumed that the MCT should be able to serve all the aforementioned types of vessels. The departure time, negotiated for each vessel, was estimated utilizing the following equation: $\theta_v^d = abs(\theta_v^a + \widetilde{\theta_{vb}^{ht}} \cdot U[1.20; 1.50])\ \forall v \in V, b \in B$. Based on the latter relationship, the departure time of a given vessel is the rounded value of a summation of 2 items: (1) the vessel arrival time (θ_v^a); (2) the handling time at the preferred berthing position ($\widetilde{\theta_{vb}^{ht}}$), increased by 20% to 50% ($U[1.20; 1.50]$); such an increase was applied to consider any disruption, which may negatively affect the handling time.

Table 24 Developed problem instances specification.

Problem Instance	Classification	No. of Vessels	No. of Berths	Problem Instance	Classification	No. of Vessels	No. of Berths
P1	Small-Size	5	2	P25	Large-Size	65	4
P2	Small-Size	5	3	P26	Large-Size	65	6
P3	Small-Size	5	4	P27	Large-Size	65	8
P4	Small-Size	7	2	P28	Large-Size	70	4
P5	Small-Size	7	3	P29	Large-Size	70	6
P6	Small-Size	7	4	P30	Large-Size	70	8
P7	Small-Size	9	2	P31	Large-Size	75	4
P8	Small-Size	9	3	P32	Large-Size	75	6
P9	Small-Size	9	4	P33	Large-Size	75	8
P10	Small-Size	11	2	P34	Large-Size	80	4
P11	Small-Size	11	3	P35	Large-Size	80	6
P12	Small-Size	11	4	P36	Large-Size	80	8
P13	Small-Size	13	2	P37	Large-Size	85	4
P14	Small-Size	13	3	P38	Large-Size	85	6
P15	Small-Size	13	4	P39	Large-Size	85	8
P16	Small-Size	15	2	P40	Large-Size	90	6
P17	Small-Size	15	3	P41	Large-Size	90	8
P18	Small-Size	15	4	P42	Large-Size	90	10
P19	Small-Size	17	2	P43	Large-Size	100	6
P20	Small-Size	17	3	P44	Large-Size	100	8
P21	Small-Size	17	4	P45	Large-Size	100	10
P22	Small-Size	20	3	P46	Large-Size	110	6
P23	Small-Size	20	4	P47	Large-Size	110	8
P24	Small-Size	20	5	P48	Large-Size	110	10

Table 24 includes description of two groups of problem instances (i.e., small-size and large-size problem instances), developed in this chapter of the dissertation. The problem instances were utilized to conduct all the computational experiments and evaluate performance of UIMA against other adopted algorithms (i.e., EA, PSO, EDA, DE, TS, SA, and VNS). Note that all the

EA, PSO, EDA, and DE algorithms, adopted to cover the islands in UIMA, were executed as independent algorithms to evaluate performance of UIMA against each one of them, separately.

7.3.2. Algorithmic parameters tuning

In order to enhance performance of all the algorithms, developed in this chapter of the dissertation, a parameter tuning analysis was conducted to select the appropriate values for the algorithmic parameters. Specifically, the parameter tuning is the process of the appropriate value selection for the algorithmic parameters (Eiben and Smith, 2003; Dulebenets, 2018h).

The parameter tuning can be conducted utilizing one of the two approaches, introduced in the metaheuristic algorithms literature (Eiben and Smith, 2003): (1) "Taguchi's Scheme"; and (2) "Full Factorial Design". The second approach (i.e., "Full Factorial Design") was adopted in this chapter of the dissertation. The parameter tuning results for UIMA, EA, PSO, EDA, DE, TS, SA, and VNS are summarized in Table 25. Note that a superscript "*" is utilized to mark the population size and stopping criterion for EA, PSO, EDA, and DE. The latter means that these parameters were used only when the four aforementioned algorithms were executed independently.

All the algorithmic parameters for UIMA, EA, PSO, EDA, and DE were described in section 7.2 of this dissertation. The TS algorithm stores the solutions discovered throughout the search process in an archive, called Tabu List. The capacity of Tabu List for storage of the solutions already discovered is denoted as (S_{tl}^{TS}) in Table 25 (Glover, 1986). The single-solution-based algorithms, adopted in this chapter, (i.e., TS, SA, and VNS) search the neighborhood of the current solution locally. The local search size of TS and VNS (i.e., S_{ls}^{TS} for TS and S_{ls}^{VNS} for VNS) should be tuned, while the SA just evaluates one of the neighbors of the current solution (Glover, 1986; Das and Chakrabarti, 2005; Hansen and Mladenović, 2010). Note that it was assumed that two vessels to berth assignments (randomly selected) had to be changed in a given solution in order to create a new solution during the local search within the considered single-solution-based algorithms. One of the important algorithmic parameters for SA is the initial Boltzmann temperature (τ), as it models the annealing phenomenon (i.e., the process of controlled heating and cooling) in the metallurgy industry (Das and Chakrabarti, 2005). Moreover, the stopping

criterion ($MaxIter$) for all three single-solution-based algorithms was set during the parameter tuning analysis (see Table 25).

Table 25 Algorithmic parameter tuning results adopting the full factorial design approach.

Algorithm	Parameter	Candidate Values	Selected Value
UIMA	Population size ($PopSize$)	[240; 280; 320]	240
UIMA	Immigration 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* ($PopSize$)	[40; 50; 60]	60
EA	Crossover rate (σ^c)	[0.3; 0.5; 0.8]	0.8
EA	Mutation rate (σ^m)	[0.01; 0.04; 0.06]	0.06
EA	Number of selected solutions at each	[4; 5; 6]	4
EA	Stopping criterion* ($MaxGen$)	[4000; 5000; 6000]	6000
PSO	Population size* ($PopSize$)	[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* ($MaxIter$)	[4000; 5000; 6000]	6000
EDA	Population size* ($PopSize$)	[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* ($MaxGen$)	[4000; 5000; 6000]	6000
DE	Population size* ($PopSize$)	[40; 50; 60]	60
DE	Mutation coefficient (α)	[0.4; 0.6; 0.8]	0.8
DE	Crossover probability (C^{cr})	[0.3; 0.5; 0.6]	0.3
DE	Stopping criterion* ($MaxGen$)	[4000; 5000; 6000]	6000
TS	Tabu list size (S_{tl}^{TS})	[10; 15; 20]	20
TS	Local search size (S_{ls}^{TS})	[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
VNS	Local search size (S_{ls}^{VNS})	[15; 20; 25]	20
VNS	Stopping criterion ($MaxIter$)	[4000; 5000; 6000]	6000

7.3.3. Optimality gap analysis

The quality of solutions discovered by developed algorithms can be evaluated against the global optima using the optimality gap analysis (Dulebenets et al., 2018b). In order to discover the global optima for the small-size problem instances, the SCBSP was encoded in the GAMS environment. GAMS utilizes CPLEX to solve the mixed-integer linear mathematical models. The

computational time and the relative optimality gap for CPLEX were limited to 7200 seconds and 0.01%, respectively. Note that the large-size problem instances could not be solved by CPLEX considering the limited computational time, as the larger the size of the problem instance, the larger the search space and more computational effort is required for discovering the global optimum.

The objective values and computational times, returned by CPLEX, UIMA, and the population-based metaheuristic algorithms (i.e., EA, PSO, EDA, and DE) that were developed in this chapter of the dissertation, are provided in Table 26. The average values of objective function and computational time over the small-size problem instances were estimated by execution of a total of 10 replications for each one of the algorithms. The outmost left column includes the list of the small-size problem instances (i.e., P1 to P24) used to conduct the optimality gap analysis. Moreover, Table 26 includes the number of vessels and berthing positions for each problem instance along with the objective values (U.S. $) and computational times (sec.), recorded for the considered solution methodologies (i.e., CPLEX, UIMA, EA, PSO, EDA, and DE). Based on the results presented in Table 26, the problem instances P1 to P19 could be successfully solved by CPLEX considering the defined stopping criterion (i.e., 7200 seconds), whereas CPLEX could not return the global optima for problem instances P20 to P24. The latter can be justified by the fact that the search space of problem instances P20 to P24 was too large, and CPLEX was not able to search that search space for the global optimum given the limited computational time.

According to the information provided in Table 26, UIMA could solve problem instances P1 to P13 to the global optimality (i.e., the objective function values returned by UIMA and CPLEX are equal), while none of the other solution algorithms could return the global optimum for P11 to P13. Also, the maximum optimality gap of the UIMA algorithm was lower than 2.50%. The latter findings show the efficiency of UIMA as compared to other algorithms due to the fact that UIMA utilizes several search tools (i.e., EA, PSO, EDA, and DE) with their own specific features for discovering more promising areas in the search space. Nonetheless, EA found the global optimum for problem instances P1 to P10, and PSO, EDA, and DE could discover the global optimum for problem instances P1 to P9. The latter results show a better performance of EA as compared to PSO, EDA, and DE, which can be explained by the fact that all the algorithmic operations of EA are adjusted to the integer-coded nature of SCBSP.

Table 26 Objective values and computational times by CPLEX, UIMA, and the developed 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 (U.S. $)	CPU Time, (sec)	Objective Value, 10^7 (U.S. $)	CPU Time, (sec)	Objective Value, 10^7 (U.S. $)	CPU Time, (sec)	Objective Value, 10^7 (U.S. $)	CPU Time, (sec)	Objective Value, 10^7 (U.S. $)	CPU Time, (sec)	Objective Value, 10^7 (U.S. $)	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	0.9	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	0.8	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	0.7	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	1.5	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	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	7328.1	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 27 Objective values and computational times by CPLEX, UIMA, and the developed single-solution-based algorithms for the small-size problem instances.

Problem Instance	No. of Vessels	No. of Berths	CPLEX		UIMA		VNS		TS		SA	
			Objective Value, 10^7 (U.S. $)	CPU Time, (sec)	Objective Value, 10^7 (U.S. $)	CPU Time, (sec)	Objective Value, 10^7 (U.S. $)	CPU Time, (sec)	Objective Value, 10^7 (U.S. $)	CPU Time, (sec)	Objective Value, 10^7 (U.S. $)	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	0.9	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	0.8	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	0.7	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	1.5	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	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	7328.1	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**

On the other hand, PSO and DE rely on specific algorithmic operations, which cannot handle integer-coded solutions and need to map the solutions to the real-coded ones. This issue can disrupt the building block structures in the solutions, which negatively affects performance of PSO and DE. On the other hand, EDA utilizes a probability matrix to discover new areas in the search space. Specifically, EDA generates new solutions by assignment of vessels to berthing positions one by one, and also the probability matrix is shaken in each generation (see section 7.2.1.3 for more details). Two latter processes in EDA may interrupt the structure of building blocks and worsen performance of EDA. Furthermore, the computational time of CPLEX was substantially lower than other algorithms for the problem instances P1 to P6. Besides, for the next problem instances (i.e., P7 to P24) the computational time by CPLEX increased exponentially (on average equal to 1837 sec.), while the other algorithms converged in a remarkably more practical computational time (on average between about 12 to 41 sec.).

The results returned by CPLEX, UIMA, and the single-solution-based metaheuristic algorithms (i.e., VNS, TS, and SA) over the small-size problem instances (i.e., P1 to P24) are presented in Table 27. It can be observed that VNS, TS, and SA could solve the problem instances P1 to P7 to the global optimality, while UIMA returned the global optimum for problem instances P1 to P13. Weak performance of VNS, TS, and SA as compared to UIMA can be explained due to their single-solution-based nature. VNS, TS, and SA utilize a single point to start exploring the search space, which drastically affects their ability for exploration of new and promising areas in the search space as compared to the population-based metaheuristic algorithms. The latter can also explain the short computational times that were required by VNS, TS, and SA. Specifically, the single-solution-based algorithms just deal with one solution (and its neighbors) at each iteration, whereas the population-based ones work with a population of solutions in parallel (i.e., more than one solution), which obviously increases the computational effort. However, the computational effort required by population-based algorithms was found to be quite reasonable from the practical standpoint.

7.3.4. Developed algorithm evaluation against the alternative metaheuristics

In this section of the dissertation some widely-used performance indicators in the BSP literature are utilized to evaluate performance of the designed UIMA algorithm and all other

developed metaheuristic algorithms. In section 7.3.4.1, the developed UIMA algorithm and all the other population-based and single-solution-based algorithms, adopted in this chapter of the dissertation (i.e., EA, PSO, EDA, DE, VNS, TS, and SA), are used to solve the large-size problem instances. In section 7.3.4.2, the convergence patterns are plotted for all the adopted solution algorithms considering the 12 largest problem instances; moreover, the convergence rate is calculated for each one of the developed algorithms over all the large-size problem instances. Performance of the solution algorithms in terms of the stochastic variations is analyzed in section 7.3.4.3. The capability of the algorithms in terms of the exploration and exploitation of different areas of the search space is analyzed by estimation of the population diversity values throughout the search process in section 7.3.4.4.

7.3.4.1. Evaluation of solution quality and computational efforts

The objective function and computational time values, as two important performance indicators, are estimated using all the metaheuristic algorithms, developed in this chapter of the dissertation, for the large-size problem instances. The quality of the solutions returned by a given algorithm is highlighted by the objective function value, where the computational time illustrates the computational effort required by the solution methodology to solve the mathematical model. Moreover, the algorithmic performance can be evaluated from the practical standpoint utilizing the computational time (Dulebenets et al., 2018b). Table 28 includes the average values of the objective function and computational time over 10 algorithmic replications of UIMA, and the developed population-based algorithms (i.e., EA, PSO, EDA, and DE) for the large-size problem instances (i.e., P25 to P48).

As it can be observed in Table 28, the UIMA algorithm improved the objective function values of EA, PSO, EDA, and DE by 11.03%, 12.03%, 12.92%, and 11.76%, respectively. The latter can be supported by the fact that UIMA takes the advantage of the unique specifications of EA, PSO, EDA, and DE to discover different areas (i.e., domains) of the search space. Hence, the UIMA algorithm, which relies on the features of four different search tools, can discover new areas in the search space more effectively as compared to each one of the EA, PSO, EDA, and DE algorithms alone.

Table 28 Objective values and computational times by UIMA and the developed population-based algorithms for large-size problem instances.

Problem Instance	No. of Vessels	No. of Berths	UIMA		EA		PSO		EDA		DE	
			Objective Value, 10^7 (U.S. $)	CPU Time, (sec.)	Objective Value, 10^7 (U.S. $)	CPU Time, (sec.)	Objective Value, 10^7 (U.S. $)	CPU Time, (sec.)	Objective Value, 10^7 (U.S. $)	CPU Time, (sec.)	Objective Value, 10^7 (U.S. $)	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 29 Objective values and computational times by UIMA and the developed single-solution-based algorithms for large-size problem instances.

Problem Instance	No. of Vessels	No. of Berths	UIMA		VNS		TS		SA	
			Objective Value, 10^7 (U.S. $)	CPU Time, (sec.)	Objective Value, 10^7 (U.S. $)	CPU Time, (sec.)	Objective Value, 10^7 (U.S. $)	CPU Time, (sec.)	Objective Value, 10^7 (U.S. $)	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**

The traditional EA algorithm works based on the constant crossover and mutation probabilities. Specifically, EAs keep exploring and exploiting the search space from the beginning towards the convergence, and all the EA operators are compatible with the integer-coded nature of SCBSP. On the other hand, PSO and DE rely on the algorithmic operators, which are not adjusted to the integer-coded nature of SCBSP. The latter leads to generation of the real-coded solutions by PSO and DE, which need to be mapped to the integer-coded ones throughout the search process. The building block structures might be ruined throughout the process of mapping, which can negatively affect performance of the algorithm. Besides, EDA utilizes the probability matrix to assign vessels to the berthing positions one by one and to generate new solutions; moreover, the probability matrix is shaken in each iteration (see section 7.2.1.3 for more details). The latter two processes throughout the search process in EDA may ruin the building blocks and negatively impact performance of EDA as well.

In terms of computational effort, the average computational times over 10 replications of the large-size problem instances for UIMA, EA, PSO, EDA, and DE are equal to 211.81, 155.60, 70.29, 132.26, and 86.34 seconds, respectively. Based on the aforementioned results, the highest computational time was recorded for UIMA. The latter can be justified by the fact that UIMA has several extra steps throughout the search process as compared to each one of the population-based algorithms executed independently. The extra steps in UIMA comprise: (1) UIMA initializes a population with the size of 240 solutions, while other algorithms generate an initial population with the size of 60 solutions; (2) process of repairing the solutions; (3) assignment of a sub-population to each one of the algorithms; (4) exchanging immigrants, which includes the selection of immigrants in the origin islands and removal of some individuals in the destination islands; and (5) checking the migration criterion. Although UIMA returned the highest computational time, it is completely justifiable from the practical point of view.

The average objective function and computational time values over 10 replications of large-size problem instances, recorded for UIMA, VNS, TS, and SA, are presented in Table 28. The UIMA algorithm returned substantially better objective function values and outperformed VNS, TS, and SA by 19.01%, 20.21%, and 19.13%, respectively. The latter can be supported due to the single-solution-based nature of VNS, TS, and SA as compared to UIMA. Besides, the

computational time values, recorded for UIMA, VNS, TS, and SA, are equal to 211.81, 11.62, 12.74, and 11.85 seconds, respectively. As stated before, a relatively high computational time, recorded for UIMA, is practically justifiable. Furthermore, application of UIMA increased the revenue by about 14,000,000 U.S. $ as compared to the other single-solution-based algorithms; hence, UIMA can be introduced as an economically promising decision support tool.

A total of 7 paired z-tests were conducted to evaluate how significantly the UIMA algorithm improved the solutions for the developed SCBSP as compared to the other algorithms, adopted in this chapter of the dissertation (i.e., EA, PSO, EDA, DE, VNS, TS, and SA). Note that each z-test was implemented over the average of objective values, obtained by 10 replications that were executed for each algorithm and each large-size problem instance. The null hypothesis was defined for all the z-tests as follows: it was assumed that the average objective values by UIMA were equal to the average objective values by an alternative algorithm. On the other hand, the alternative hypothesis was defined as follows: it was assumed that the average objective values by UIMA were different as compared to the average objective values by an alternative algorithm. The results obtained by z-tests are presented in Table 30. An index is attributed to each one of the z-tests listed in the outmost left column in Table 30, where the second column comprises the algorithms, which were participated in each z-test. The number of observations is the same as the number of large-size problem instances, developed in this chapter of the dissertation. Furthermore, Table 30 provides the information regarding the average objective values of the large-size problem instances by each participant algorithm (equal to the last rows in Table 28 and Table 29), and the calculated standard deviations and z-values as well.

The level of confidence was assumed to be equal to 5.00% for this chapter of the dissertation, which corresponds to the critical z-value of 1.96. The comparison between the critical z-value and the obtained z-values shows that the null hypothesis in all the paired z-tests is rejected. The rejection of the null hypothesis implies that the average objective function values, provided by UIMA for the large-size problem instances, are statistically different and lower as compared to the average objective values returned by other algorithms, developed in this chapter of the dissertation (i.e., EA, PSO, EDA, DE, VNS, TS, and SA).

Table 30 The results of conducted statistical test utilizing z-test.

Test	Participant Algorithms	Number of Observations	Average Objective Value 10^7 (U.S. $)	Standard Deviation 10^6 (U.S. $)	z-Value
1	UIMA	24	7.441	1.554	**2.59**
	EA	24	8.261	1.724	
2	UIMA	24	7.441	1.182	**2.82**
	PSO	24	8.336	1.737	
3	UIMA	24	7.441	1.182	**3.03**
	EDA	24	8.402	1.754	
4	UIMA	24	7.441	1.182	**2.76**
	DE	24	8.316	1.732	
5	UIMA	24	7.441	1.182	**4.46**
	VNS	24	8.855	1.851	
6	UIMA	24	7.441	1.182	**4.74**
	TS	24	8.945	1.851	
7	UIMA	24	7.441	1.182	**4.49**
	SA	24	8.864	1.861	

7.3.4.2.Solution improvement throughout the search process

The progress of each one of the adopted algorithms (i.e., EA, PSO, EDA, DE, VNS, TS, and SA) throughout the search process is analyzed under this section of the dissertation. Specifically, the average objective function values over 10 replications, returned by a given algorithm in each generation/iteration (the term "generation" is related to EA, EDA, and DE, whereas the term "iteration" works for UIMA, PSO, VNS, TS, and SA), is plotted against the number of generations/iterations mapped to the range [0,1]. This analysis illustrates the behavior of the algorithm throughout the search process, where one of its important applications is to evaluate the adopted stopping criterion. In detail, if the diagram approaches a horizontal line towards the termination of the algorithmic run, it can be concluded that the appropriate stopping criterion has been adopted for the algorithm. The stopping criterion for UIMA was limited to 1600 iterations, whereas a total of 6000 generations/iterations were considered as the stopping criterion for all other algorithms (i.e., EA, PSO, EDA, DE, VNS, TS, and SA). In the convergence pattern diagrams (Figure 51), the number of the generations/iterations (X axis) is plotted against the average objective function values over 10 replications (Y axis) discovered in the corresponding generation/iteration. Considering the different stopping criteria adopted for UIMA and the other algorithms, it was challenging to plot the convergence patterns of all the algorithms in a single diagram. Hence, the number of generations/iterations was mapped to the range (0,1] for all the

algorithms to solve the latter issue, where the X axis in the convergence pattern diagrams covers the range (0,1]. To do the latter, the regular numbers on the X axis (i.e., $X_i = 1,2, \dots, SC^{Alg}$, where SC^{Alg} is the stopping criterion adopted for the algorithm Alg), were replaced with the values mapped to the range (0,1] using this relationship: $X_i = \frac{i}{SC^{Alg}}, i \in [1, SC^{Alg}]$.

The convergence pattern diagrams plotted for the algorithms, developed in this chapter of the dissertation (i.e., UIMA, EA, PSO, EDA, DE, VNS, TS, and SA), are illustrated in Figure 51 considering the 12 largest problem instances (i.e., P37 to P48). Since all the algorithms, adopted in this chapter, utilized the same initial population generation methodology (i.e., the FCFS-SR heuristic), all the convergence pattern diagrams start from the same point in Figure 51 (note that only one initial solution of the single-solution-based algorithms was initialized using FCFS-SR as well). It can be observed that UIMA starts finding more promising domains as compared to the other algorithms from the beginning of the search process. The latter can be supported by the fact that UIMA relies on four different search tools with different features; hence, UIMA is able to explore new domains in the search space more effectively as compared to other developed algorithms (i.e., EA, PSO, EDA, DE, VNS, TS, and SA). The superior exploitative ability of UIMA as compared to the other algorithms can be observed in Figure 51 as well, as UIMA keeps improving the quality of the solutions towards the convergence. Note that similar convergence patterns were observed for the other large-size problem instances.

The estimation of the convergence rate values for the developed metaheuristic algorithms over the large-size problem instances is included in the scope of computational experiments in this chapter of the dissertation. The convergence rate or convergence speed is defined based on the number of objective function evaluations until finding the minimum objective value, divided by the total number of objective function evaluations (Pelusi et al., 2018). Note that all the developed metaheuristic algorithms were executed 10 times utilizing the large-size problem instances to estimate the average convergence rate values. The information of the large-size problem instances and the results recorded from the conducted numerical experiments are provided in Table 31: (1) the average convergence rate over 10 replications denoted as CR (Avg); and (2) the average standard deviation over 10 replications denoted as CR (SD).

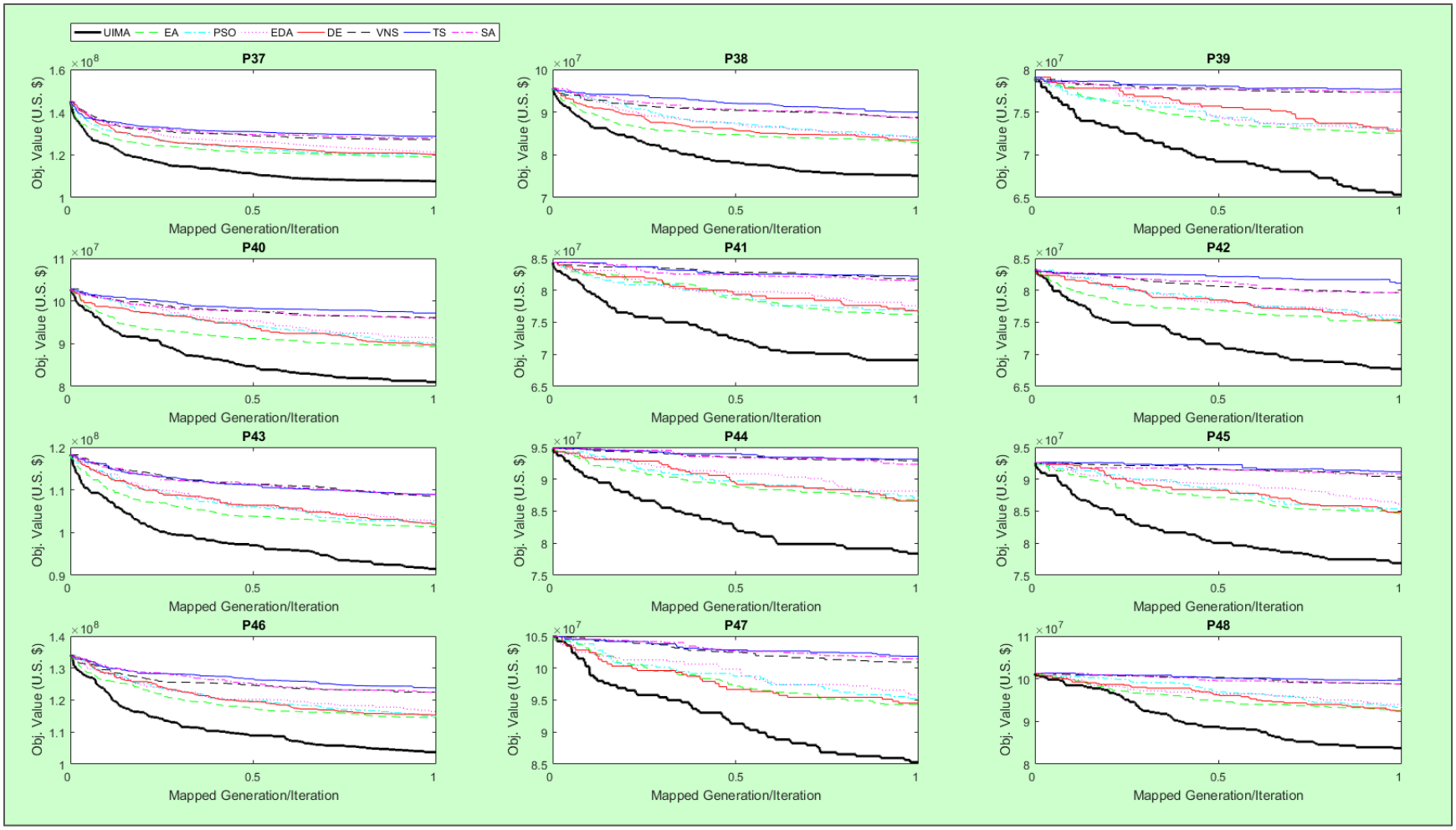

Figure 51 The convergence pattern diagrams for all the algorithms considering problem instances P37 to P48.

Table 31 The convergence rate values recorded for the developed algorithms over the large-size problem instances.

Problem Instance	No. of Vessels	No. of Berths	UIMA		EA		PSO		DE		EDA		VNS		TS		SA	
			CR (Avg)	CR (SD)	CR (Avg)	CR (SD)	CR (Avg)	CR (SD)	CR (Avg)	CR (SD)	CR (Avg)	CR (SD)	CR (Avg)	CR (SD)	CR (Avg)	CR (SD)	CR (Avg)	CR (SD)
P25	65	4	0.880	0.119	0.958	0.039	0.760	0.065	0.891	0.062	0.916	0.044	0.948	0.139	0.997	0.131	0.768	0.088
P26	65	6	0.947	0.106	0.976	0.065	0.929	0.122	0.930	0.050	0.928	0.040	0.991	0.053	0.813	0.085	0.948	0.066
P27	65	8	0.870	0.130	0.983	0.037	0.888	0.145	0.950	0.115	0.911	0.066	0.991	0.056	0.908	0.094	0.788	0.043
P28	70	4	0.910	0.055	0.972	0.062	0.799	0.139	0.776	0.041	0.928	0.050	0.958	0.163	0.939	0.102	0.899	0.038
P29	70	6	0.959	0.054	0.967	0.069	0.934	0.094	0.782	0.076	0.728	0.073	0.804	0.162	0.979	0.068	0.882	0.058
P30	70	8	0.846	0.131	0.876	0.038	0.890	0.082	0.958	0.050	0.839	0.034	0.912	0.154	0.762	0.122	0.989	0.073
P31	75	4	0.890	0.092	0.915	0.089	0.849	0.110	0.949	0.129	0.800	0.033	0.894	0.071	0.803	0.062	0.762	0.101
P32	75	6	0.928	0.038	0.806	0.098	0.926	0.140	0.791	0.043	0.845	0.044	0.787	0.104	0.808	0.068	0.971	0.032
P33	75	8	0.919	0.075	0.920	0.070	0.956	0.107	0.821	0.080	0.786	0.054	0.857	0.151	0.998	0.107	0.774	0.083
P34	80	4	0.874	0.090	0.794	0.086	0.889	0.091	0.801	0.105	0.766	0.077	0.789	0.135	0.787	0.030	0.895	0.053
P35	80	6	0.817	0.036	0.895	0.094	0.840	0.153	0.799	0.099	0.812	0.052	0.816	0.165	0.834	0.098	0.944	0.080
P36	80	8	0.842	0.104	0.941	0.029	0.904	0.148	0.877	0.086	0.876	0.039	0.767	0.026	0.995	0.083	0.840	0.058
P37	85	4	0.938	0.069	0.867	0.059	0.936	0.168	0.947	0.051	0.864	0.051	0.798	0.067	0.786	0.127	0.844	0.067
P38	85	6	0.968	0.143	0.910	0.087	0.902	0.189	0.897	0.126	0.844	0.056	0.999	0.155	0.976	0.125	0.836	0.053
P39	85	8	0.968	0.053	0.911	0.087	0.953	0.150	0.958	0.047	0.816	0.052	0.947	0.130	0.833	0.079	0.857	0.086
P40	90	6	0.868	0.030	0.961	0.094	0.899	0.072	0.797	0.130	0.747	0.047	0.863	0.127	0.933	0.105	0.794	0.042
P41	90	8	0.817	0.078	0.958	0.050	0.926	0.153	0.861	0.045	0.880	0.049	0.913	0.144	0.881	0.054	0.780	0.090
P42	90	10	0.926	0.042	0.896	0.090	0.949	0.068	0.782	0.086	0.908	0.040	0.774	0.025	0.951	0.055	0.845	0.100
P43	100	6	0.966	0.095	0.808	0.098	0.826	0.051	0.920	0.054	0.834	0.045	0.897	0.092	0.971	0.096	0.994	0.083
P44	100	8	0.890	0.072	0.775	0.038	0.760	0.053	0.822	0.101	0.930	0.078	0.888	0.074	0.990	0.119	0.777	0.045
P45	100	10	0.936	0.062	0.839	0.058	0.759	0.160	0.810	0.100	0.776	0.056	0.805	0.035	0.803	0.148	0.901	0.043
P46	110	6	0.837	0.147	0.794	0.067	0.843	0.102	0.867	0.082	0.759	0.076	0.998	0.033	0.769	0.135	0.882	0.072
P47	110	8	0.935	0.101	0.970	0.029	0.876	0.087	0.953	0.091	0.803	0.046	0.818	0.148	0.863	0.096	0.980	0.090
P48	110	10	0.890	0.135	0.791	0.063	0.767	0.151	0.977	0.062	0.768	0.033	0.984	0.140	0.861	0.093	0.784	0.094
Average			**0.901**	**0.086**	**0.895**	**0.067**	**0.873**	**0.117**	**0.872**	**0.08**	**0.836**	**0.051**	**0.883**	**0.106**	**0.885**	**0.095**	**0.864**	**0.068**

As it can be observed in Table 31, fairly large values were recorded for all the developed algorithms over the large-size problem instances (i.e., the average convergence rate values for UIMA, EA, PSO, EDA, DE, VNS, TS, and SA over all the large-size problem instances are equal to 0.901, 0.895, 0.873, 0.872, 0.836, 0.883, 0.885, and 0.864, respectively). The recorded convergence rate values are remarkably close to 1 for all the algorithms, which highlights that all the metaheuristics kept finding better solutions as compared to the previous generation/iteration towards the convergence. However, the superior explorative characteristics of UIMA enabled the algorithm to discover more promising domains in the search space during the initial iterations and outperform other solution algorithms (see section 7.3.4.1 for more details).

7.3.4.3.Analysis of stochastic performance of the algorithms

The metaheuristic algorithms return a different solution from one replication to another for a given problem instance due to their stochastic nature. However, if the difference between the solutions, returned after several replications, is not too high, performance of a given metaheuristic can be acceptable from the practical standpoint; otherwise, the algorithm cannot be considered as a reliable decision support tool. The reliability of a given algorithm can be assessed by conducting an analysis on stochastic performance of the algorithm. The coefficient of variation was calculated to evaluate the stochastic performance of the algorithms, developed in this chapter of the dissertation (i.e., UIMA, EA, PSO, EDA, DE, VNS, TS, and SA).

Note that the coefficient of variation was calculated for the objective function values, returned by metaheuristic algorithms over 10 algorithmic replications for the large-size problem instances. The equation below was adopted to estimate the coefficient of variation:

$$CV_{inst}^{Alg} = {SD_{inst}^{Alg}}\Big/{Mean_{inst}^{Alg}} \tag{70}$$

where CV_{inst}^{Alg} is the coefficient of variation calculated over 10 replications of algorithm Alg for problem instance $inst$; and SD_{inst}^{Alg} and $Mean_{inst}^{Alg}$ are the standard deviation and average of objective function values calculated over 10 replications of algorithm Alg for problem instance $inst$, respectively.

The coefficient of variations, estimated over 10 replications of the developed metaheuristic algorithms for the large-size problem instances (i.e., P25 to P48), are presented in Table 32. The results provided in Table 32 show that the maximum coefficient of variation was recorded for SA, which is equal to 2.073%. A fairly low value of the maximum recorded coefficient of variation demonstrates the stability of the algorithms, developed in this chapter of the dissertation. Moreover, lower objective function coefficient of variation values were recorded for UIMA and other population-based metaheuristic algorithms, which demonstrates the superiority of these algorithms as compared to the single-solution-based metaheuristics in terms of stochastic reliability.

Table 32 The coefficient of variation calculaated for the objective function values returned by developed metaheuristic algorithms over 10 replications of large-size problem instances.

Problem Instance	UIMA	EA	PSO	EDA	DE	VNS	TS	SA
	CV (%)	CV (%)	CV (%)	CV (%)	CV (%)	CV (%)	CV (%)	CV (%)
P25	0.992	0.312	0.346	1.684	1.119	1.606	1.512	0.458
P26	0.728	1.465	1.892	1.631	0.392	0.733	1.002	1.401
P27	0.343	1.597	0.470	0.187	1.588	0.087	1.416	0.519
P28	0.118	1.405	0.106	0.940	1.115	1.198	1.977	0.562
P29	0.843	1.361	0.124	1.910	0.253	2.029	0.498	1.799
P30	0.345	1.817	1.230	0.955	1.208	1.932	1.018	2.026
P31	0.174	1.724	1.999	0.310	1.849	1.747	1.206	2.073
P32	0.971	0.694	1.677	0.442	0.825	1.892	1.373	1.309
P33	0.534	0.834	1.228	1.943	1.419	1.151	0.801	0.724
P34	0.774	1.571	1.331	1.133	1.792	1.539	0.171	1.666
P35	0.115	0.652	0.955	0.530	0.850	0.059	0.142	1.222
P36	0.640	1.667	1.776	0.929	1.632	1.903	0.382	0.767
P37	0.207	1.102	0.279	1.295	1.153	0.242	0.838	1.921
P38	0.406	1.624	1.743	1.583	1.461	1.630	0.791	0.276
P39	0.851	0.768	0.325	0.427	1.266	0.207	0.180	0.517
P40	0.338	1.181	0.930	1.201	0.752	0.550	0.734	0.790
P41	0.844	0.910	0.706	0.688	1.687	1.217	0.931	0.607
P42	0.386	0.500	0.915	1.132	1.221	0.473	0.489	1.703
P43	0.958	1.127	1.310	1.786	0.901	2.072	1.986	1.560
P44	0.198	0.670	0.665	1.076	0.856	1.582	0.501	1.475
P45	0.899	1.605	0.529	1.985	0.223	0.493	1.607	1.835
P46	0.853	1.333	1.024	0.776	0.238	0.119	0.789	0.509
P47	0.245	1.512	1.266	1.787	1.829	0.274	0.598	1.850
P48	0.283	1.526	1.165	0.841	0.808	0.717	0.349	1.219
Maximum	**0.992**	**1.817**	**1.999**	**1.985**	**1.849**	**2.072**	**1.986**	**2.073**
Minimum	**0.115**	**0.312**	**0.106**	**0.187**	**0.223**	**0.059**	**0.142**	**0.276**
Average	**0.544**	**1.206**	**1.000**	**1.132**	**1.102**	**1.060**	**0.887**	**1.199**

7.3.4.4.Population diversity analysis

The explorative and exploitative capabilities of a given metaheuristic algorithm can be evaluated utilizing the population diversity as a performance indicator. A diverse population can cover different areas of the search space, which includes more promising domains of the search space (i.e., high-quality solutions) and other areas of the search space that contain solutions with medium and low quality (Eiben and Smith, 2015). A given algorithm with poor population diversity might not be able to explore new promising domains in the search space and consequently returns a local optimum as the best-found solution (i.e., premature convergence). The population diversity was estimated for all the developed population-based algorithms (i.e., UIMA, EA, PSO, EDA, DE) over the large-size problem instances in this chapter of the dissertation. To do the latter, the standard deviation of the objective function values at each generation was calculated for the first replication of the population-based metaheuristic algorithms, developed in this chapter of the dissertation for the six largest problem instances (i.e., P43 to P48).

In Figure 52, a limited number of generations/iterations are shown in the diagrams to make the results more clear. Specifically, the results in the mapped range $(0\ 0.05]$ (equivalent to iterations from 1 to 80 for UIMA and 1 to 300 for the others) are presented in the diagrams on the left side, and the ones on the right side illustrate the results in the mapped range $[0.5\ 1]$ (equivalent to iterations from 800 to 1600 for UIMA and 3000 to 6000 for the others). Note that similar patterns were observed for other problem instances and replications.

As it can be observed in Figure 52, a fairly high population diversity ($\approx 5 \times 10^7\ U.S.\$$) is shown at the beginning for all the population-based algorithms. The latter can be justified by the fact that the initial population for all the algorithms is generated using a combination of a FCFS-SR heuristic and a random population generation approach. Although, the randomly generated solutions might represent low-quality areas in the search space, they can positively increase the population diversity. However, the population diversity dropped to the much lower fitness function standard deviation values after several generations/iterations. The latter means that the algorithms discovered more promising areas of the search space and started exploiting them to improve the quality of the solutions.

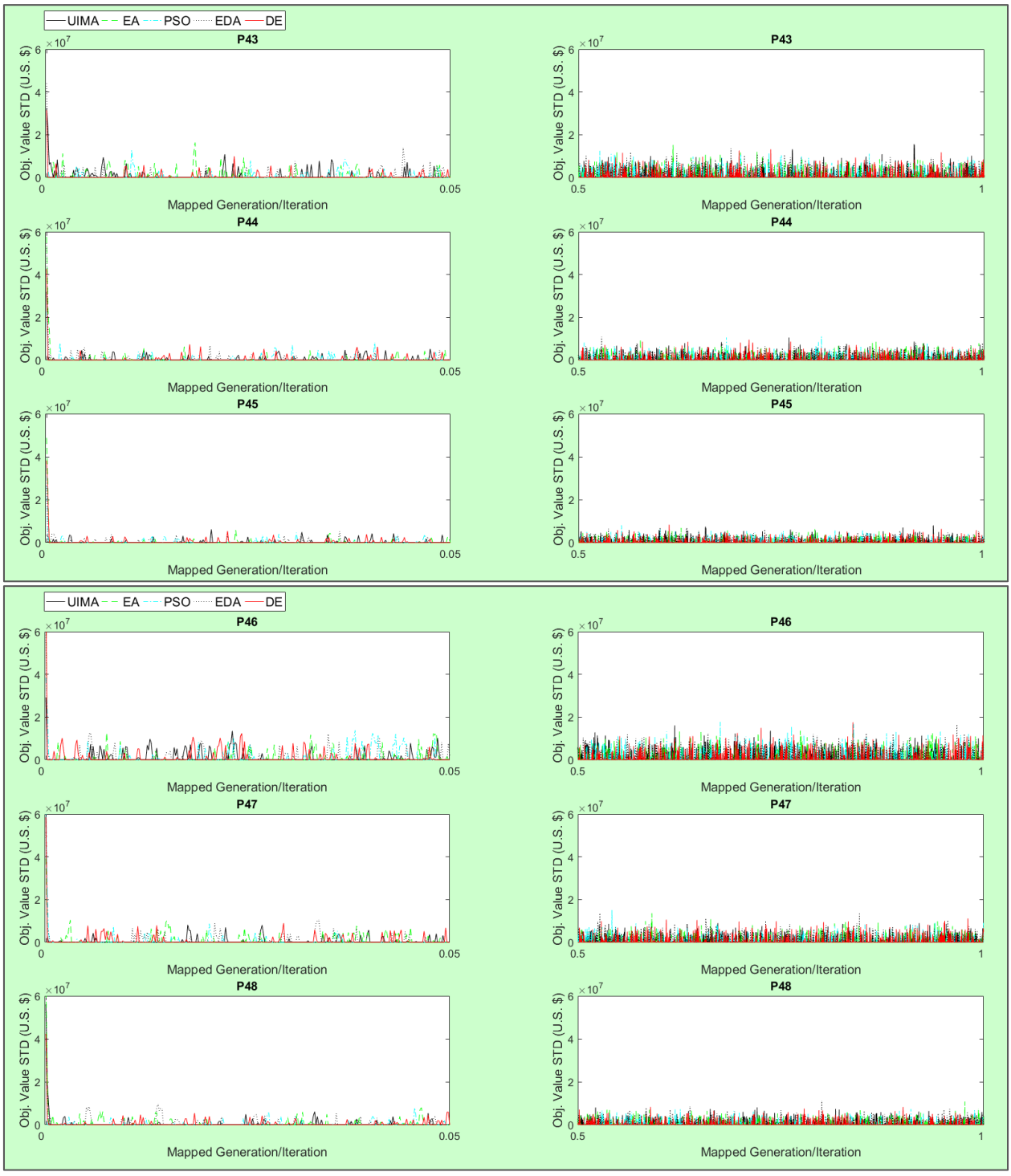

Figure 52 The population diversity estimated for the population-based algorithms over three largest problem instances (i.e., P46 to P48).

Based on the diagrams in Figure 52, all the algorithms kept a reasonable level of the population diversity ($\approx 0.5 \times 10^7\ U.S.\$$) towards the convergence. The diagrams in Figure 52 demonstrated that fairly the same performance was observed for UIMA and other population-based algorithms (i.e., EA, PSO, EDA, DE) in terms of population diversity; however, the UIMA algorithm showed a superior performance considering the solution quality at termination (see section 7.3.4.1).

7.3.5. Managerial insights

Performance of the UIMA algorithm to assist the MCT operator with managerial insights is evaluated under this section of the dissertation. Two scenarios were developed to conduct the managerial insight analysis as follows: (1) variations in the number of available berthing positions at the MCT to assess its impact on the objective function values, handling time, waiting time, and late departure time; and (2) variations in the values for the unit cost components (i.e., unit handling cost, unit waiting cost, and unit late departure cost) in the SCBSP mathematical model to evaluate their impacts on the objective function values, handling time, waiting time, and late departure time.

7.3.5.1. Sensitivity analysis for the number of available berthing positions

A total of 13 scenarios were developed to evaluate the impact of the number of available berthing positions on the solutions (i.e., the generated berth scheduling plan) returned by UIMA. The 13 aforementioned scenarios were generated using the problem instance P48. The number of available berthing positions was changed from 4 to 16 with an increment of 1 berthing position, and the number of vessels was kept constant and equal to 110 arriving vessels. Note that the same information, used for basic numerical experiments, was utilized for all the other information, required for the execution of UIMA (a comprehensive explanation regarding the generated input data is available in section 7.3.1). Each one of the developed 13 scenarios was solved by the UIMA algorithm. A total of 10 algorithmic replications were performed to estimate the average values of total vessel handling times ($\sum_{v\in V}\sum_{b\in B}\sum_{t\in T}\left(\theta_{vb}^{ht}\boldsymbol{x}_{vbt}\right)$), total vessel waiting times ($\sum_{v\in V}(\boldsymbol{\theta}_v^{wt})$), and total vessel late departure times ($\sum_{v\in V}(\boldsymbol{\theta}_v^{lt})$). The results of the numerical experiments conducted over all the 13 scenarios are illustrated in Figure 53.

It can be observed in Figure 53 that three diagrams plotted for the objective function values, waiting time, and late departure time demonstrate a sharp decreasing tendency at the beginning (from 4 to 8 available berthing positions), and then they tend to approach to almost constant value as the number of available berthing positions increases (for 9 to 16 available berthing positions). The latter can be justified by the fact that in case of existence of a limited number of berthing positions available (between 4 to 8 berthing positions), the arriving vessels had to stay in the waiting area for a long time to get served, and a long stay in the waiting area consequently caused a significant late departure time for the arriving vessels. Besides, by increasing the number of available berthing positions (between 9 to 16 berthing positions), the MCT operator could serve the vessels once they arrived at the MCT, and the vessels did not have to stay in the waiting area. The latter led to zero waiting time and accordingly zero late departure time. Furthermore, the total objective function value was mostly composed of the total handling cost of arriving vessels considering the fact that the total waiting time and the total late departure time were almost equal to zero (see equation (43) in section 7.1).

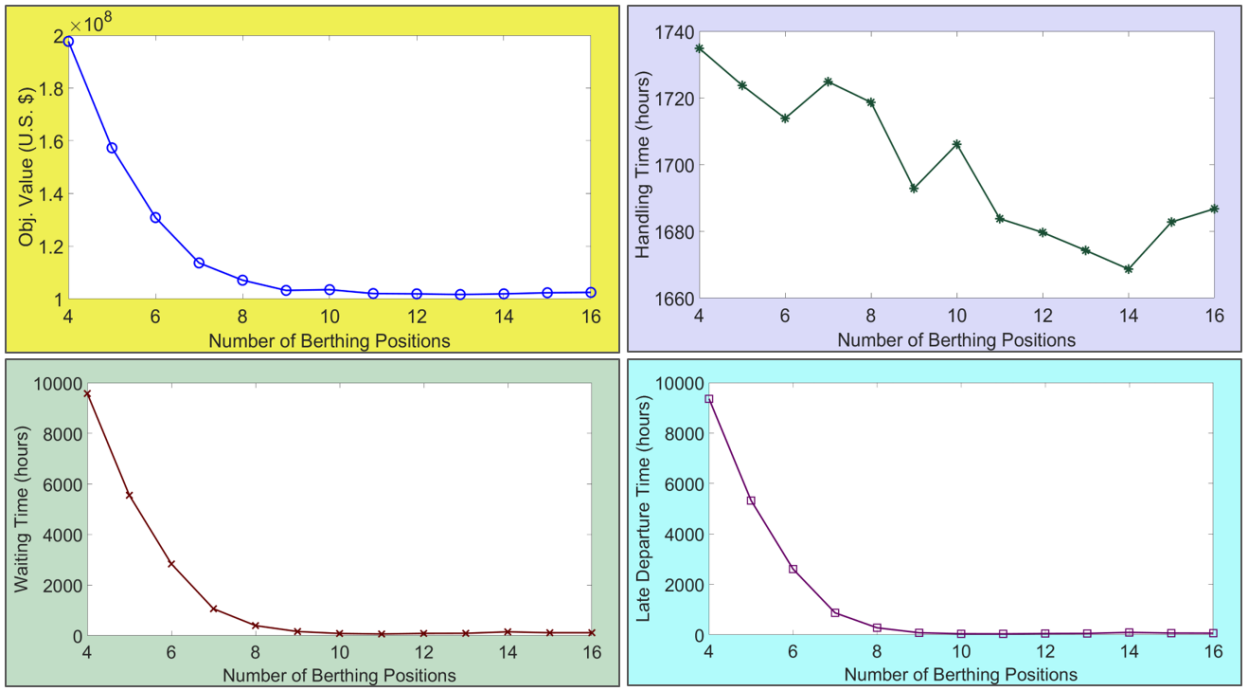

Figure 53 Sensitivity of berth schedule to the number of available berthing positions.

As indicated earlier (section 7.3.1 of the dissertation), the location of the berthing position, assigned to a given vessel, has the most important role on the handling time of the vessel. In detail, a preferred berthing position was defined for each one of the arriving vessels, where the further the assigned berthing position from the preferred one, the longer the handling time. Based on the latter fact, the diagram that corresponds to the total handling time in Figure 53 shows a fluctuating behavior (but generally decreasing) for different number of available berthing positions. All the diagrams, presented in Figure 53, demonstrate that the berth scheduling plans developed by UIMA tried to keep the minimum possible objective function values by assignment of vessels to the first available berthing positions. In general, the UIMA algorithm aimed to assign as many vessels as possible to their preferred berthing positions (therefore, the total handling time typically decreased from one scenario to another). After increasing the number of available berthing positions, more vessels were typically assigned to their preferred berthing positions.

7.3.5.2. Sensitivity analysis for the unit cost components

The impacts of unit cost components, which comprise handling, waiting, and late departure unit cost components, on the berth schedules are analyzed in this section of the managerial insights. Specifically, the average values of the results returned by UIMA were recorded over a total of 10 algorithmic replications adopting the problem instance P48 (i.e., the largest developed problem instance with 110 arriving vessels and 10 berthing positions). Three sensitivity analyses, mentioned above, were developed as follows: (1) variable handling unit cost component – based on Table 23, the handling unit cost component for the basic numerical experiments (i.e., sections 7.3.2 to 7.3.4) was randomly generated in the range $[50000; 70000]$ utilizing a uniform distribution (i.e., $\varphi_v^{ht} = U[50000; 70000]\ \forall v \in V$). Also, the 21 different scenarios herein were developed by changing the ranges for the random generation of handling unit cost components. The latter ranges were developed as follows: $\varphi_v^{ht1} = U[50000; 70000], \varphi_v^{ht2} = U[51000; 71000], \dots, \varphi_v^{ht21} = U[70000; 90000]\ \forall v \in V$, where the waiting and late departure unit cost components were set equal to the values generated for the basic numerical experiments; (2) variable waiting unit cost component – the handling and late departure unit cost components were kept equal to the values generated for the basic numerical experiments, while 21 different scenarios were developed by the generation of 21 different sets of waiting unit cost components: $\varphi_v^{wt1} = U[1000; 5000], \varphi_v^{wt2} = U[1200; 5200], \dots, \varphi_v^{wt21} = U[5000; 9000]\ \forall v \in V;$ (3)

variable late departure unit cost component – similar to the other two analyses, the handling and waiting unit cost components were considered equal to the values in basic numerical experiments, whereas 11 different scenarios were developed by changing the ranges for the generation of the late departure unit cost component: $\varphi_v^{lt1} = U[5000; 10000], \varphi_v^{lt2} = U[5500; 10500], \dots, \varphi_v^{lt11} = U[10000; 15000] \ \forall v \in V$.

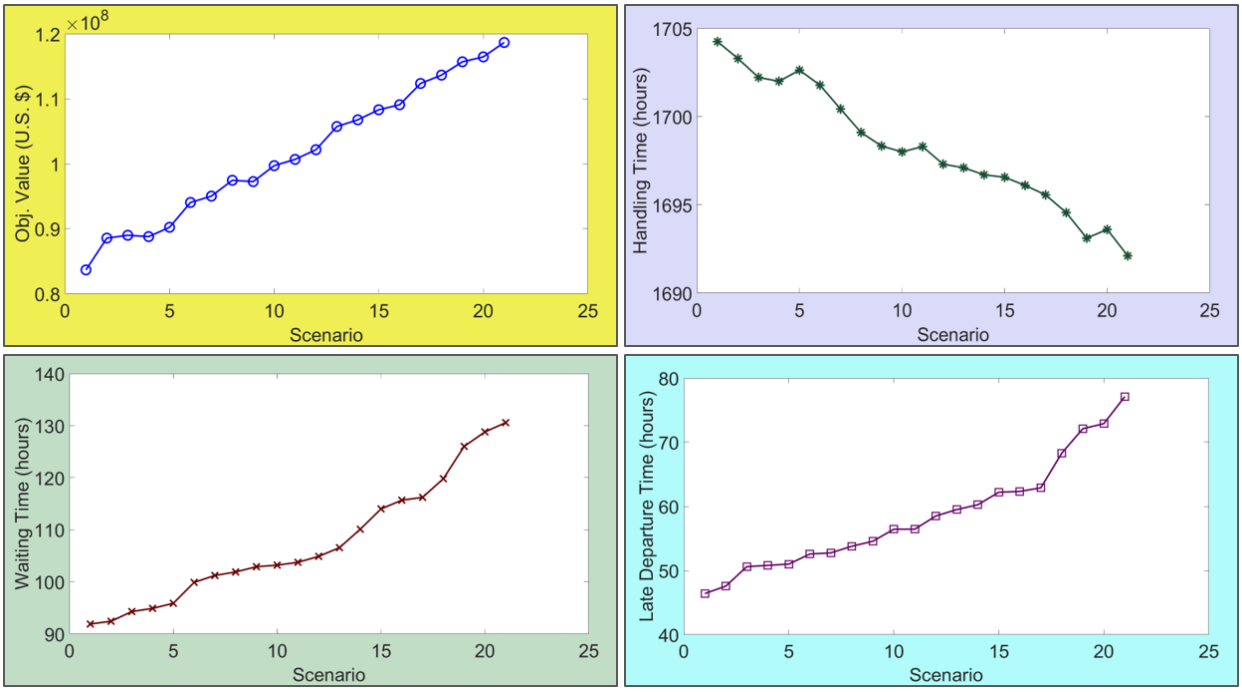

Figure 54 Sensitivity of berth schedules to the handling unit cost component.

The results of the numerical experiments, conducted for sensitivity analysis of the berth schedules to the handling time unit cost component, are illustrated in Figure 54. As it can be observed, the objective function values show an increasing pattern from scenario 1 to 21 (the total objective value increased by 41.86%) by increasing the lower bound and upper bound of the range considered for the generation of the handling time unit cost component (i.e., from $\varphi_v^{ht1} = U[50000; 70000]$ to $\varphi_v^{ht21} = U[70000; 90000]$). Increasing the handling time unit cost component obviously led to a rise in the objective function values, since the waiting time and late departure time unit cost components were kept constant. Furthermore, a decreasing tendency was recorded for handling time values over the 21 developed scenarios (see the diagram relevant to handling time in Figure 54). The latter can be justified by the fact that the UIMA algorithm tried

to decrease the handling time, which could be achieved by assignment of the arriving vessels to the preferred berthing positions, since the handling unit cost component increased from scenario 1 to 21. Increasing patterns of the diagrams related to waiting time and late departure time in Figure 54 can be supported by the fact that assignment of the vessels to their specific preferred berthing positions resulted in a longer waiting time and consequently a rise in the late departure time.

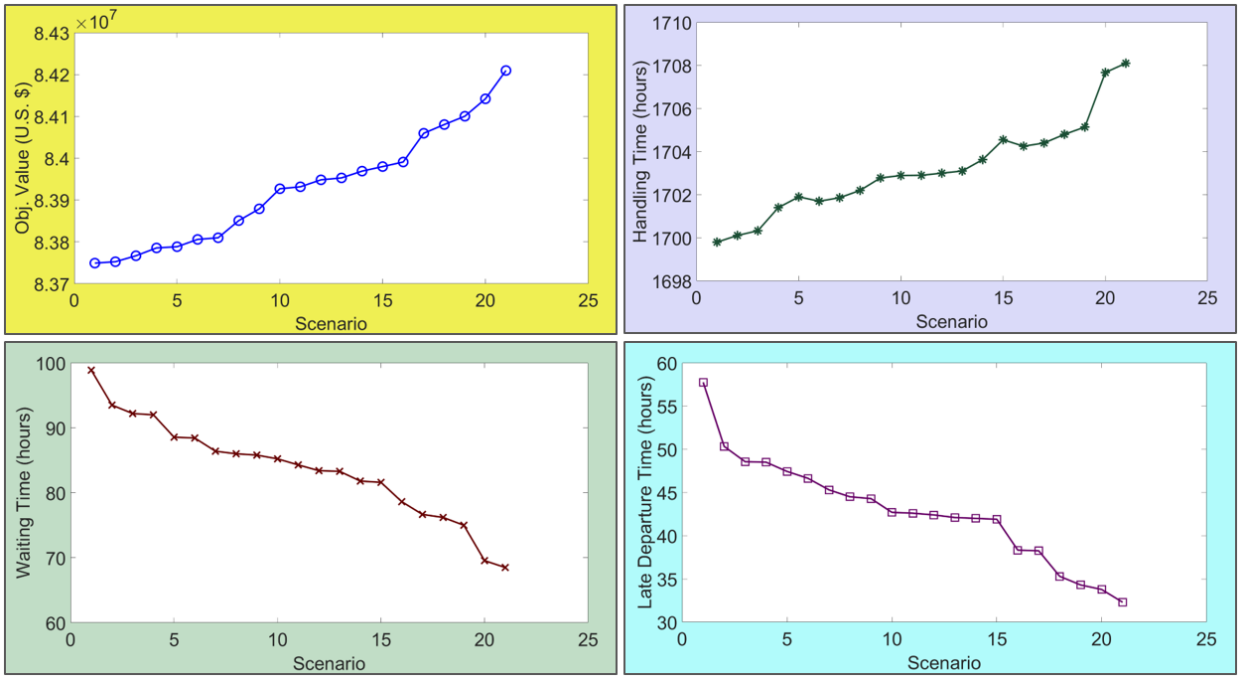

Figure 55 Sensitivity of berth schedules to the waiting unit cost component.

Figure 55 presents the results of the conducted sensitivity analysis of the berth schedules to the waiting unit cost component. As the handling time and late departure time unit cost components were constant and the waiting unit cost component increased from scenario 1 to 21, the values recorded for the objective function showed an increasing tendency (the total objective value increased by 0.55%). Furthermore, since the developed optimization model was a minimizing one and the waiting unit cost component was increasing from scenario 1 to 21, the UIMA algorithm tried to decrease the waiting time values (the decreasing pattern of the waiting time diagram can be obviously observed in Figure 55). The UIMA algorithm developed the berth schedule based on consideration of the minimum possible waiting time; hence, the arriving vessels were assigned to the first available berthing positions, no matter whether they were preferred

berthing positions for the corresponding vessels or not, which resulted in an increasing pattern for the handling time diagram in Figure 55. Moreover, the decreasing pattern of the late departure time can be justified due to the decreasing waiting time as well (see Figure 55).

The results obtained from the conducted sensitivity analysis of the berth schedules to the late departure unit cost component are illustrated in Figure 56. The developed scenarios, which included a group of increasing late departure unit cost components, led to an increasing pattern for the objective function values (the total objective value increased by 0.73%). Besides, the best solutions, returned by UIMA, tended to minimize the late departure time, since SCBSP was a mathematical model with a minimizing objective and the late departure unit cost component was increasing from scenario 1 to 11 (see the late departure diagram in Figure 56). Due to the latter fact, the berth schedule was developed in a way to assign the vessels to the first available berthing positions, which were not necessarily the preferred berthing positions. The latter not only further resulted in an increase of the handling time, but it also led to a decrease in the waiting time.

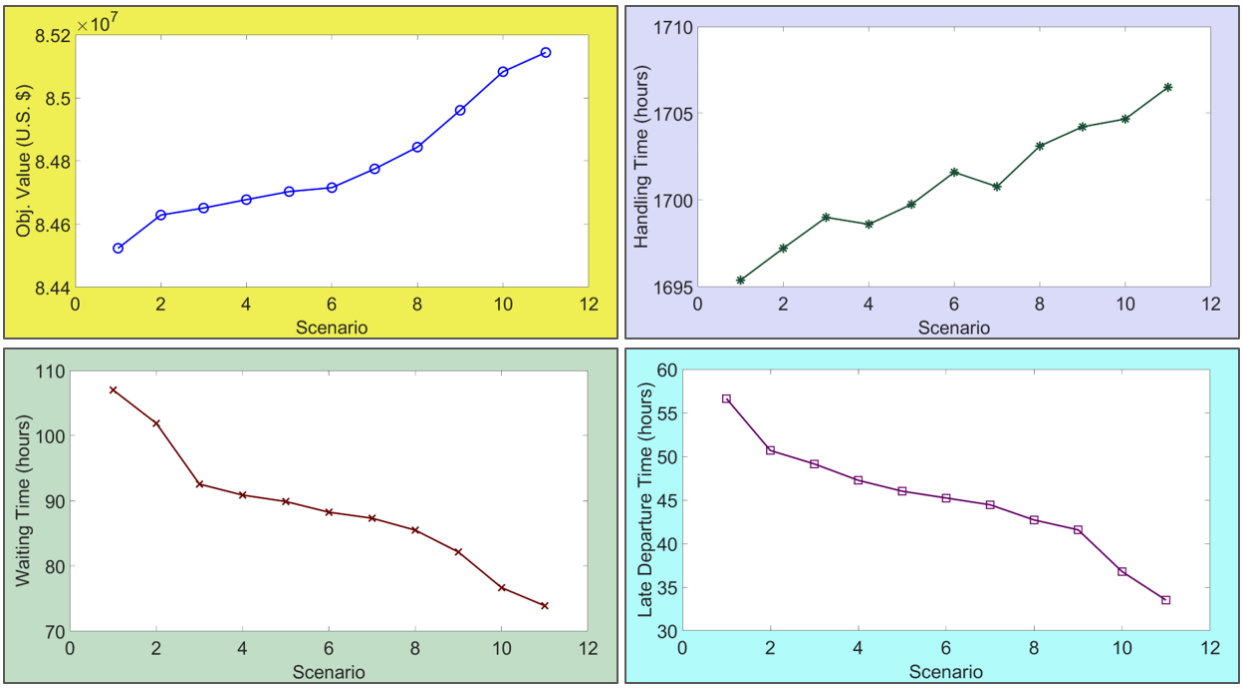

Figure 56 Sensitivity of berth schedules to the late departure unit cost component.

7.4. Conclusions for Chapter 7

A Universal Island-based Evolutionary Algorithm (UIMA) was designed in this chapter of the dissertation to solve the developed BSP mathematical model. The canonical island-based metaheuristic algorithms cover all the islands with the same algorithm, while the algorithm designed herein adopted four different metaheuristic algorithms to explore and exploit the islands throughout the search process. The migration process was conducted utilizing an adaptive strategy throughout the search process. The migrating individuals and the individuals, which should be removed to make room for the newly migrating ones, were selected adopting a roulette wheel selection operator. Comprehensive numerical experiments were conducted to evaluate performance of UIMA against seven widely-used algorithms in the BSP literature. The developed small-size problem instances were solved by CPLEX as an exact optimization algorithm. The results demonstrated that all the algorithms could solve some of the small-size problem instances to the global optimality. Nonetheless, the UIMA algorithm could solve more problem instances to the global optimality as compared to the other algorithms.

In order to assess performance of UIMA in terms of the objective value and computational time, all the developed large-size problem instances were solved by the algorithms adopted in this chapter of the dissertation. The results showed that UIMA outperformed EA, PSO, EDA, and DE by 11.03%, 12.03%, 12.92%, and 11.76%, respectively. Moreover, UIMA demonstrated a superior performance against the single-solution-based algorithms and improved the solutions returned by VNS, TS, and SA by 19.01%, 20.21%, and 19.13%, respectively. The computational time by UIMA did not exceed 5.08 minutes, which is completely justifiable from the practical standpoint. A set of paired z-tests was conducted to evaluate performance of the UIMA algorithm statistically. Specifically, the UIMA algorithm was evaluated using a z-test against each one of the algorithms, developed in this chapter of the dissertation, based on the objective function values obtained for the large-size problem instances.

The results demonstrated the statistical superiority of UIMA against all the algorithms considering a level of confidence of 5%. The stability of the algorithms was evaluated by conducting an analysis over the stochastic performance of the algorithms, where all the algorithms showed an acceptable level of stability. Moreover, the population diversity of the algorithms was

evaluated as well, where the results showed that all the algorithms kept a reasonable level of the population diversity ($\approx 0.5 \times 10^7\ U.S.\$$) towards the termination of the search process. In conclusion, the results of the conducted numerical experiments demonstrated that the developed UIMA algorithm can assist the MCT operator with berth scheduling as a decision support tool.

CHAPTER 8

CONCLUSIONS AND FUTURE RESEARCH

Since ancient times, seaborne transportation has been continuously demonstrating a remarkable impact on the global trade and the world macroeconomics. The information, presented by United Nations Conference on Trade and Development, showed that the demand for containerized transportation had a growing pattern during the last three decades, and it is predicted that this tendency will continue for the next years. Hence, the marine container terminals (MCTs) should be managed in an optimum way to make the best use of the available resources. Different types of operations are regularly conducted at MCTs, where the seaside operations and specifically berth scheduling have substantial impacts on the total efficiency of the MCTs. Therefore, this dissertation aimed to improve the berth scheduling plans at MCTs. The berth scheduling plan was modeled as a berth scheduling problem (BSP) in this dissertation. Since BSPs have NP-hard complexity, the exact optimization algorithms cannot solve the real-size BSPs in a reasonable computational time. The latter justifies the application of metaheuristic algorithms to solve BSPs in the MCT literature. Evolutionary Algorithms (EAs) are among the most popular metaheuristics utilized in the BSP literature. Determination of the algorithmic parameters is an important task, which can be implemented using the parameter tuning or parameter control strategies. Two classifications of parameter control strategies were proposed in this dissertation.

An EA with a self-adaptive parameter control strategy was developed to solve a berth scheduling problem with spatial requirements. Based on a self-adaptive parameter control strategy, the crossover and mutation probabilities were encoded in the solutions and evolved with the EA. The developed BSP was formulated as a mixed-integer linear programming model, aimed to minimize the total weighted vessel turnaround times and the total weighted vessel late departures. Performance of the Self-Adaptive EA (SAEA) was evaluated by conducting comprehensive numerical experiments. The results, obtained by SAEA, were compared against the results, provided by the alternative EAs, which relied on the different parameter selection strategies. Findings demonstrated a promising performance of all the considered solution algorithms in terms of the objective function values at termination. Moreover, the statistical and stochastic assessments

demonstrated the reliability of all the solution algorithms. However, application of the self-adaptive parameter control strategy substantially improved the objective function values at termination without a significant impact on the computational time.

The future research extensions for the developed SAEA algorithm may include:

1) evaluate performance of the developed solution algorithm against the alternative EAs using the realistic operational data, collected from the MCTs;
2) compare the developed solution algorithm against the other state-of-the-art solution algorithms;
3) develop local search heuristics to improve performance of the crossover and mutation operators;
4) evaluate different strategies for altering the crossover and mutation probabilities (e.g., use feedback from the search instead of deployment of the stochastic operators);
5) assess performance of the developed solution algorithm for different termination criteria;
6) apply the proposed SAEA for the multi-objective berth scheduling problem, where the considered objectives are conflicting in their nature;
7) assess the effects of encoding additional parameters of the SAEA (e.g., population size, penalty terms for infeasible individuals, selection operator parameters) on its performance; and
8) evaluate performance of the developed SAEA for different selection mechanisms.

Another EA with parameter control strategy was proposed in this dissertation, which relied on an augmented self-adaptive parameter control strategy. Based on an augmented self-adaptive parameter control strategy, not only the crossover and mutation probabilities were encoded in the solutions but they were also updated based on the feedback from the search. The developed discrete berth scheduling problem with dynamic vessel arrivals and spatial restrictions was solved using the designed Augmented Self-Adaptive EA (ASAEA). The considered BSP was formulated as a mixed-integer linear programming mathematical model, aiming to minimize the total waiting costs, the total handling costs, and the total late departure costs of the arriving vessels at the MCT. Nine alternative state-of-the-art metaheuristic algorithms, which have been widely utilized in the

BSP literature, were used to evaluate performance of ASAEA against them. The results demonstrated the capability of all the developed algorithms to return high-quality solutions at termination. Moreover, the computational experiments conducted over different performance indicators proved the superiority of the designed ASAEA over the alternative algorithms.

The future research extensions for the developed ASAEA algorithm may include:

1) development of the alternative heuristics to generate the initial population considering both population diversity and quality of the produced solutions;
2) consideration of new constraint handling techniques within the developed algorithm;
3) design of the alternative chromosome representations (e.g., the encoding may include additional algorithmic parameters, such as population size, parent selection parameters, offspring selection parameters);
4) development of various strategies for updating the algorithmic parameters throughout the search process;
5) evaluation of the proposed algorithm for multi-objective mathematical formulations;
6) evaluation of different berthing layouts (e.g., hybrid, continuous, channel); and
7) modeling uncertainties in different MCT operations (e.g., uncertainty in the vessel arrivals, uncertainty in the vessel handling times, potential handling equipment breakdowns).

The last solution methodology, developed in this dissertation to solve the BSP, was an island-based metaheuristic algorithm. In particular, a Universal Island-based Metaheuristic Algorithm (UIMA) was designed to solve the developed mixed-integer linear mathematical model for the BSP. The latter mathematical model aimed to minimize the total cost for providing service to the arriving vessels at the MCT. The developed UIMA algorithm included four islands, which were covered by four different types of metaheuristic algorithms, including: (1) Evolutionary Algorithm; (2) Particle Swarm Optimization; (3) Estimation of Distribution Algorithm; and (4) Differential Evolution. Comprehensive numerical experiments were implemented to assess performance of the island-based algorithm against seven widely used metaheuristic algorithms in the BSP literature. The results demonstrated the stability and the capability of the adopted

algorithms in providing high-quality solutions at termination. Nonetheless, the UIMA algorithm outperformed the other adopted algorithms based on different performance indicators.

The future research extensions for the developed UIMA algorithm may include:

1) development of new approaches for the migration criterion;
2) consideration of new selection strategies for immigrant selection and removing individuals in islands to make room for the new immigrants;
3) development of new strategies for the population initialization and assignment of sub-populations to the islands;
4) consideration of a new solution representation, which is compatible with all the adopted algorithms to skip the process of solution mapping;
5) development of new strategies for handling the infeasible solutions throughout the search process;
6) consideration of uncertainties in the MCT operations (e.g., uncertainty in the vessel arrival times, uncertainty in the vessels handling times, uncertainty in internal transport vehicles breakdown);
7) modeling different berthing layout (i.e., continuous, hybrid, channel); and
8) evaluation of the developed island-based algorithm by integrating different operations at the MCT (e.g., quay crane assignment, internal transport vehicle management, marshaling yard management).

In summary, the important role of MCTs in the global trade was investigated in this dissertation. Moreover, the routine operations at MCTs were analyzed, where it was proven that berth scheduling as a seaside operation can substantially affect the general efficiency of the MCTs. Then, a comprehensive literature review on the studies published in the field of BSPs was conducted. Three innovative solution algorithms were developed in this dissertation, including the following: (1) Self-Adaptive Evolutionary Algorithm; (2) Augmented Self-Adaptive Evolutionary Algorithm; and (3) Universal Island-based Metaheuristic Algorithm. Performance of all the three developed algorithms was evaluated against the state-of-the-art metaheuristic algorithms, which have been widely-used in the BSP literature considering a wide range of performance indicators (e.g., objective function values at termination, required computational time, convergence patterns,

population diversity, objective function stochastic variations). The results demonstrated that all the developed algorithms can assist the MCT operators with development of cost-effective berth schedules.

REFERENCES

Abioye, O.F., Dulebenets, M.A., Pasha, J. and Kavoosi, M., 2019. A vessel schedule recovery problem at the liner shipping route with Emission Control Areas. Energies, 12 (12), pp. 1-28.

Alba, E. and Tomassini, M., 2002. Parallelism and evolutionary algorithms. IEEE Transactions on Evolutionary Computation, 6(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, 42(4), pp. 2026-2035.

Alix, Y., Slack, B. and Comtois, C., 1999. Alliance or acquisition? Strategies for growth in the container shipping industry, the case of CP ships. Journal of Transport Geography, 7(3), pp. 203-208.

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, 23(2), pp. 141-155.

Ancarani, A., Di Mauro, C., Fratocchi, L., Orzes, G. and Sartor, M., 2015. Prior to reshoring: A duration analysis of foreign manufacturing ventures. International Journal of Production Economics, 169, pp. 141-155.

Anwar, S., Mohamad Zain, J., Zolkipli, M.F., Inayat, Z., Khan, S., Anthony, B. and Chang, V., 2017. From intrusion detection to an intrusion response system: fundamentals, requirements, and future directions. Algorithms, 10(2), pp. 1-23.

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, 38(3), pp. 1964-1979.

Arango, C., Cortés, P., Muñuzuri, J. and Onieva, L., 2011. Berth allocation planning in Seville inland port by simulation and optimization. Advanced Engineering Informatics, 25(3), pp. 452-461.

Ashar, A., 2013. Long-term trends in container shipping–the revised Forth Revolution. Port Technology International, 55, pp. 77-93.

Ballis, A., Golias, J. and Abarkoumkin, C., 1997. A comparison between conventional and advanced handling systems for low volume container maritime terminals. Maritime Policy & Management, 24(1), pp. 73-92.

Bierwirth, C. and Meisel, F., 2010. A survey of berth allocation and quay crane scheduling problems in container terminals. European Journal of Operational Research, 202(3), pp. 615-627.

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, 244(3), pp. 675-689.

Boile, M., Theofanis, S. and Golias, M., 2006. Berth allocation with service priorities: A linear reformulation. In Proceedings of the 5th WSEAS/IASME International Conference on System Science and Simulation in Engineering (ICOSSSE 2006), Tenerife, Canary Islands, Spain, pp. 16-18.

Bonabeau, E., Marco, D.R.D.F., Dorigo, M. and Theraulaz, G., 1999. Swarm intelligence: from natural to artificial systems (No. 1). Oxford university press.

Boussaïd, I., Lepagnot, J. and Siarry, P., 2013. A survey on optimization metaheuristics. Information Sciences, 237, pp. 82-117.

Bozorg-Haddad, O., Solgi, M. and Loáiciga, H.A., 2017. Meta-heuristic and evolutionary algorithms for engineering optimization. John Wiley & Sons.

Brester, C., Ryzhikov, I. and Semenkin, E., 2017. Multi-objective optimization algorithms with the island metaheuristic for effective project management problem solving. Organizacija, 50(4), pp. 364-373.

Brown, G.G., Cormican, K.J., Lawphongpanich, S. and Widdis, D.B., 1997. Optimizing submarine berthing with a persistence incentive. Naval Research Logistics (NRL), 44(4), pp. 301-318.

Buhrkal, K., Zuglian, S., Ropke, S., Larsen, J. and Lusby, R., 2011. Models for the discrete berth allocation problem: A computational comparison. Transportation Research Part E: Logistics and Transportation Review, 47(4), pp. 461-473.

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, 27(2-3), pp. 224-262.

Chang, D., Jiang, Z., Yan, W. and He, J., 2010. Integrating berth allocation and quay crane assignments. Transportation Research Part E: Logistics and Transportation Review, 46(6), pp. 975-990.

Chen, G., Govindan, K. and Golias, M.M., 2013. Reducing truck emissions at container terminals in a low carbon economy: proposal of a queueing-based bi-objective model for optimizing truck arrival pattern. Transportation Research Part E: Logistics and Transportation Review, 55, pp. 3-22.

Chen, S.H. and Chen, M.C., 2013. Addressing the advantages of using ensemble probabilistic models in estimation of distribution algorithms for scheduling problems. International Journal of Production Economics, 141(1), pp. 24-33.

Cheong, C.Y., Tan, K.C., Liu, D.K. and Lin, C.J., 2010. Multi-objective and prioritized berth allocation in container ports. Annals of Operations Research, 180(1), pp. 63-103.

Cirani, S., Ferrari, G. and Veltri, L., 2013. Enforcing security mechanisms in the IP-based internet of things: An algorithmic overview. Algorithms, 6(2), pp. 197-226.

Cohoon, J.P., Hegde, S.U., Martin, W.N. and Richards, D., 1987. Punctuated equilibria: a parallel genetic algorithm. Proceedings of the Second International Conference on Genetic Algorithms on Genetic Algorithms and Their Application, PP. 148-154.

Cohoon, J.P., Martin, W.N. and Richards, D.S., 1990. Genetic algorithms and punctuated equilibria in VLSI. In International Conference on Parallel Problem Solving from Nature, pp. 134-144.

Cordeau, J.F., Laporte, G., Legato, P. and Moccia, L., 2005. Models and tabu search heuristics for the berth-allocation problem. Transportation Science, 39(4), pp. 526-538.

Correcher, J.F. and Alvarez-Valdes, R., 2017. A biased random-key genetic algorithm for the time-invariant berth allocation and quay crane assignment problem. Expert Systems with Applications, 89, pp. 112-128.

Cubillos, C., Díaz, R., Urra, E., Cabrera-Paniagua, D., Cabrera, G. and Lefranc, G., 2013. An agent-based solution for the berth allocation problem. International Journal of Computers Communications & Control, 8(3), pp. 384-394.

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

Dadashi, A., Dulebenets, M.A., Golias, M.M. and Sheikholeslami, A., 2017. A novel continuous berth scheduling model at multiple marine container terminals with tidal considerations. Maritime Business Review, 2(2), pp. 142-157.

Das, A. and Chakrabarti, B.K. eds., 2005. Quantum annealing and related optimization methods (Vol. 679). Springer Science & Business Media.

De León, A.D., Lalla-Ruiz, E., Melián-Batista, B. and Moreno-Vega, J.M., 2017. A Machine Learning-based system for berth scheduling at bulk terminals. Expert Systems with Applications, 87, pp. 170-182.

De Lima, E.B., Pappa, G.L., de Almeida, J.M., Gonçalves, M.A. and Jr. Meira, W., 2010. Tuning Genetic Programming parameters with factorial designs. In Proceedings of the IEEE Congress on Evolutionary Computation, Barcelona, Spain, pp. 1–8.

De Oliveira, R.M., Mauri, G.R. and Lorena, L.A.N., 2012. Clustering search for the berth allocation problem. Expert Systems with Applications, 39(5), pp. 5499-5505.

Deep, K. and Thakur, M., 2007. A new crossover operator for real coded genetic algorithms. Applied Mathematics and Computation, 188(1), pp. 895-911.

Ding, J.F. and Liang, G.S., 2005. Using fuzzy MCDM to select partners of strategic alliances for liner shipping. Information Sciences, 173(1-3), pp. 197-225.

Du, Y., Chen, Q., Quan, X., Long, L. and Fung, R.Y., 2011. Berth allocation considering fuel consumption and vessel emissions. Transportation Research Part E: Logistics and Transportation Review, 47(6), pp. 1021-1037.

Dulebenets, M.A., 2016. Advantages and disadvantages from enforcing emission restrictions within emission control areas. Maritime Business Review, 1(2), pp. 107-132.

Dulebenets, M.A., 2017. A novel memetic algorithm with a deterministic parameter control for efficient berth scheduling at marine container terminals. Maritime Business Review, 2(4), pp. 302-330.

Dulebenets, M.A., 2018a. A diploid evolutionary algorithm for sustainable truck scheduling at a cross-docking facility. Sustainability, 10(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, 196, pp. 293-318.

Dulebenets, M.A., 2018c. Application of Evolutionary Computation for berth scheduling at marine container terminals: Parameter tuning versus parameter control. IEEE Transactions on Intelligent Transportation Systems, 19(1), pp. 25-37.

Dulebenets, M.A., 2018d. 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, 7(1), pp. 26-44.

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, 57(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, 16(1), pp. 19-32.

Dulebenets, M.A., 2018g. A comprehensive evaluation of weak and strong mutation mechanisms in Evolutionary Algorithms for truck scheduling at cross-docking terminals. IEEE Access, 6, pp. 65635-65650.

Dulebenets, M.A., 2018h. Evaluation of non-parametric selection mechanisms in evolutionary computation: A case study for the machine scheduling problem. In Nature-inspired Methods for Stochastic, Robust and Dynamic Optimization (Editors: Del Ser Lorente, J.; Osaba, E.; ISBN: 978-1-78923-329-2), IntechOpen, United Kingdom, pp. 23-45.

Dulebenets, M.A., 2019a. Minimizing the total liner shipping route service costs via application of an efficient collaborative agreement. IEEE Transactions on Intelligent Transportation Systems, 20(1), pp. 123-136.

Dulebenets, M.A., 2019b. A Delayed Start Parallel Evolutionary Algorithm for just-in-time truck scheduling at a cross-docking facility. International Journal of Production Economics, 212, pp. 236-258.

Dulebenets, M.A., 2019c. An Adaptive Island Evolutionary Algorithm for the berth scheduling problem. Memetic Computing – in press.

Dulebenets, M.A., Golias, M.M. and Mishra, S., 2018a. A collaborative agreement for berth allocation under excessive demand. Engineering Applications of Artificial Intelligence, 69, pp. 76-92.

Dulebenets, M.A., Kavoosi, M., Abioye, O. and Pasha, J., 2018b. A Self-Adaptive Evolutionary Algorithm for the berth scheduling problem: Towards efficient parameter control. Algorithms, 11(7), pp. 1-35.

Dulebenets, M.A., Moses, R., Ozguven, E.E. and Vanli, A., 2017a. Minimizing carbon dioxide emissions due to container handling at marine container terminals via hybrid evolutionary algorithms. IEEE Access, 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, pp. 1-18.

Dulebenets, M.A., Rahimi, A.M. and Mazaheri, A., 2017b. Assessing the effects of indirect left turn on a signalized intersection performance: A case study for the Tehran Metropolitan Area. Open Journal of Applied Sciences, 7(11), pp. 617-634.

Eberhart, R.C. and Shi, Y., 2001. Tracking and optimizing dynamic systems with particle swarms. In Proceedings of the 2001 Congress on Evolutionary Computation (IEEE Cat. No. 01TH8546), Vol. 1, pp. 94-100.

Eiben, A.E. and Smith, J.E., 2003. Introduction to evolutionary computing. Springer.

Eiben, A.E. and Smith, J.E., 2015. Introduction to evolutionary computing. Springer.

Elbeltagi, E., Hegazy, T. and Grierson, D., 2005. Comparison among five evolutionary-based optimization algorithms. Advanced Engineering Informatics, 19(1), pp. 43-53.

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, 67(4), pp. 551-563.

Eniram. Evolution of Shipping. Available online: https://www.eniram.fi/evolution-of-shipping/ (accessed on 06 March 2019).

Esmer, S., Cetin, I.B. and Tuna, O., 2010. A simulation for optimum terminal truck number in a Turkish port based on lean and green concept. The Asian Journal of Shipping and Logistics, 26(2), pp. 277-296.

Fransoo, J.C. and Lee, C.Y., 2013. The critical role of ocean container transport in global supply chain performance. Production and Operations Management, 22(2), pp. 253-268.

Frojan, P., Correcher, J.F., Alvarez-Valdes, R., Koulouris, G. and Tamarit, J.M., 2015. The continuous Berth Allocation Problem in a container terminal with multiple quays. Expert Systems with Applications, 42(21), pp. 7356-7366.

GAMS. Cutting Edge Modeling. Available online: https://www.gams.com/ (accessed on 3 May 2018).

Garcıa, J., Berlanga, A. and Molina, J.M., 2007. Evolutionary algorithms in multiply-specified engineering. The MOEAs and WCES strategies. Advanced Engineering Informatics, 21, pp. 3-21.

Ghoumari, A., Nakib, A. and Siarry, P., 2018. Evolutionary algorithm with ensemble strategies based on maximum a posteriori for continuous optimization. Information Sciences, 460, pp. 1-22.

Glover, F., 1986. Future paths for integer programming and links to artificial intelligence. Computers & Operations Research, 13(5), pp. 533-549.

Golias, M., Boile, M. and Theofanis, S., 2006. The berth allocation problem: a formulation reflecting time window service deadlines, Submitted for Presentation at the 48th Transportation Research Forum Annual Meeting, Boston, Massachusetts, and Publication in the Conference Proceedings, pp. 1-21.

Golias, M., Boile, M., Theofanis, S. and Efstathiou, C., 2010. The berth-scheduling problem: Maximizing berth productivity and minimizing fuel consumption and emissions production. Transportation Research Record: Journal of the Transportation Research Board, (2166), pp. 20-27.

Golias, M., Portal, I., Konur, D., Kaisar, E. and Kolomvos, G., 2014. Robust berth scheduling at marine container terminals via hierarchical optimization. Computers & Operations Research, 41, pp. 412-422.

Golias, M.M. and Haralambides, H.E., 2011. Berth scheduling with variable cost functions. Maritime Economics & Logistics, 13(2), pp. 174-189.

Golias, M.M., Boile, M. and Theofanis, S., 2009. Berth scheduling by customer service differentiation: A multi-objective approach. Transportation Research Part E: Logistics and Transportation Review, 45(6), pp. 878-892.

Golias, M.M., Boile, M. and Theofanis, S., 2010. A lamda-optimal based heuristic for the berth scheduling problem. Transportation Research Part C: Emerging Technologies, 18(5), pp. 794-806.

Golias, M.M., Boile, M. and Theofanis, S., 2010. Discrete berth-scheduling problem: toward a unified mathematical formulation. Transportation Research Record, 2168(1), pp. 1-8.

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

Grefenstette, J.J., 1981. Parallel Adaptive Algorithms for Function Optimization: (preliminary Report). Computer Science Department, Vanderbilt University.

Guerrero, M., Montoya, F.G., Baños, R., Alcayde, A. and Gil, C., 2018. Community detection in national-scale high voltage transmission networks using genetic algorithms. Advanced Engineering Informatics, 38, pp. 232-241.

Hansen, P. and Oguz, C., 2003. A note on formulations of the static and dynamic berth allocation problems. Groupe d'études et de recherche en analyse des décisions, HEC Montréal.

Hansen, P., Mladenović, N. and Pérez, J.A.M., 2010. Variable neighborhood search: methods and applications. Annals of Operations Research, 175(1), pp. 367-407.

Hansen, P., Oğuz, C. and Mladenović, N., 2008. Variable neighborhood search for minimum cost berth allocation. European Journal of Operational Research, 191(3), pp. 636-649.

Hao, X., Lin, L., Gen, M. and Ohno, K., 2013. Effective estimation of distribution algorithm for stochastic job shop scheduling problem. Procedia Computer Science, 20, pp. 102-107.

He, J., Huang, Y. and Yan, W., 2015a. Yard crane scheduling in a container terminal for the trade-off between efficiency and energy consumption. Advanced Engineering Informatics, 29(1), pp. 59-75.

He, J., Huang, Y., Yan, W. and Wang, S., 2015b. Integrated internal truck, yard crane and quay crane scheduling in a container terminal considering energy consumption. Expert Systems with Applications, 42(5), pp. 2464-2487.

He, J., Zhang, W., Huang, Y. and Yan, W., 2013. A simulation optimization method for internal trucks sharing assignment among multiple container terminals. Advanced Engineering Informatics, 27(4), pp. 598-614.

Hendriks, M., Laumanns, M., Lefeber, E. and Udding, J.T., 2010. Robust cyclic berth planning of container vessels. OR Spectrum, 32(3), pp. 501-517.

Henesey, L., 2004. Enhancing container terminal performance: a multi agent systems approach. Doctoral dissertation, Blekinge Institute of Technology.

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, 27, pp. 156-168.

Hu, Z.H., 2015. Multi-objective genetic algorithm for berth allocation problem considering daytime preference. Computers & Industrial Engineering, 89, pp. 2-14.

Imai, A., Nishimura, E. and Papadimitriou, S., 2001. The dynamic berth allocation problem for a container port. Transportation Research Part B: Methodological, 35(4), pp. 401-417.

Imai, A., Nishimura, E. and Papadimitriou, S., 2003. Berth allocation with service priority. Transportation Research Part B: Methodological, 37(5), pp. 437-457.

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, 44(1), pp. 136-151.

Imai, A., Nishimura, E. and Papadimitriou, S., 2013. Marine container terminal configurations for efficient handling of mega-containerships. Transportation Research Part E: Logistics and Transportation Review, 49(1), pp. 141-158.

Imai, A., Yamakawa, Y. and Huang, K., 2014. The strategic berth template problem. Transportation Research Part E: Logistics and Transportation Review, 72, pp. 77-100.

Imai, A., Zhang, J.T., Nishimura, E. and Papadimitriou, S., 2007. The berth allocation problem with service time and delay time objectives. Maritime Economics & Logistics, 9(4), pp. 269-290.

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, 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, 2018, pp. 1-20.

Karafa, J., Golias, M.M., Ivey, S., Saharidis, G.K. and Leonardos, N., 2013. The berth allocation problem with stochastic vessel handling times. The International Journal of Advanced Manufacturing Technology, 65(1-4), pp. 473-484.

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 – in press.

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.

Kordić, S., Davidović, T., Kovač, N. and Dragović, B., 2016. Combinatorial approach to exactly solving discrete and hybrid berth allocation problem. Applied Mathematical Modelling, 40(21-22), pp. 8952-8973.

Kurdi, M., 2015. A new hybrid island model genetic algorithm for job shop scheduling problem. Computers & Industrial Engineering, 88, pp. 273-283.

Lalla-Ruiz, E., Expósito-Izquierdo, C., Melián-Batista, B. and Moreno-Vega, J.M., 2016. A set-partitioning-based model for the berth allocation problem under time-dependent limitations. European Journal of Operational Research, 250(3), pp. 1001-1012.

Lalla-Ruiz, E., MeliáN-Batista, B. and Moreno-Vega, J.M., 2012. Artificial intelligence hybrid heuristic based on tabu search for the dynamic berth allocation problem. Engineering Applications of Artificial Intelligence, 25(6), pp. 1132-1141.

Lalla-Ruiz, E., Voß, S., Expósito-Izquierdo, C., Melián-Batista, B. and Moreno-Vega, J.M., 2017. A POPMUSIC-based approach for the berth allocation problem under time-dependent limitations. Annals of Operations Research, 253(2), pp. 871-897.

Lang, N. and Veenstra, A., 2010. A quantitative analysis of container vessel arrival planning strategies. OR spectrum, 32(3), pp. 477-499.

Lardeux, F. and Goëffon, A., 2010, December. 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 & Business Media.

Lee, C.Y. and Song, D.P., 2017. Ocean container transport in global supply chains: Overview and research opportunities. Transportation Research Part B: Methodological, 95, pp. 442-474.

Legato, P., Mazza, R.M. and Gullì, D., 2014. Integrating tactical and operational berth allocation decisions via simulation–optimization. Computers & Industrial Engineering, 78, pp. 84-94.

Lei, L., Fan, C., Boile, M. and Theofanis, S., 2008. Collaborative vs. non-collaborative container-vessel scheduling. Transportation Research Part E: Logistics and Transportation Review, 44(3), pp. 504-520.

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, 28(11), pp. 3163-3182.

Li, S., Tangb, L. and Liuc, J., 2005. An exact method for berth allocation at raw material docks. International Federation of Automatic Control, pp. 1-6.

Liang, C., Huang, Y. and Yang, Y., 2009. A quay crane dynamic scheduling problem by hybrid evolutionary algorithm for berth allocation planning. Computers & Industrial Engineering, 56(3), pp. 1021-1028.

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, 33(06), pp. 1-28.

Lohmann, R., 1990, October. Application of evolution strategy in parallel populations. In International Conference on Parallel Problem Solving from Nature. Springer, Berlin, Heidelberg, pp. 198-208.

Lu, Z., Han, X. and Xi, L., 2011. Simultaneous berth and quay crane allocation problem in container terminal. Advanced Science Letters, 4(6-7), pp. 2113-2118.

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, pp. 1-20.

Martin, W.N., Lienig, J. and Cohoon, J.P., 1997. Island (migration) models: evolutionary algorithms based on punctuated equilibria. Handbook of Evolutionary Computation, C6.3, pp. 101-124.

Mathworks. Release 2016a. Available online: https://www.mathworks.com/products/new_products/release2016a.html (accessed on 3 May 2018).

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 & Operations Research, 70, pp. 140-154.

McCalla, R.J., 1999. Global change, local pain: intermodal seaport terminals and their service areas. Journal of Transport Geography, 7(4), pp. 247-254.

Meng, Q., Wang, S., Andersson, H. and Thun, K., 2013. Containership routing and scheduling in liner shipping: overview and future research directions. Transportation Science, 48(2), pp. 265-280.

Meng, Q., Wu, J., Ellis, J. and Kennedy, P.J., 2017, May. Dynamic island model based on spectral clustering in genetic algorithm. In 2017 International Joint Conference on Neural Networks (IJCNN), pp. 1724-1731.

Moorthy, R. and Teo, C.P., 2007. Berth management in container terminal: the template design problem. Container Terminals and Cargo Systems, pp. 63-86.

Mourão, M.C., Pato, M.V. and Paixão, A.C., 2002. Ship assignment with hub and spoke constraints. Maritime Policy & Management, 29(2), pp. 135-150.

Nishimura, E., Imai, A. and Papadimitriou, S., 2001. Berth allocation planning in the public berth system by genetic algorithms. European Journal of Operational Research, 131(2), pp. 282-292.

Norstad, I., Fagerholt, K. and Laporte, G., 2011. Tramp ship routing and scheduling with speed optimization. Transportation Research Part C: Emerging Technologies, 19(5), pp. 853-865.

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, 14(11), pp. 815-821.

Paixão, A.C. and Bernard Marlow, P., 2003. Fourth generation ports–a question of agility?. International Journal of Physical Distribution & Logistics Management, 33(4), pp. 355-376.

Pan, J., Xu, Y. and Zhang, G., 2018. Online integrated allocation of berths and quay cranes in container terminals with 1-lookahead. Journal of Combinatorial Optimization, 36(2), pp. 617-636.

Panayides, P.M. and Wiedmer, R., 2011. Strategic alliances in container liner shipping. Research in Transportation Economics, 32(1), pp. 25-38.

Pelusi, D., Mascella, R., Tallini, L., Nayak, J., Naik, B. and Abraham, A., 2018. Neural network and fuzzy system for the tuning of Gravitational Search Algorithm parameters. Expert Systems with Applications, 102, pp. 234-244.

Peng, J., Zhou, Z. and Li, R., 2015. A collaborative berth allocation problem with multiple ports based on genetic algorithm. Journal of Coastal Research, 73(1), pp. 290-297.

Pinedo, M.L., 2016. Scheduling: theory, algorithms, and systems. Springer.

Pratap, S., Nayak, A., Kumar, A., Cheikhrouhou, N. and Tiwari, M.K., 2017. An integrated decision support system for berth and ship unloader allocation in bulk material handling port. Computers & Industrial Engineering, 106, pp. 386-399.

Rashidi, H. and Tsang, E.P., 2013. Novel constraints satisfaction models for optimization problems in container terminals. Applied Mathematical Modelling, 37(6), pp. 3601-3634.

Ribeiro, G.M., Mauri, G.R., de Castro Beluco, S., Lorena, L.A.N. and Laporte, G., 2016. Berth allocation in an ore terminal with demurrage, despatch and maintenance. Computers & Industrial Engineering, 96, pp. 8-15.

Rodrigue, J.P., Comtois, C. and Slack, B., 2009. The geography of transport systems. Routledge.

Rodriguez-Molins, M., Ingolotti, L., Barber, F., Salido, M.A., Sierra, M.R. and Puente, J., 2014. A genetic algorithm for robust berth allocation and quay crane assignment. Progress in Artificial Intelligence, 2(4), pp. 177-192.

Saharidis, G.K.D., Golias, M.M., Boile, M., Theofanis, S. and Ierapetritou, M.G., 2010. The berth scheduling problem with customer differentiation: a new methodological approach based on hierarchical optimization. The International Journal of Advanced Manufacturing Technology, 46(1-4), pp. 377-393.

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

Santos, Á., Gomes, A. and Mendes, A.J., 2010. Integrating new technologies and existing tools to promote programming learning. Algorithms, 3(2), pp. 183-196.

Schepler, X., Absi, N., Feillet, D. and Sanlaville, E., 2019. The stochastic discrete berth allocation problem. EURO Journal on Transportation and Logistics, pp. 1-34.

Sedgewick, R., 1998. Algorithms in C: Fundamentals, Data Structures, Sorting, Searching, Parts 1-4, 3rd ed.; Pearson Education.

Segura, C., Segredo, E. and León, C., 2011, July. Parallel island-based multiobjectivised memetic algorithms for a 2D packing problem. In Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation, pp. 1611-1618.

Sheikholeslami, A., Ilati, G. and Yeganeh, Y.E., 2013. Practical solutions for reducing container ships' waiting times at ports using simulation model. Journal of Marine Science and Application, 12(4), pp. 434-444.

Sivanandam, S.N., Deepa, S., 2008. Introduction to Genetic Algorithms, 1st ed.; Springer-Verlag Berlin Heidelberg.

Song, L., Cherrett, T. and Guan, W., 2012. Study on berth planning problem in a container seaport: Using an integrated programming approach. Computers & Industrial Engineering, 62(1), pp. 119-128.

Srinivas, M. and Patnaik, L.M., 1994. Adaptive probabilities of crossover and mutation in genetic algorithms. IEEE Transactions on Systems, Man, and Cybernetics, 24(4), pp. 656-667.

Storn, R. and Price, K., 1997. Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. Journal of Global Optimization, 11(4), pp. 341-359.

Sun, D., 2012. Optimization of Berth allocations in container terminals. Doctoral dissertation, The University of Hong Kong.

Tanese, R., 1987. Parallel genetic algorithm for a hypercube. In Genetic algorithms and their applications: proceedings of the second International Conference on Genetic Algorithms, Massachusetts Institute of Technology, Cambridge, MA, pp. 177-183.

Ting, C.J., Wu, K.C. and Chou, H., 2014. Particle swarm optimization algorithm for the berth allocation problem. Expert Systems with Applications, 41(4), pp. 1543-1550.

Tong, S. and Zhao, J., 2011. A Time-Space Diagram for Solving Dynamic Berth Allocation Problem at Container Terminal. In ICTE 2011, pp. 2316-2321.

Tsai, A.H., Lee, C.N., Wu, J.S. and Chang, F.S., 2016, October. A Novel Wharf-based Genetic Algorithm for Berth Allocation Planning. Soft Computing, 21, pp. 2897-2910.

Umang, N., Bierlaire, M. and Erera, A.L., 2017. Real-time management of berth allocation with stochastic arrival and handling times. Journal of Scheduling, 20(1), pp. 67-83.

UNCTAD, 2017. Review of maritime transport 2017, United Nations Conference on Trade and Development, New York, NY, Geneva. Available : http://unctad.org/en/PublicationsLibrary/rmt2017_en.pdf.

Ursavas, E. and Zhu, S.X., 2016. Optimal policies for the berth allocation problem under stochastic nature. European Journal of Operational Research, 255(2), pp. 380-387.

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, 54, pp. 142-159.

Wang, R., Nguyen, T.T., Li, C., Jenkinson, I., Yang, Z. and Kavakeb, S., 2019. Optimising discrete dynamic berth allocations in seaports using a Levy Flight based meta-heuristic. Swarm and Evolutionary Computation, 44, pp. 1003-1017.

Wang, S., Alharbi, A. and Davy, P., 2014. Liner ship route schedule design with port time windows. Transportation Research Part C: Emerging Technologies, 41, pp. 1-17.

Wang, S., Meng, Q. and Liu, Z., 2013. A note on "Berth allocation considering fuel consumption and vessel emissions". Transportation Research Part E: Logistics and Transportation Review, 49(1), pp. 48-54.

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, 25, pp. 576-582.

Wang, X., Xing, K., Li, X. and Luo, J., 2018. An estimation of distribution algorithm for scheduling problem of flexible manufacturing systems using Petri nets. Applied Mathematical Modelling, 55, pp. 776-788.

Xiang, X., Liu, C. and Miao, L., 2018. Reactive strategy for discrete berth allocation and quay crane assignment problems under uncertainty. Computers & Industrial Engineering, 126, pp. 196-216.

Xin Cao, J., Ren, J., Mandula and Ting Shang, X., 2016. A Genetic Algorithm for Integrated Berth Allocation and Quay Crane Assignment Problems in Container Terminals. In CICTP 2016, pp. 543-554.

Xu, D., Li, C.L. and Leung, J.Y.T., 2012. Berth allocation with time-dependent physical limitations on vessels. European Journal of Operational Research, 216(1), pp. 47-56.

Xu, Y., Chen, Q. and Quan, X., 2012. Robust berth scheduling with uncertain vessel delay and handling time. Annals of Operations Research, 192(1), pp. 123-140.

Yang, C., Wang, X. and Li, Z., 2012. An optimization approach for coupling problem of berth allocation and quay crane assignment in container terminal. Computers & Industrial Engineering, 63(1), pp. 243-253.

Yang, D., Liu, M. and Shi, X., 2011. Verifying liner shipping alliance's stability by applying core theory. Research in Transportation Economics, 32(1), pp. 15-24.
Yang, X.S., 2014. Nature-inspired optimization algorithms. Elsevier.

Zhen, L. and Chang, D.F., 2012. A bi-objective model for robust berth allocation scheduling. Computers & Industrial Engineering, 63(1), pp. 262-273.

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, 106, pp. 193-217.

Zhou, P., Wang, K., Kang, H. and Jia, J., 2006, January. Study on berth allocation problem with variable service priority in multi-user container terminal. In the Sixteenth International Offshore and Polar Engineering Conference. International Society of Offshore and Polar Engineers, 4, pp. 684–688.

BIOGRAPHICAL SKETCH