https://dx.doi.org/10.1016/j.trb.2020.10.007

A Simulation Optimization Method for Deep-Sea Vessel Berth Planning and Feeder Arrival Scheduling at a Container Port

Shuai Jia
Institute of Big Data Intelligent Management and Decision,
College of Management,
Shenzhen University,
Shenzhen, 518061, China,
and Institute of Operations Research and Analytics,
National University of Singapore,
Singapore 117602
shuai.jia@connect.polyu.hk

Chung-Lun Li
Department of Logistics and Maritime Studies,
The Hong Kong Polytechnic University,
Hung Hom, Kowloon, Hong Kong
chung-lun.li@polyu.edu.hk

Zhou Xu*
Department of Logistics and Maritime Studies,
The Hong Kong Polytechnic University,
Hung Hom, Kowloon, Hong Kong
zhou.xu@polyu.edu.hk

October 30, 2019
Revised March 30, 2020
Revised August 21, 2020

*Corresponding author

Abstract

Vessels served by a container port can usually be classified into two types: deep-sea vessels and feeders. While the arrival times and service times of deep-sea vessels are known to the port operator when berth plans are being devised, the service times of feeders are usually uncertain due to lack of data interchange between the port operator and the feeder operators. The uncertainty of feeder service times can incur long waiting lines and severe port congestion if the service plans for deep-sea vessels and feeders are poorly devised. This paper studies the problem of how to allocate berths to deep-sea vessels and schedule arrivals of feeders for congestion mitigation at a container port where the number of feeders to be served is significantly larger than the number of deep-sea vessels, and where the service times of feeders are uncertain. We develop a stochastic optimization model that determines the berth plans of deep-sea vessels and arrival schedules of feeders, so as to minimize the departure delays of deep-sea vessels and schedule displacements of feeders. The model controls port congestion through restricting the expected queue length of feeders. We develop a three-phase simulation optimization method to solve this problem. Our method comprises a global phase, a local phase, and a clean-up phase, where the simulation budget is wisely allocated to the solutions explored in different phases so that a locally optimal solution can be identified with a reasonable amount of computation effort. We evaluate the performance of the simulation optimization method using test instances generated based on the operational data of a container port in Shanghai.

Keywords: berth allocation; port operations; service time uncertainty; congestion mitigation; simulation optimization

1 Introduction

Vessels served by a container port can usually be classified into two types: deep-sea vessels and feeders (Cordeau et al. 2005; Emde and Boysen 2016; Ursavas and Zhu 2016). Deep-sea vessels are large in size, and they transport containers between ports along long-haul ocean routes. Feeders are small-sized vessels, and they provide transportation services between ports that are relatively close to each other. For container ports located on the estuaries of busy waterways, such as the Port of Shanghai, which is located on the Yangtze River estuary, feeders play an important role in container transshipment. In these container ports, the number of feeders served can be significantly larger than the number of deep-sea vessels.

One important issue faced by a port operator is the need for effective berth planning for both deep-sea vessels and feeders. Deep-sea vessels visit container ports regularly by following their voyage schedules. Thus, the scheduled port arrival and departure times of each deep-sea vessel are known to the port operator. Using electronic data interchange, the port operator can acquire the throughput information of deep-sea vessels. The scheduled arrival and departure times and the throughput information are essential for making detailed service plans for the vessels. However, feeders are usually operated by small companies that do not have advanced information systems, and there is a lack of data interchange between the port operator and the feeder operators. As a result, accurate throughput information of feeders may not be available to the port operator in advance. This poses a great challenge for berth planning, as service plans are made several days before vessels arrive. Because of the uncertainties in service times, port operators do not usually reserve berth space for feeders when making berth plans, but allocate berths to feeders dynamically according to certain predetermined service rules (e.g., arbitrarily assigning an available berth to a feeder when it arrives at the port). However, this practice may lead to severe congestion and also lower the port's operational efficiency. On the other hand, developing berth plans for deep-sea vessels and simultaneously taking into account the uncertainties of feeder service times may alleviate congestion and thus improve the efficiency of the port.

For a container port such as the Port of Shanghai, which serves a large number of feeders, another important issue faced by the port operator is the need to mitigate port congestion by controlling the waiting lines of feeders. Unlike deep-sea vessels that wait at the anchorage ground upon arrival, feeders usually wait at the terminal basin, which is close to the berths. Long waiting lines of feeders obstruct the traffic in the port, impeding the service of vessels and increasing the risk of vessel collisions. One method that has been implemented in practice for congestion mitigation

is to serve feeders by appointment. That is, the port operator makes adjustments to the voyage schedules of feeders by assigning updated arrival times to the feeders, and the feeders are required to arrive at the port at or close to their assigned arrival times. Assigning arrival times to feeders allows the port operator to reduce the number of arrivals during peak hours. However, the queue length of feeders depends not only on the feeder arrival times, but also on their service times. Hence, to effectively control congestion in the port, the assignment of arrival times to feeders should also take into account the service time uncertainties of feeders.

In this paper, we study a problem that allocates berths to deep-sea vessels and assigns arrival times to feeders for a container port where the service times of feeders are stochastic with known probability distribution. We develop a stochastic optimization model for the problem. The model aims to minimize the departure delays of deep-sea vessels and schedule displacements of feeders, subject to berth availability and a queue length limit. We develop a three-phase simulation optimization method for solving this model. Our model and solution method can be used for berth planning at a container port where information on feeders is limited and mitigating congestion is of great importance.

Berth allocation problems have attracted tremendous research efforts over the past two decades. Various models have been developed by researchers for decision-making at tactical and operational levels; see Steenken et al. (2004), Bierwirth and Meisel (2010, 2015), Kim and Lee (2015), and Gharehgozli et al. (2016) for comprehensive reviews of the literature. Existing berth allocation problems typically allocate berths to vessels with given arrival schedules of vessels. The problem studied in this paper extends the traditional berth allocation problems by allocating berths to deep-sea vessels and scheduling the arrivals of feeders. There exist some works that study the scheduling of vessel arrivals from the perspective of a port operator or a shipping line. Golias et al. (2009) assign arrival times to vessels in a berth allocation model to minimize weighted total waiting time and departure delay of vessels. Alvarez et al. (2010) and Lang and Veenstra (2010) use simulation to evaluate different arrival scheduling strategies for minimizing the fuel consumption cost of vessels. Lee and Jin (2013) consider a feeder vessel management problem in which one of the decisions is to assign feeders to different working shifts. Li and Pang (2011), Pang et al. (2011), and Pang and Liu (2014) analyze different integrated berth allocation and vessel routing problems that require the determination of vessels' arrival times at different ports. Du et al. (2015) assign arrival times to vessels in a tidal port to reduce vessel waiting times caused by tidal effect in the navigation channels. Li and Lam (2017) schedule the arrivals of vessels at a container port to

resolve vessel conflicts in the fairways nearby the port. Venturini et al. (2017) and De et al. (2020) schedule the berth utilization and the vessel arrivals to minimize the cost incurred by vessel fuel consumption and terminal operations. In our model, the arrival times of deep-sea vessels are input parameters, whereas the arrival times of feeders are decision variables. This model configuration is motivated by the following facts: (i) Deep-sea vessels visit multiple ports by following their voyage schedules; changing the arrival time of a deep-sea vessel at a port may have a significant impact on its schedule for visiting other ports, which is undesirable in practice. (ii) The voyage schedules of feeders are generally quite flexible, as feeders typically transport containers between only two ports. Thus, scheduling the arrivals of feeders helps alleviate port congestion without incurring unacceptable compromises in service quality.

Another important feature of our model is the stochasticity of the service times of feeders. Many researchers have studied berth allocation problems with uncertain vessel information. Some berth allocation models consider uncertain vessel arrival times (see, e.g., Moorthy and Teo 2006), while some models consider uncertain vessel service times (see, e.g., Golias 2011 and Shang et al. 2016). Some berth allocation models take into account uncertainties in both vessel arrivals and vessel services; see, for example, Zhen et al. (2011), Xu et al. (2012), Umang et al. (2017), Xiang et al. (2017, 2018), and Iris and Lam (2019). Liu et al. (2020) provide a summary of the existing works on stochastic berth allocation problems. In most of the existing works, berths are explicitly allocated to vessels in the hope that vessels can start being served as planned at their designated berths even if the vessel information is uncertain. Our model differs from these models in that the berths for serving deep-sea vessels are determined explicitly with deterministic arrival and service times, whereas the berths for serving feeders are modeled implicitly due to the uncertainties of feeder service times. For evaluating the performance of the service plan generated by our model, we adopt an operational service rule that dynamically allocates berths to feeders when the service times of feeders are observed.

In addition to the limited berth availability, which is the major constraint considered in the berth allocation literature, our model considers the queueing behavior of feeders and imposes a restriction on the expected queue length of feeders. This additional constraint is motivated by the need to control vessel congestion in the terminal basin of a busy container port. The queueing behavior of vessels has been studied using queueing and simulation models. These include a study that models tug services in harbor terminals as queueing systems (Easa 1987), studies that model the arrival and service processes of vessels as queueing systems and evaluate the port performances under

different service patterns of vessels (see, e.g., Radmilovich 1992; Zrnić et al. 1999), and studies that apply simulation models to evaluate the performance of quay crane and berth allocation policies (see, e.g., Legato and Mazza 2001; Dragović et al. 2005, 2006). Existing models on vessel queueing behavior typically focus on the evaluation of predetermined vessel service policies. Our model, however, incorporates vessel queueing behavior into berth allocation and feeder arrival scheduling decisions, so as to optimize berth utilization while at the same time keeping vessel congestion under control.

We develop a simulation optimization method for solving the problem studied in this paper. Simulation optimization methods are widely used for solving optimization problems where performance of each solution is evaluated by doing simulation experiments. Xu et al. (2015) and Amaran et al. (2016) review various simulation optimization methods and their applications. As noted by Lee et al. (2006), performing a large number of simulation replications to obtain an accurate estimation of solution performance would consume an unaffordable amount of computation effort, especially when the solution space is large, and thus one should seek to balance the effort spent in running simulations and in sampling solutions. In the literature of berth allocation, there exist a few works that apply simulation optimization for solving problems with uncertainties. However, existing works either allocate a large simulation budget to each solution and incur a heavy computation burden (see, e.g., Han et al. 2010), or attempt to explore the solution space efficiently by ignoring uncertainties and then use simulation to evaluate only a limited number of the visited solutions (see, e.g., Arango et al. 2011, 2013; Legato et al. 2014). Unlike these simulation optimization methods, we develop a three-phase simulation optimization method, which consists of a global phase, a local phase, and a clean-up phase, where different amounts of simulation budget are allocated to different phases so that a locally optimal solution can be identified with a reasonable amount of computation effort. Our method is an adaptation of the method proposed by Xu et al. (2010) for solving discrete simulation optimization problems. Since Xu et al.'s method samples solutions directly from a given set of candidate solutions, their method can only handle problems with relatively small solution spaces and without any complicated constraint. Our method extends Xu et al.'s method by introducing constraint relaxation techniques and new solution sampling strategies to tackle the berth allocation and arrival scheduling problem, which has a large solution space and contains complicated berth capacity constraints and queue length constraints. To the best of our knowledge, this is the first work demonstrating the effectiveness of such a method on a large-scale constrained discrete simulation optimization problem.

Our main contributions are twofold. First, we study a new berth allocation problem that arises in real-life port operations. Our problem differs from the berth allocation problems studied in the literature in the following aspects: (i) In our problem, deep-sea vessels' information is known to the port operator before the vessels arrive, while the feeders' service times are available only when the feeders have arrived the port. Thus, berth space is allocated to deep-sea vessels and feeders in a two-stage fashion, with only the probability distributions of the feeder service times being available at the planning stage. This characteristic is unique and has not been considered by other berth allocation research. (ii) Because feeders' service times are uncertain at the planning stage, the berth plan for deep-sea vessels needs to be developed by taking into consideration the potential congestion that the random arrivals of feeders may cause during execution. We model this by imposing an upper limit on the expected queue length of feeders, so as to mitigate port congestion. This is a new feature that has not been considered in existing berth allocation models. Second, we apply a three-phase simulation optimization method for solving the stochastic optimization model. We develop new solution methods for the global and local phases, so that our method can solve the stochastic optimization problem with a large number of decision variables. In addition, our method strikes a balance between exploration and exploitation, and allocates the simulation budget to solutions wisely, so that solutions with satisfactory performance can be identified using a reasonable amount of computation effort. We demonstrate the performance of the simulation optimization method via a computational study, with problem instances developed based on the operational data of a container port in Shanghai.

The remainder of the paper is organized as follows. Section 2 describes the problem and presents the stochastic optimization model. Section 3 elaborates on our simulation optimization method. Section 4 describes the settings of our computational experiments and reports on the computational results. Section 5 concludes the paper. The mathematical proof of a property, pseudo-code of a subroutine, and descriptions of a benchmark heuristic are presented in the Appendix.

2 Problem Description and Formulation

We consider a set of vessels that need to be served at a container port during a planning horizon. The vessels are classified into deep-sea vessels and feeders. Each deep-sea vessel has a scheduled port arrival time and a target departure time. Deep-sea vessels strictly follow their voyage schedules and arrive at the port on time. If the port fails to complete service at or before the target departure

time of a deep-sea vessel, then departure delay is incurred, resulting in a tardiness penalty cost. Each feeder has an initially scheduled port arrival time. However, in order to mitigate congestion, the port operator may need to control the arrivals of the feeders by altering their arrival plans. The deviation between the assigned port arrival time and the initially scheduled port arrival time of each feeder is referred to as the schedule displacement of the feeder. Schedule displacement of a feeder indicates the amount of adjustments that need to be made on the voyage schedule of the feeder, and it incurs a penalty cost. Furthermore, large schedule displacements are undesirable for feeder operators, and a pre-specified upper limit is thus imposed on the schedule displacement of each feeder. Consequently, each feeder has a time window during which it is allowed to arrive at the port.

Following the real-life practice, berth segments are allocated to deep-sea vessels and feeders in a two-stage fashion, which comprises a planning stage and an execution stage. At the planning stage, the service times of deep-sea vessels are known to the port operator. However, accurate throughput information of feeders is unavailable. Only the probability distributions of the feeder service times, derived from historical data, are available. At this stage, berth segments are allocated to deep-sea vessels based on deterministic port arrival times and service times, while each feeder is only assigned a port arrival time. Once a berth segment has been allocated to a deep-sea vessel, the berth segment will be reserved for the deep-sea vessel. In other words, at the execution stage, the berth segment will not be assigned to any feeder during the deep-sea vessel's reserved time interval.

At the execution stage, each deep-sea vessel is served by the berth segments during the time interval reserved for it. Feeders arrive at the port according to the assigned arrival times. Once a feeder arrives at the port, its throughput information becomes available, and its service time can be accurately estimated. Berth segments are dynamically allocated to the arrived feeders according to a certain service rule. In our model, we consider a first-come first-served rule, which is being used at the Port of Shanghai, for allocating berth segments to feeders. Specifically, when a feeder i arrives at the port, it will join the waiting line (ties broken randomly if multiple feeders arrive at the same time). Suppose feeder i has a service time τ_i and is the first feeder in the waiting line. Then, feeder i will start being served, say at time t, if a berth segment becomes available and is not reserved for any deep-sea vessel throughout the time interval $[t, t + \tau_i]$ (ties broken randomly if multiple berth segments are available at the same time). If such berth segment is not available when feeder i arrives, then feeder i will need to wait in the queue.

Note that since berth segments are reserved for deep-sea vessels, the departure time of each deep-sea vessel is deterministic and is determined solely at the planning stage. Because berth segments are allocated to feeders at the execution stage based on the berth plans of deep-sea vessels, it is important that at the planning stage, berth plans of deep-sea vessels and arrival schedules of feeders are made by taking into account both the uncertainties of feeder service times and the operational service rules of feeders so that the queue length of feeders is well controlled. Note also that although our model considers a first-come first-served rule, our solution method can be easily extended to handle the situation where feeders are served according to other service rules; see Section 5.

For the purpose of modeling, we discretize the planning horizon and assume that all time-related parameters are integer-valued. For simplicity, we assume that the port is empty at the beginning of the planning horizon, and the planning horizon is set to be long enough so that all the vessels considered can finish service within the planning horizon (see Section 5 for a discussion of how to handle the situation when these two assumptions are violated). We assume that all feeders have the same length σ, and that the lengths of deep-sea vessels are multiples of σ. This assumption is justified by the fact that the lengths of deep-sea vessels are much larger than the lengths of feeders, and the lengths of feeders are within a relatively small range compared to the lengths of deep-sea vessels. We assume that the safety clearance between two adjacent vessels along the quay is included in the vessel lengths. We discretize the quay into a set of berth segments, with each berth segment having a length equal to the length of a feeder. As a result, each feeder occupies exactly one berth segment while each deep-sea vessel occupies multiple berth segments when being served. We also assume that there are no space restrictions; that is, each berth segment is compatible with all vessels, regardless of the vessels' draft and width. In Section 5, we discuss how our solution method can be modified to handle problems with non-identical feeder lengths and problems with space restrictions on berth assignment.

In practice, containers may need to be transshipped between feeders and deep-sea vessels (see, e.g., Jin et al. 2015 and Iris et al. 2018). However, transshipment operations usually cannot be conducted within a short period of time. For example, if containers are to be transshipped from a feeder to a deep-sea vessel, then the containers often need to be discharged from the feeder and stored in the yard for several days before the deep-sea vessel arrives, so that the port operator can arrange the loading and discharging plans for the deep-sea vessel with detailed container information. Note that our model considers deep-sea vessels and feeders that need to be served in a short time period (e.g., one to two days), and the solution obtained is for short-term operational

planning. We therefore assume that in our model no transshipment is required between feeders and deep-sea vessels. This assumption helps avoid the necessity of accounting for the inter-relationship between feeders and deep-sea vessels, and thus improves the tractability of our model.

When we determine the berth plan, we impose an upper limit on the expected queue length of feeders, so as to control the traffic in the port and the feeder waiting time (see Jacquillat and Odoni 2015 for an application of such an expected queue length constraint in an airport traffic control model). Note that an alternative way to restrict the queue length of feeders is to setup a chance constraint on the probability of exceeding a queue length limit. A discussion of how our solution method can be applied to problems with a chance constraint is provided in Section 5.

Our problem involves decisions made at the planning stage. They include decisions for assigning each deep-sea vessel a berthing time and a berthing position, and for assigning each feeder a port arrival time. The objective of our problem is to minimize the weighted total departure tardiness of deep-sea vessels and the weighted total schedule displacement of feeders. We denote this problem as **P**. The mathematical formulation of problem **P** is presented as follows:

Input data:

T: Length of the planning horizon.

N_1: Number of deep-sea vessels to be served.

N_2: Number of feeders to be served.

B: Number of berth segments available.

H_i: Service time of deep-sea vessel i.

R_i: Number of berth segments that deep-sea vessel i needs to occupy.

A_i: Port arrival time of deep-sea vessel i.

D_i: Target port departure time of deep-sea vessel i.

S_i: Initially scheduled port arrival time of feeder i.

$[\underline{E}_i, \bar{E}_i]$: Time interval during which feeder i is allowed to arrive at the port.

$\bar{Q}$: Upper limit on the expected queue length of feeders.

C_{1i}: Unit cost of departure tardiness of deep-sea vessel i.

C_{2i}: Unit cost of schedule displacement of feeder i.

The input data also include the probability distribution of the service time of each feeder, and we let G_i denote the mean service time of feeder i.

Decision variables:

x_{ibt}: $= 1$ if deep-sea vessel i is served by berth segments $b, b+1, \ldots, b+R_i-1$ during the time interval $[t, t+H_i]$; 0 otherwise.

y_{it}: $= 1$ if feeder i is assigned an arrival time t; 0 otherwise.

l_{1i}: Departure tardiness of deep-sea vessel i.

l_{2i}: Schedule displacement of feeder i.

Random variables:

$Q_t(\mathbf{x}, \mathbf{y})$: Number of feeders that are waiting for service during the time period $[t, t+1]$ if the feeders are served according to a given service plan $(\mathbf{x}, \mathbf{y})$ and the feeders are allocated to the available berths on a first-come first-served basis at the execution stage, where $\mathbf{x} = (x_{ibt} \mid i = 1, \ldots, N_1;\ b = 1, \ldots, B;\ t = 0, 1, \ldots, T-H_i)$ and $\mathbf{y} = (y_{it} \mid i = 1, \ldots, N_2;\ t = 0, 1, \ldots, T-1)$.

Mathematical programming formulation:

$$\mathbf{P}: \text{ minimize } \sum_{i=1}^{N_1} C_{1i} l_{1i} + \sum_{i=1}^{N_2} C_{2i} l_{2i} \tag{1}$$

$$\text{subject to } \sum_{b=1}^{B-R_i+1} \sum_{t=A_i}^{T-H_i} x_{ibt} = 1 \quad (i = 1, \ldots, N_1) \tag{2}$$

$$\sum_{t=\underline{E}_i}^{\bar{E}_i} y_{it} = 1 \quad (i = 1, \ldots, N_2) \tag{3}$$

$$\sum_{i=1}^{N_1} \sum_{b'=\max\{1, b-R_i+1\}}^{\min\{b, B-R_i+1\}} \sum_{t'=\max\{0, t-H_i+1\}}^{\min\{t, T-H_i\}} x_{ib't'} \le 1 \quad (b = 1, \ldots, B;\ t = 0, 1, \ldots, T-1) \tag{4}$$

$$l_{1i} = \max\left\{0, \sum_{b=1}^{B-R_i+1} \sum_{t=A_i}^{T-H_i} (t + H_i) x_{ibt} - D_i\right\} \quad (i = 1, \ldots, N_1) \tag{5}$$

$$l_{2i} = \left| \sum_{t=\underline{E}_i}^{\bar{E}_i} t y_{it} - S_i \right| \quad (i = 1, \ldots, N_2) \tag{6}$$

$$E[Q_t(\mathbf{x}, \mathbf{y})] \le \bar{Q} \quad (t = 0, 1, \ldots, T-1) \tag{7}$$

$$x_{ibt} \in \{0, 1\} \quad (i = 1, \ldots, N_1;\ b = 1, \ldots, B - R_i + 1;\ t = 0, 1, \ldots, T - H_i) \tag{8}$$

$$y_{it} \in \{0, 1\} \quad (i = 1, \ldots, N_2;\ t = 0, 1, \ldots, T-1) \tag{9}$$

In objective function (1), the unit cost C_{1i} represents the weight of the departure tardiness of deep-sea vessel i, and the unit cost C_{2i} represents the weight of the schedule displacement of feeder i. Thus, objective function (1) minimizes the weighted total tardiness and schedule displacement

of vessels. Constraint (2) ensures that each deep-sea vessel is served during the planning horizon. Constraint (3) ensures that each feeder i arrives at the port during its feasible arrival time interval $[\underline{E}_i, \bar{E}_i]$. Note that if a feeder i cannot arrive earlier than its initially scheduled port arrival time S_i, then we can set $\underline{E}_i$ equal to S_i. Constraint (4) ensures that each berth segment b is occupied by at most one deep-sea vessel during each time period $[t, t+1]$. This is because if $b - R_i + 1 \leq b' \leq b$ and $t - H_i + 1 \leq t' \leq t$, then vessel i will occupy berth segment b during the time period $[t, t+1]$ when $x_{ib't'} = 1$ (i.e., when this vessel is assigned to berth segments $b', b'+1, \ldots, b'+R_i-1$ and time interval $[t', t' + H_i]$). Constraint (5) determines the departure tardiness of each deep-sea vessel. Constraints (6) determines the schedule displacement of each feeder. Constraint (7) imposes an upper limit on the expected queue length of feeders throughout the planning horizon. Constraints (8) and (9) specify the binary requirements of x_{ibt} and y_{it}.

3 Solution Method

Solving problem **P** is highly challenging, because (i) the mathematical formulation contains a large number of constraints and binary decision variables; and (ii) the expected queue length $E[Q_t(\mathbf{x}, \mathbf{y})]$ in constraint (7) is dependent on both the service plan $(\mathbf{x}, \mathbf{y})$ and the service time distribution of the feeders. In fact, when $N_2 = 0$ (i.e., there are no feeders), the problem becomes a berth allocation problem with a minimum weighted total tardiness objective. This special case is a generalization of the single machine scheduling problem with a minimum weighted total tardiness objective, which is known to be strongly NP-hard (Garey and Johnson 1979, p. 237). To tackle this difficult problem **P**, we develop a simulation optimization method that runs in three phases: global phase, local phase, and clean-up phase. In the global phase, we attempt to quickly identify a set of solutions with good estimated performance. Since the purpose of the global phase is to explore the solution space efficiently, a relatively small simulation budget is allocated to the evaluation of visited solutions. In the local phase, we construct a set of solution clusters using the solutions generated in the global phase, and for each cluster we identify the most promising area, in which we use local search to obtain a locally optimal solution. In this phase, we allow the solutions to be evaluated more intensively, as the quality of the solutions obtained in this phase will have a higher impact on the overall performance of the simulation optimization method. The clean-up phase selects the best solution among the solutions generated by the local phase, and the performance of the best solution is evaluated with high precision via simulation. Figure 1 provides a flow chart of our simulation

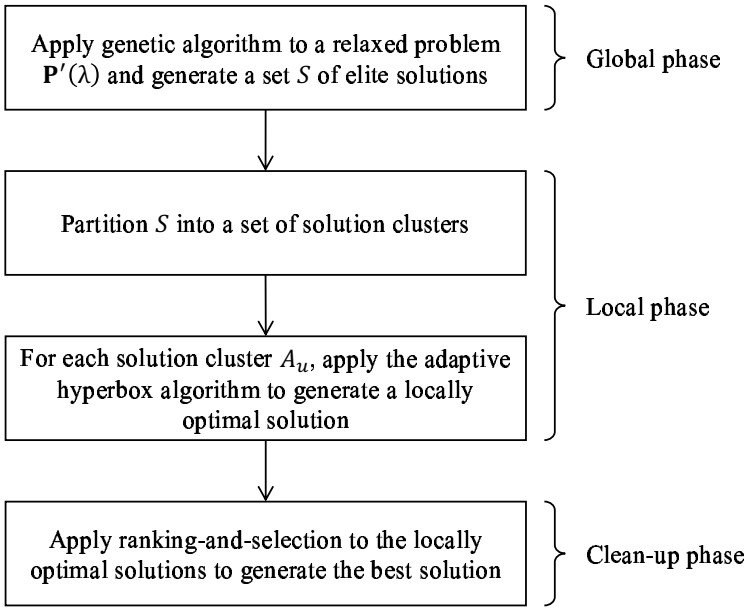


Figure 1: Flow chart of the proposed three-phase simulation optimization method.

optimization method. Our method adapts the method proposed by Xu et al. (2010) for solving discrete simulation optimization problems. Xu et al.'s method samples solutions from a given set of candidate solutions, and thus is typically used for solving problems with relatively small solution spaces and without complicated constraints. We extend Xu et al.'s method for solving problem **P** by incorporating relaxation techniques into the method and introducing new solution sampling strategies in the global and local phases. With these extensions, our method is able to efficiently generate good solutions for practical problems that are usually characterized by large numbers of decision variables.

The global phase, local phase, and clean-up phase of our method are described in Sections 3.1, 3.2, and 3.3, respectively. The simulator that we use to evaluate the performance of a solution is presented in Section 3.4. The parameters of the algorithms used in different phases are presented in Table 1.

3.1 Global Phase

The global phase of our simulation optimization method aims to explore the solution space and quickly identify a set of solutions with relatively good estimated performance. This is done via a genetic algorithm. The advantage of using a genetic algorithm is that the algorithm employs a population-based search which considers many good solutions in parallel. Compared to iterative

Table 1: Parameters used in each phase of the simulation optimization method.

Phase	Parameter	Description
Global phase	s_1	Population size used by the genetic algorithm
	s_2	No. of individual pairs selected for crossover in each iteration
	μ	Mutation rate
	$\bar{\ell}$	Maximum number of iterations
	n_1	No. of simulation replications allocated to each new individual
	n_2	No. of simulation replications allocated to each individual that has been visited before
	λ_0	Initial value of the parameter λ of the relaxed problem $\mathbf{P}'(\lambda)$
	ζ	Step size for updating parameter λ
Local phase	s_3	Maximum number of solutions in each solution cluster
	s_4	Maximum number of solutions evaluated in each iteration
	h	Parameter of the adaptive hyperbox algorithm's stopping condition
	n_3	No. of simulation replications allocated to each visited solution
	$\bar{\lambda}$	Value of the parameter λ used in the local phase
Clean-up phase	δ	Indifference zone
	ϵ	Parameter used to define the confidence level
	n_4	No. of simulation replications allocated to the best solution

search methods that search the solution space by considering only one solution at a time, a genetic algorithm tends to be more robust to stochastic noise (Xu et al. 2010) and is widely used as a component of simulation optimization (Fu el al. 2005; Xu et al. 2015; Amaran et al. 2016).

Our genetic algorithm solves a relaxed version of problem $\mathbf{P}$ iteratively. This relaxed problem, which has a nonnegative parameter λ, is defined as follows:

$$\mathbf{P}'(\lambda):\ \text{minimize}\quad \sum_{i=1}^{N_1} C_{1i} l_{1i} + \sum_{i=1}^{N_2} C_{2i} l_{2i} + \lambda\Delta \tag{10}$$

$$\text{subject to } \Delta = \max\Big\{0,\ \max_{0,1,\ldots,T-1}\big\{E[Q_t(\mathbf{x},\mathbf{y})]\big\} - \bar{Q}\Big\} \tag{11}$$

$$\text{constraints (2)–(6) and (8)–(9)}$$

Variable Δ measures the extent to which constraint (7) is violated, and λ is the unit penalty on the constraint violation. Clearly, problem $\mathbf{P}'(\lambda)$ is always feasible. We have the following property.

Property 1 *If problem* $\mathbf{P}$ *is feasible, then there exists* $\hat{\lambda} > 0$ *such that any optimal solution of problem* $\mathbf{P}'(\hat{\lambda})$ *is also optimal to problem* $\mathbf{P}$.

Proof: See Appendix.

Property 1 implies that problem $\mathbf{P}$ can be solved optimally by solving $\mathbf{P}'(\lambda)$ with a sufficiently large λ, provided that $\mathbf{P}$ is feasible. Instead of solving problem $\mathbf{P}$ directly, our genetic algorithm solves problem $\mathbf{P}'(\lambda)$ iteratively, where the value of λ is updated periodically. Solving $\mathbf{P}'(\lambda)$ for a variety of λ values enables the genetic algorithm to explore a broader set of possible service plans.

When λ is small, the service plans generated tend to have smaller tardiness and displacement costs but be more likely to violate constraint (7). When λ is large, the service plans generated tend to be more likely to satisfy constraint (7) but have larger tardiness and displacement costs. Both types of service plans, however, may possess certain characteristics of a good solution that can pass onto their offspring in the crossover process.

Let Θ denote the finite set of $(\mathbf{x}, \mathbf{y})$ values that satisfy constraints (2)–(4) and (8)–(9). For each $\lambda \geq 0$ and each $(\mathbf{x}, \mathbf{y}) \in \Theta$, define

$$g_\lambda(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{N_1} C_{1i} l_{1i} + \sum_{i=1}^{N_2} C_{2i} l_{2i} + \lambda \Delta,$$

where l_{1i}, l_{2i}, and Δ are defined by equations (5), (6), and (11), respectively. We refer to $g_\lambda(\mathbf{x}, \mathbf{y})$ as the performance of the solution $(\mathbf{x}, \mathbf{y})$ for problem $\mathbf{P}'(\lambda)$. The exact value of $g_\lambda(\mathbf{x}, \mathbf{y})$ cannot be obtained easily because $Q_t(\mathbf{x}, \mathbf{y})$ has no closed form over $(\mathbf{x}, \mathbf{y})$. However, since $Q_t(\mathbf{x}, \mathbf{y})$ can be observed through simulation experiments, we can estimate $g_\lambda(\mathbf{x}, \mathbf{y})$ using the sample mean of $Q_t(\mathbf{x}, \mathbf{y})$ observed in multiple independent simulation replications. We let $\bar{g}_\lambda(\mathbf{x}, \mathbf{y})$ denote the estimated performance of $(\mathbf{x}, \mathbf{y})$ obtained via simulation.

In a genetic algorithm, information of a solution is usually encoded into a string of numbers, called an *individual*. In our implementation, each individual is a string $\mathbf{p} = (p_1, \ldots, p_{N_1+N_2})$, which is a permutation of the $(N_1 + N_2)$-tuple $(1, \ldots, N_1 + N_2)$. For each $i = 1, \ldots, N_1 + N_2$, the number p_i represents deep-sea vessel p_i if $p_i \leq N_1$, and represents feeder $p_i - N_1$ if $p_i > N_1$. Furthermore, vessel p_i has a higher service priority than vessel p_j if $i < j$. An individual $\mathbf{p}$ is translated into a solution $(\mathbf{x}, \mathbf{y}) \in \Theta$ via a decoding scheme. This decoding scheme assigns berth segments and service start times to deep-sea vessels. It also assigns arrival times to feeders by reserving some berth space for each feeder. Let $\eta_{bt} = 1$ if berth segment b has been allocated to a vessel during time period $[t, t+1]$, and $\eta_{bt} = 0$ otherwise. The decoding scheme is presented as follows:

Decoding Scheme:

Step 1: Set $\eta_{bt} \leftarrow 0$ for $b = 1, \ldots, B$ and $t = 0, 1, \ldots, T-1$. Set $i \leftarrow 1$.

Step 2: If $p_i \leq N_1$, then (allocate service capacity to deep-sea vessel):

Select the smallest t among $\{A_i, \ldots, T - H_{p_i}\}$ such that a $b \in \{1, \ldots, B - R_{p_i} + 1\}$ exists with $\sum_{b'=b}^{b+R_{p_i}-1} \sum_{t'=t}^{t+H_{p_i}-1} \eta_{b't'} = 0$. Then, select the smallest b such that $\sum_{b'=b}^{b+R_{p_i}-1} \sum_{t'=t}^{t+H_{p_i}-1} \eta_{b't'} = 0$. Set $x_{p_i bt} \leftarrow 1$. Set $\eta_{b't'} \leftarrow 1$ for $b' = b, b+1, \ldots, b+R_{p_i}-1$ and $t' = t, t+1, \ldots, t+H_{p_i}-1$.

Otherwise (assign arrival time to and reserve service capacity for feeder):

Select the smallest t among $\{\underline{E}_{p_i-N_1}, \ldots, T - G_{p_i-N_1}\}$ such that a $b \in \{1, \ldots, B\}$ exists with $\sum_{t'=t}^{t+G_{p_i-N_1}} \eta_{bt'} = 0$. Then, select the smallest b such that $\sum_{t'=t}^{t+G_{p_i-N_1}} \eta_{bt'} = 0$. Set

$$\bar{t} \leftarrow \begin{cases} t, & \text{if } t \leq \bar{E}_{p_i-N_1}; \\ \bar{E}_{p_i-N_1}, & \text{otherwise.} \end{cases}$$

Set $y_{p_i-N_1,\bar{t}} \leftarrow 1$. Set $\eta_{bt'} \leftarrow 1$ for $t' = t, t+1, \ldots, t+G_{p_i-N_1}-1$.

Step 3: If $i = N_1 + N_2$, then stop. Otherwise, set $i \leftarrow i+1$ and go to Step 2.

This decoding scheme allocates service capacity to vessels according to the string of $N_1 + N_2$ numbers in the individual. Specifically, the decoding scheme considers each $i = 1, \ldots, N_1 + N_2$. If $p_i \leq N_1$, then p_i represents deep-sea vessel p_i. In this case, the decoding scheme allocates a feasible service start time and R_{p_i} consecutive berth segments to the vessel so that the departure tardiness of the vessel is minimized. If $p_i > N_1$, then p_i represents feeder $p_i - N_1$. In this case, the decoding scheme allocates a target service start time t and a target berth segment b to feeder i based on the feeder's mean service time $G_{p_i-N_1}$, so that feeder i can complete service as early as possible when being served at berth segment b starting at time t. A feasible arrival time $\bar{t}$ is then assigned to the feeder, so that the waiting time of the feeder is minimized when the feeder's service starts at time t. Note that the decoding scheme makes use of the mean service times of feeders rather than sampled service times; therefore, all parameters in the decoding scheme are deterministic.

Let Ω denote the set of all individuals. It is not difficult to see that by means of the decoding scheme, each individual $\mathbf{p} \in \Omega$ corresponds to one solution $(\mathbf{x}, \mathbf{y}) \in \Theta$. For each $\mathbf{p} \in \Omega$, let $(\mathbf{x}(\mathbf{p}), \mathbf{y}(\mathbf{p}))$ denote the solution obtained by applying the decoding scheme on individual $\mathbf{p}$. The fitness of each individual $\mathbf{p} \in \Omega$ is defined as $1/g_\lambda(\mathbf{x}(\mathbf{p}), \mathbf{y}(\mathbf{p}))$ and is estimated by $1/\bar{g}_\lambda(\mathbf{x}(\mathbf{p}), \mathbf{y}(\mathbf{p}))$. Note that $\bar{g}_\lambda(\mathbf{x}(\mathbf{p}), \mathbf{y}(\mathbf{p}))$ can be obtained by running simulation experiments on the solution $(\mathbf{x}(\mathbf{p}), \mathbf{y}(\mathbf{p}))$. The genetic algorithm, which aims to identify a subset of individuals that result in solutions of $\mathbf{P}$ with small tardiness and displacement costs of vessels, is described as follows:

Genetic Algorithm:

Step 1 (Initialization): Set $\ell \leftarrow 1$ and $\lambda \leftarrow \lambda_0$. Generate an initial population with s_1 individuals. Each individual $\mathbf{p}$ is generated randomly as follows: For each $i = 1, \ldots, N_1 + N_2$, select an available position from $\mathbf{p}$ with all available positions being selected with equal probability, and insert i into the selected position.

Step 2 (Crossover): Randomly select s_2 pairs of distinct individuals from the current population with equal probability. For each pair of individuals $\mathbf{p} = (p_1, \ldots, p_{N_1+N_2})$ and $\bar{\mathbf{p}} = (\bar{p}_1, \ldots, \bar{p}_{N_1+N_2})$, generate two numbers j and j' randomly from $\{1, \ldots, N_1 + N_2\}$ with equal probability, where $j < j'$. Set

$$\mathbf{q} \leftarrow (\underbrace{0, \ldots, 0}_{j-1 \text{ times}}, p_j, p_{j+1}, \ldots, p_{j'}, \underbrace{0, \ldots, 0}_{N_1+N_2-j' \text{ times}}) \text{ and } \bar{\mathbf{q}} \leftarrow (\underbrace{0, \ldots, 0}_{j-1 \text{ times}}, \bar{p}_j, \bar{p}_{j+1}, \ldots, \bar{p}_{j'}, \underbrace{0, \ldots, 0}_{N_1+N_2-j' \text{ times}}).$$

For $i = 1, \ldots, N_1 + N_2$, if $\bar{p}_i$ is not in $\mathbf{q}$ then replace the first 0 in $\mathbf{q}$ with $\bar{p}_i$, and if p_i is not in $\bar{\mathbf{q}}$ then replace the first 0 in $\bar{\mathbf{q}}$ with p_i. Add $\mathbf{q}$ and $\bar{\mathbf{q}}$ to the population.

Step 3 (Mutation): For each individual $\mathbf{p} = (p_1, \ldots, p_{N_1+N_2})$ in the current population, generate a real number $\hat{\mu}$ from a uniform distribution on $[0, 1]$. If $\hat{\mu} \leq \mu$, then randomly select two distinct numbers j and j' from $\{1, \ldots, N_1 + N_2\}$ with equal probability. Swap the positions of p_j and $p_{j'}$ in $\mathbf{p}$.

Step 4 (Evaluation): For each individual $\mathbf{p}$ in the current population, if the individual has been evaluated previously, then perform n_2 additional simulation replications on solution $(\mathbf{x}(\mathbf{p}), \mathbf{y}(\mathbf{p}))$; otherwise perform n_1 simulation replications on solution $(\mathbf{x}(\mathbf{p}), \mathbf{y}(\mathbf{p}))$. Set the fitness of individual $\mathbf{p}$ to be the cumulative sample mean of the solution performance obtained in all replications.

Step 5 (Selection): Sort the individuals of the current population in descending order of fitness. Keep the first s_1 individuals, and discard the other individuals.

Step 6 (Update λ): If $\ell = \bar{\ell}$, then stop. Otherwise, set $\ell \leftarrow \ell + 1$. Consider the first individual $\mathbf{p}$ in the sorted individual list. If the estimated value of $\max_{0,1,\ldots,T-1} \{E[Q_t(\mathbf{x}(\mathbf{p}), \mathbf{y}(\mathbf{p}))]\}$ is greater than $\bar{Q}$, then set $\lambda \leftarrow (1 + \zeta)\lambda$. Go to Step 2.

This genetic algorithm starts with a small λ value. When λ is relatively small, the genetic algorithm focuses on searching for solutions that have small vessel tardiness and displacement costs. If the best individual found so far corresponds to a solution of problem $\mathbf{P}'(\lambda)$ that violates constraint (7), then the value of λ will be increased in the next iteration so that heavier emphasis will be given to the satisfaction of constraint (7). This process is executed repeatedly until the number of iterations reaches the upper limit $\bar{\ell}$.

3.2 Local Phase

When the global phase terminates, we obtain a list of $(\mathbf{x}, \mathbf{y})$ values. These $(\mathbf{x}, \mathbf{y})$ values will be used to initialize the local phase. Unlike the global phase, which diversifies the population of individuals

by solving the relaxed problem $\mathbf{P}'(\lambda)$ with various λ values, the local phase aims to intensify the search for near-optimal solutions of problem $\mathbf{P}'(\bar{\lambda})$, where $\bar{\lambda}$ is a large input parameter. Note that if $\bar{\lambda}$ is sufficiently large, then an optimal solution of problem $\mathbf{P}'(\bar{\lambda})$ is also optimal to problem $\mathbf{P}$ (see Property 1).

Let $\mathcal{S}$ denote the list of $(\mathbf{x}, \mathbf{y})$ values passed onto the local phase, sorted in ascending order of their estimated performance $\bar{g}_{\bar{\lambda}}(\mathbf{x}, \mathbf{y})$. Let $(\mathbf{x}_j, \mathbf{y}_j)$ denote the jth element in the sorted list $\mathcal{S}$. For each $(\mathbf{x}, \mathbf{y}) \in \Theta$, define a vector of integer values $(\mathbf{e}_1(\mathbf{x}); \mathbf{e}_2(\mathbf{x}); \mathbf{e}_3(\mathbf{y}))$, in which $\mathbf{e}_1(\mathbf{x}) = (e_{11}(\mathbf{x}), \ldots, e_{1N_1}(\mathbf{x}))$, $\mathbf{e}_2(\mathbf{x}) = (e_{21}(\mathbf{x}), \ldots, e_{2N_1}(\mathbf{x}))$, and $\mathbf{e}_3(\mathbf{y}) = (e_{31}(\mathbf{y}), \ldots, e_{3N_2}(\mathbf{y}))$, where

$$e_{1i}(\mathbf{x}) = \sum_{b=1}^{B-R_i+1} \sum_{t=0}^{T-H_i} b x_{ibt} \quad \text{and} \quad e_{2i}(\mathbf{x}) = \sum_{b=1}^{B-R_i+1} \sum_{t=0}^{T-H_i} t x_{ibt}$$

for $i = 1, \ldots, N_1$, and

$$e_{3i}(\mathbf{y}) = \sum_{t=\underline{E}_i}^{\bar{E}_i} t y_{it}$$

for $i = 1, \ldots, N_2$. Note that $e_{1i}(\mathbf{x})$ is the first berth segment allocated to deep-sea vessel i, $e_{2i}(\mathbf{x})$ is the service start time of deep-sea vessel i, and $e_{3i}(\mathbf{y})$ is the arrival time of feeder i.

We define the *distance* between any two solutions $(\mathbf{x}_j, \mathbf{y}_j), (\mathbf{x}_{j'}, \mathbf{y}_{j'}) \in \Theta$ as

$$d_{jj'} = \sqrt{\sum_{i=1}^{N_1} \left[\frac{e_{1i}(\mathbf{x}_j) - e_{1i}(\mathbf{x}_{j'})}{B}\right]^2 + \sum_{i=1}^{N_1} \left[\frac{e_{2i}(\mathbf{x}_j) - e_{2i}(\mathbf{x}_{j'})}{T}\right]^2 + \sum_{i=1}^{N_2} \left[\frac{e_{3i}(\mathbf{y}_j) - e_{3i}(\mathbf{y}_{j'})}{T}\right]^2},$$

which is the Euclidean distance between the normalized vectors $\left(\frac{\mathbf{e}_1(\mathbf{x}_j)}{B}; \frac{\mathbf{e}_2(\mathbf{x}_j)}{T}; \frac{\mathbf{e}_3(\mathbf{y}_j)}{T}\right)$ and $\left(\frac{\mathbf{e}_1(\mathbf{x}_{j'})}{B}; \frac{\mathbf{e}_2(\mathbf{x}_{j'})}{T}; \frac{\mathbf{e}_3(\mathbf{y}_{j'})}{T}\right)$. The local phase is initialized with a set of solution clusters obtained by partitioning solution list $\mathcal{S}$. The solution clusters are generated in such a way that the number of solutions in each cluster is no greater than s_3, and that for each cluster, the distance between the best solution in the cluster and any other solution in the same cluster is relatively small. This solution clustering method is similar to the one used by Xu et al. (2010) in their Industrial Strength COMPASS which has been shown to have competitive performance. The procedure for generating solution clusters is given as follows, in which $\mathcal{A}_u$ denotes the uth cluster generated:

Cluster Generation Procedure:

Step 1: Set $u \leftarrow 1$.

Step 2: Set $j \leftarrow \min\{l \mid (\mathbf{x}_l, \mathbf{y}_l) \in \mathcal{S}\}$. Create a new cluster $\mathcal{A}_u \leftarrow \{(\mathbf{x}_j, \mathbf{y}_j)\}$. Set $\mathcal{S} \leftarrow \mathcal{S} \backslash \{(\mathbf{x}_j, \mathbf{y}_j)\}$.

Step 3: If $|\mathcal{A}_u| < s_3$ and $\mathcal{S} \neq \emptyset$, then let k be the element of $\mathcal{S}$ that has the smallest d_{jk} value (ties broken arbitrarily), set $\mathcal{A}_u \leftarrow \mathcal{A}_u \cup \{(\mathbf{x}_k, \mathbf{y}_k)\}$, set $\mathcal{S} \leftarrow \mathcal{S} \setminus \{(\mathbf{x}_k, \mathbf{y}_k)\}$, and repeat Step 3.

Step 4: If $\mathcal{S} = \emptyset$, then stop. Otherwise, set $u \leftarrow u + 1$ and go to Step 2.

For each solution cluster $\mathcal{A}_u$ generated by this procedure, we apply local search to identify a locally optimal solution. We use the adaptive hyperbox algorithm developed by Xu et al. (2013) as the local search method. The adaptive hyperbox algorithm is a locally convergent random search algorithm that has a special neighborhood structure called the *most promising area*, which was initially proposed by Hong and Nelson (2006). The algorithm randomly samples solutions from the most promising area and then updates the most promising area using the visited solutions. This process is executed repeatedly until the solutions converge or some predetermined stopping conditions are satisfied.

Consider any solution cluster $\mathcal{A}_u$. Let $(\mathbf{x}^*, \mathbf{y}^*)$ be the best sampled solution in $\mathcal{A}_u$; that is, $(\mathbf{x}^*, \mathbf{y}^*) = \arg\min_{(\mathbf{x},\mathbf{y})\in\mathcal{A}_u}\{\bar{g}_{\bar{\lambda}}(\mathbf{x}, \mathbf{y})\}$. For notational convenience, we denote

$$(\mathbf{e}_1; \mathbf{e}_2; \mathbf{e}_3) = ((e_{11}, \ldots, e_{1N_1}); (e_{21}, \ldots, e_{2N_1}); (e_{31}, \ldots, e_{3N_2})) = (\mathbf{e}_1(\mathbf{x}); \mathbf{e}_2(\mathbf{x}); \mathbf{e}_3(\mathbf{y}))$$

and

$$(\mathbf{e}_1^*; \mathbf{e}_2^*; \mathbf{e}_3^*) = ((e_{11}^*, \ldots, e_{1N_1}^*); (e_{21}^*, \ldots, e_{2N_1}^*); (e_{31}^*, \ldots, e_{3N_2}^*)) = (\mathbf{e}_1(\mathbf{x}^*); \mathbf{e}_2(\mathbf{x}^*); \mathbf{e}_3(\mathbf{y}^*)).$$

For any set $\mathcal{V}$ of integer vectors $(\mathbf{e}_1; \mathbf{e}_2; \mathbf{e}_3)$, let

$$L_{1i} = \begin{cases} \max_{(\mathbf{e}_1;\mathbf{e}_2;\mathbf{e}_3)\in\mathcal{V}}\{e_{1i} \mid e_{1i} < e_{1i}^*\}, & \text{if exists;} \\ 1, & \text{otherwise;} \end{cases} \tag{12}$$

$$U_{1i} = \begin{cases} \min_{(\mathbf{e}_1;\mathbf{e}_2;\mathbf{e}_3)\in\mathcal{V}}\{e_{1i} \mid e_{1i} > e_{1i}^*\}, & \text{if exists;} \\ B - R_i + 1, & \text{otherwise;} \end{cases} \tag{13}$$

$$L_{2i} = \begin{cases} \max_{(\mathbf{e}_1;\mathbf{e}_2;\mathbf{e}_3)\in\mathcal{V}}\{e_{2i} \mid e_{2i} < e_{2i}^*\}, & \text{if exists;} \\ A_i, & \text{otherwise;} \end{cases} \tag{14}$$

and

$$U_{2i} = \begin{cases} \min_{(\mathbf{e}_1;\mathbf{e}_2;\mathbf{e}_3)\in\mathcal{V}}\{e_{2i} \mid e_{2i} > e_{2i}^*\}, & \text{if exists;} \\ T - H_i + 1, & \text{otherwise;} \end{cases} \tag{15}$$

for $i = 1, \ldots, N_1$. For any $\mathcal{V}$, let

$$L_{3i} = \begin{cases} \max_{(\mathbf{e}_1;\mathbf{e}_2;\mathbf{e}_3)\in\mathcal{V}}\{e_{3i} \mid e_{3i} < e_{3i}^*\}, & \text{if exists;} \\ \underline{E}_i, & \text{otherwise;} \end{cases} \tag{16}$$

and

$$U_{3i} = \begin{cases} \min_{(\mathbf{e}_1;\mathbf{e}_2;\mathbf{e}_3)\in\mathcal{V}}\{e_{3i} \mid e_{3i} > e^*_{3i}\}, & \text{if exists;} \\ \bar{E}_i, & \text{otherwise;} \end{cases} \tag{17}$$

for $i = 1, \ldots, N_2$. The quantity L_{1i} is the e_{1i} value, among the e_{1i} values of those elements of $\mathcal{V}$, that is the closest to e^*_{1i} but is smaller than e^*_{1i} (if such an e_{1i} value exists). Similarly, U_{1i} is the e_{1i} value, among the e_{1i} values of those elements of $\mathcal{V}$, that is the closest to e^*_{1i} but is larger than e^*_{1i} (if such an e_{1i} value exists). The quantities L_{2i}, U_{2i}, L_{3i}, and U_{3i} can be interpreted in a similar fashion. The hyperbox containing $(\mathbf{e}^*_1; \mathbf{e}^*_2; \mathbf{e}^*_3)$ is

$$\mathcal{H} = \big\{(\mathbf{e}_1; \mathbf{e}_2; \mathbf{e}_3) \,\big|\, L_{1i} \le e_{1i} \le U_{1i}; L_{2i} \le e_{2i} \le U_{2i}; L_{3j} \le e_{3j} \le U_{3j}; 1 \le i \le N_1; 1 \le j \le N_2\big\}. \tag{18}$$

In other words, $\mathcal{H}$ is a $(2N_1 + N_2)$-dimensional hyperrectangle. The two boundaries of the first N_1 dimensions are L_{1i} and U_{1i} $(i = 1, \ldots, N_1)$; the two boundaries of the next N_1 dimensions are L_{2i} and U_{2i} $(i = 1, \ldots, N_1)$; and the two boundaries of the other N_2 dimensions are L_{3j} and U_{3j} $(j = 1, \ldots, N_2)$. Hyperbox $\mathcal{H}$ contains vectors $(\mathbf{e}_1; \mathbf{e}_2; \mathbf{e}_3)$, which represent the solutions by the deep-sea vessels' first berth segments, the deep-sea vessels' service start times, and the feeders' arrival times.

Let $\mathbb{Z}^{2N_1+N_2}$ denote the $(2N_1+N_2)$-dimensional integer space. The set $\mathcal{H} \cap \mathbb{Z}^{2N_1+N_2}$ is the most promising area in which the adaptive hyperbox algorithm focuses on sampling solutions. Note that in a straightforward implementation of the adaptive hyperbox algorithm, the hyperbox is defined over the binary vectors $(\mathbf{x}, \mathbf{y})$ instead of the integer vectors $(\mathbf{e}_1; \mathbf{e}_2; \mathbf{e}_3)$. Our implementation of the adaptive hyperbox algorithm using the integer vectors $(\mathbf{e}_1; \mathbf{e}_2; \mathbf{e}_3)$ is advantageous over the straightforward implementation in the following aspects: (i) the dimension of the hyperbox is reduced from $N_1BT + N_2T$ to $2N_1 + N_2$, leading to a much smaller space in which the adaptive hyperbox algorithm samples solutions; and (ii) the difference between the values of two solutions that are close to each other is relatively small when the integer vectors are used, leading to low variability of solution values and smoother convergence of the adaptive hyperbox algorithm.

The adaptive hyperbox algorithm iteratively updates $\mathcal{V}$ and $\mathcal{H}$, and selects integer vectors randomly from $\mathcal{H} \cap \mathbb{Z}^{2N_1+N_2}$ in each iteration. Initially, we set $\mathcal{V}$ to

$$\mathcal{V}_0 = \big\{(\mathbf{e}_1; \mathbf{e}_2; \mathbf{e}_3) \;\big|\; (\mathbf{x}, \mathbf{y}) \in \mathcal{A}_u\big\}.$$

Given any integer vector $(\mathbf{e}_1; \mathbf{e}_2; \mathbf{e}_3) \in \mathcal{H} \cap \mathbb{Z}^{2N_1+N_2}$, the corresponding solution $(\mathbf{x}, \mathbf{y})$ can be obtained as follows: For each $i = 1, \ldots, N_1$, each $b = 1, \ldots, B$, and each $t = 0, 1, \ldots, T-1$, let

$x_{ibt} = 1$ if $b = e_{1i}$ and $t = e_{2i}$, and let $x_{ibt} = 0$ otherwise. For each $i = 1, \ldots, N_2$ and each $t = 0, 1, \ldots, T$, let $y_{it} = 1$ if $t = e_{3i}$, and let $y_{it} = 0$ otherwise. Note that the solution $(\mathbf{x}, \mathbf{y})$ generated from $(\mathbf{e}_1; \mathbf{e}_2; \mathbf{e}_3)$ may violate constraint (4), and a solution refinement subroutine (to be described later) is needed to obtain a feasible solution.

Our implementation of the adaptive hyperbox algorithm is described below:

Adaptive Hyperbox Algorithm:

Step 1: Determine $(\mathbf{x}^*, \mathbf{y}^*)$, $(\mathbf{e}_1^*; \mathbf{e}_2^*; \mathbf{e}_3^*)$, and $\mathcal{V}_0$. Set $\mathcal{V} \leftarrow \mathcal{V}_0$.

Step 2: Determine $\mathcal{H}$ using equations (12)–(18). Randomly generate s_4 integer vectors from $\mathcal{H} \cap \mathbb{Z}^{2N_1+N_2}$. Each integer vector $(\mathbf{e}_1; \mathbf{e}_2; \mathbf{e}_3)$ is generated as follows: For each $i = 1, \ldots, N_1$, generate e_{1i} and e_{2i} uniformly from $\{L_{1i}, L_{1i}+1, \ldots, U_{1i}\}$ and $\{L_{2i}, L_{2i}+1, \ldots, U_{2i}\}$, respectively. For each $i = 1, \ldots, N_2$, generate e_{3i} uniformly from $\{L_{3i}, L_{3i}+1, \ldots, U_{3i}\}$. Remove duplicates from the generated integer vectors. Let $\mathcal{T}$ be the set of the remaining integer vectors. Set $\mathcal{V} \leftarrow \mathcal{V} \cup \mathcal{T}$.

Step 3: For each $(\mathbf{e}_1; \mathbf{e}_2; \mathbf{e}_3) \in \mathcal{V}$, obtain the corresponding solution $(\mathbf{x}, \mathbf{y})$. If $(\mathbf{x}, \mathbf{y})$ satisfies constraint (4), then set $\mathcal{A}_u \leftarrow \mathcal{A}_u \cup \{(\mathbf{x}, \mathbf{y})\}$. Perform n_3 simulation replications on solution $(\mathbf{x}, \mathbf{y})$, and set $\bar{g}_{\bar{\lambda}}(\mathbf{x}, \mathbf{y})$ to be the cumulative sample mean of $g_{\bar{\lambda}}(\mathbf{x}, \mathbf{y})$ among all simulation replications (including the replications performed in previous iterations).

Step 4: Determine $(\mathbf{x}^*, \mathbf{y}^*) = \arg\min_{(\mathbf{x},\mathbf{y}) \in \mathcal{A}_u} \{\bar{g}_{\bar{\lambda}}(\mathbf{x}, \mathbf{y})\}$. Determine $(\mathbf{e}_1^*; \mathbf{e}_2^*; \mathbf{e}_3^*)$. If the stopping condition is satisfied, then stop; otherwise, go to Step 2.

Xu et al. (2013) have shown that when solutions are uniformly and independently sampled from the most promising area in each iteration, the adaptive hyperbox algorithm is locally convergent; that is, the sequence of solutions generated by the algorithm converges with probability 1 to a locally optimal solution that is not worse than any of its neighbor solutions in the most promising area. In Step 2 of our implementation, the solutions are generated from the integer vectors that are sampled uniformly and independently from $\mathcal{H} \cap \mathbb{Z}^{2N_1+N_2}$. Hence, our implementation of the adaptive hyperbox algorithm is also locally convergent (i.e., the sequence of solutions generated converges with probability 1 to a locally optimal solution of problem $\mathbf{P}'(\bar{\lambda})$) as the number of iterations approaches infinity. However, in practice, stopping conditions are needed to terminate the algorithm at some point. In our implementation, the algorithm is terminated when the current best solution $(\mathbf{x}^*, \mathbf{y}^*)$ is unchanged for h consecutive iterations. The solution $(\mathbf{x}^*, \mathbf{y}^*)$ is then treated as the optimal solution for cluster $\mathcal{A}_u$.

We observe from preliminary computational results that Step 3 of the adaptive hyperbox algorithm frequently generates solutions that violate constraint (4). To improve the quality of the solutions generated in Step 3, we develop a solution refinement subroutine. When a solution generated in Step 3 violates constraint (4), there are multiple deep-sea vessels occupying a berth segment simultaneously. In such a situation, we select one of these deep-sea vessels and reassign a service start time to the selected vessel, so that no conflicts will be incurred between the selected vessel and any other vessels. This process is executed repeatedly until all conflicts are resolved. Details of the solution refinement subroutine are provided in the Appendix.

We execute the adaptive hyperbox algorithm with the solution refinement subroutine for each of the solution clusters $\mathcal{A}_u$ generated by the cluster generation procedure. Therefore, at the end of the local phase, we obtain m locally optimal solutions, where m is the number of solution clusters generated by the cluster generation procedure. We then execute a clean-up phase to determine the best solution among the m locally optimal solutions.

3.3 Clean-up Phase

At the beginning of the clean-up phase, there are m candidate solutions. If $m = 1$, then we determine accurately whether the solution satisfies constraint (7) (and is therefore feasible to problem **P**) with more simulation runs. Otherwise, we need to determine accurately whether the m solutions are feasible to problem **P** and choose the best solution among the feasible solutions. This requires intensive evaluation of the expected queue lengths of the feeders in the solutions.

When the number of candidate solutions is large, evaluating each solution with high precision would require an unaffordable amount of simulation budget. One commonly used method for determining the best solution in a clean-up phase is the ranking-and-selection method proposed by Boesel et al. (2003) which aims to minimize the number of simulation replications while ensuring that the selected solution is better than the other alternatives by at least δ with probability at least $1 - \epsilon$, where δ is called the indifference zone and $1 - \epsilon$ is the desired confidence level. Given the indifference zone, the confidence level, the number of simulation replications previously performed on each solution, and the variation of solution performance obtained in previous simulation replications, the ranking-and-selection method determines the number of additional replications that need to be allocated to each solution, and chooses the solution that has the best cumulative average performance after all additional replications are performed. In our implementation of the clean-up phase, if $m > 1$, then we select the best solution among the m candidate solutions

using the ranking-and-selection method. When applying the ranking-and-selection method, the performance of each candidate solution $(\mathbf{x}, \mathbf{y})$ is measured by $g_{\bar{\lambda}}(\mathbf{x}, \mathbf{y})$, which is estimated using $\bar{g}_{\bar{\lambda}}(\mathbf{x}, \mathbf{y})$. When there is only one solution left, we evaluate the solution by running n_4 additional simulation replications, where n_4 is a large input parameter, so that the solution is evaluated with high precision. Let $\hat{Q}_{\max}(\mathbf{x}^*, \mathbf{y}^*)$ be the sample mean of $\max_{t=0,1,\ldots,T-1}\{Q_t(\mathbf{x}^*, \mathbf{y}^*)\}$ generated for the best solution $(\mathbf{x}^*, \mathbf{y}^*)$ using n_4 simulation replications. The best solution $(\mathbf{x}^*, \mathbf{y}^*)$ is deemed as feasible to problem $\mathbf{P}$ if $\hat{Q}_{\max}(\mathbf{x}^*, \mathbf{y}^*) \leq \bar{Q}$, in which case the solution value obtained for problem $\mathbf{P}$ is $g_0(\mathbf{x}^*, \mathbf{y}^*)$, and is deemed as infeasible to problem $\mathbf{P}$ otherwise.

3.4 The Simulator

For any solution $(\mathbf{x}, \mathbf{y}) \in \Theta$, we may use discrete-event simulation to determine the objective function value of $(\mathbf{x}, \mathbf{y})$ and to determine whether $(\mathbf{x}, \mathbf{y})$ satisfies constraint (7). In our discrete-event simulation, the events are feeder-arrival and vessel-departure. The feeder-arrival event is triggered whenever a feeder arrives at the port, and the vessel-departure event is triggered whenever a deep-sea vessel or feeder finishes service at the port. Note that the arrival times of feeders are given by the values of the y_{it} variables, and the time intervals during which each berth segment is reserved for deep-sea vessels can be derived using the values of the x_{ibt} variables. When a feeder i arrives, say at time t, the feeder joins the queue, and the service time of the feeder τ_i is observed (by sampling). If the feeder is in the first place of the queue and there exist some berth segments that are not reserved for any deep-sea vessels during the time interval $[t, t + \tau_i]$, then the feeder will be served immediately; otherwise, the feeder will wait in the queue. When vessel-departure occurs, some berth segments are released. If there exist some feeders in the queue at the moment, then the feeders will be assigned to the released berth segments according to the first-come first-served discipline described in Section 2. The discrete-event process is terminated when all the vessels are served and the port becomes empty. Each run of the discrete-event simulation results in an observation of $Q_t(\mathbf{x}, \mathbf{y})$ for $t = 0, 1, \ldots, T-1$. The values of $E[Q_t(\mathbf{x}, \mathbf{y})]$ and $g_\lambda(\mathbf{x}, \mathbf{y})$ can be estimated by running the simulation multiple times and taking the sample mean of multiple runs.

4 Computational Experiments

The computational experiments are designed for the following purposes. First, we would like to evaluate the computational performance of the simulation optimization method for solving problem instances of realistic sizes. For this purpose, we generate problem instances based on the operational

data of the Yangshan Deep-water Port in Shanghai, and compare the computational performance of our simulation optimization method with the performances of some benchmark methods. We also evaluate the effectiveness of the solution refinement subroutine in the local phase of our method. Since users of our simulation optimization method may want to set different queue length limits for feeders under different situations (e.g., different weather conditions), our second goal is to examine how the computational results are affected as the value of $\bar{Q}$ varies. Our third goal is to evaluate the benefits of controlling the queue length of feeders when making berth plans. For this purpose, we analyze the solutions obtained by the current practice of the port operators that ignore the queue length restriction, and compare the solutions with those obtained by our simulation optimization method.

All algorithms were implemented in C#.Net and run on a computer with a 64-bit Intel i7-6700 3.40GHz CPU and 32GB of RAM.

4.1 Problem Instances and Algorithm Parameters

We generate test instances based on the operational data of the Yangshan Deep-water Port in Shanghai. Table 2 presents the statistics of the operational data of year 2016 at the port. For each month of the year, the following statistics are presented:

- Vess_Num: Number of deep-sea vessels served.
- Feed_Num: Number of feeders served.
- Vess_Len_Min, Vess_Len_Max, Vess_Len_Avg: Minimum, maximum, and average lengths (in meters) of deep-sea vessels.
- Feed_Len_Min, Feed_Len_Max, Feed_Len_Avg: Minimum, maximum, and average lengths (in meters) of feeders.
- Vess_Time_Min, Vess_Time_Max, Vess_Time_Avg: Minimum, maximum, and average service times (in hours) of deep-sea vessels.
- Feed_Time_Min, Feed_Time_Max, Feed_Time_Avg: Minimum, maximum, and average service times (in hours) of feeders.

It can be observed from Table 2 that the number of feeders served in the port is significantly larger than the number of deep-sea vessels. The average numbers of deep-sea vessels and feeders served monthly are 377.3 and 1146.9, respectively, indicating that the daily average numbers of deep-sea vessels and feeders served are 12.6 and 38.2, respectively. In each of our test instances, the number of deep-sea vessels and the number of feeders are set to $N_1 = 25$ and $N_2 = 80$, respectively,

Table 2: Monthly statistics of the operational data for 2016 at the Yangshan Deep-water Port*

	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec	Average
Vess_Num	381	356	403	382	376	396	411	398	360	363	344	357	377.3
Feed_Num	1158	1061	1058	1122	1241	1162	1211	1250	1161	1104	1104	1131	1146.9
Vess_Len_Min	224.0	228.0	228.0	228.0	228.0	228.0	209.0	228.0	228.0	222.0	222.0	207.0	223.3
Vess_Len_Max	400.0	400.0	400.0	400.0	400.0	400.0	400.0	400.0	400.0	400.0	400.0	400.0	400.0
Vess_Len_Avg	308.1	307.4	312.5	312.8	312.8	316.6	314.5	319.6	318.3	321.0	321.2	320.9	315.5
Feed_Len_Min	75.0	75.0	75.0	75.0	75.0	75.0	75.0	75.0	75.0	75.0	75.0	75.0	75.0
Feed_Len_Max	140.0	140.0	157.0	140.0	140.0	140.0	140.0	140.0	140.0	140.0	140.0	140.0	141.4
Feed_Len_Avg	96.0	96.1	97.3	95.4	95.2	96.6	96.2	96.1	95.1	95.7	96.1	97.0	96.1
Vess_Time_Min	6.2	5.3	5.2	5.5	5.6	6.0	5.5	6.0	5.1	5.2	5.5	5.6	5.6
Vess_Time_Max	37.5	35.8	35.5	33.4	35.7	30.1	28.3	35.7	32.0	35.5	35.5	32.7	34.0
Vess_Time_Avg	14.8	13.2	14.8	14.1	14.3	13.3	14.7	14.9	15.3	16.8	16.8	16.9	15.0
Feed_Time_Min	0.4	0.4	0.7	0.6	0.2	0.7	0.2	0.3	0.3	0.5	0.2	0.5	0.4
Feed_Time_Max	10.0	8.2	9.7	8.1	9.7	9.2	9.1	8.6	9.0	9.1	9.7	8.5	9.1
Feed_Time_Avg	2.8	3.4	4.2	3.2	3.8	4.0	3.2	3.5	4.2	4.5	3.8	4.0	3.7

*These statistics are obtained by ignoring data entries that are either improperly recorded or recorded under abnormal situations (e.g., long vessel service times caused by equipment breakdowns).

which reflect the number of service requests over two days. The reason for using such a parameter setting is that accurate throughput information of deep-sea vessels usually becomes available two days before the vessels arrive at the port. The length of each time period of the planning horizon is set to 1 hour. The length of the planning horizon is set to $T = 72$ (i.e., 3 days), which is long enough to ensure service completion of all vessels.

Since the port has a quaywall of 5600 meters, we set the number of berth segments to $B = 40$, so that each berth segment is 140 meters long, which can accommodate most feeders. The length of each deep-sea vessel ranges between 207 meters and 400 meters. Since the lengths of deep-sea vessels are modeled as multiples of the length of each feeders, we generate the length of each deep-sea vessel i by $R_i = \lceil \frac{\alpha_i}{140} \rceil$, where α_i is a real number generated uniformly from $[207, 400]$.

The mean service time of deep-sea vessels over the year is 15.0 hours, and the variance of deep-sea vessel service times is 40.8. Hence, we generate the service time H_i of each deep-sea vessel i using a normal distribution with mean 15.0 and variance 40.8. When generating the value of H_i, we round the sampled value up to the nearest positive integer. In order to evaluate the impact of the feeder service time variance on the solution of problem **P**, we generate test instances with three different probability distributions of the feeder service times, where the three probability distributions differ from each other only in the variance. The mean service time of feeders over the year is 3.7 hours, and the variance of the feeder service times is 5.2. Hence, our first probability distribution is a normal distribution with mean 3.7 and variance 2.6, our second probability distribution is a normal distribution with mean 3.7 and variance 5.2, and our third probability distribution is a normal distribution with mean 3.7 and variance 7.8. When sampling feeder service times, we truncate the

Table 3: Parameter values used in the computational study.

Low service time variance			Medium service time variance			High service time variance		
Problem set	$\bar{Q}$	Service time variance	Problem set	$\bar{Q}$	Service time variance	Problem set	$\bar{Q}$	Service time variance
L1	5	2.6	M1	5	5.2	H1	5	7.8
L2	10	2.6	M2	10	5.2	H2	10	7.8
L3	15	2.6	M3	15	5.2	H3	15	7.8
L4	20	2.6	M4	20	5.2	H4	20	7.8
L5	25	2.6	M5	25	5.2	H5	25	7.8

normal distributions so that the sampled service times are nonnegative.

We generate the port arrival time A_i of each deep-sea vessel i uniformly from $\{0, 1, \ldots, 48\}$, and set the target departure time to $D_i = A_i + H_i$. We generate the initially scheduled port arrival time S_i of each feeder i uniformly from $\{0, 1, \ldots, 48\}$. A maximum displacement of 12 hours on each feeder arrival time is normally acceptable for port operators. We therefore set the earliest allowed arrival time of each feeder i to $\underline{E}_i = \max\{0, S_i - 12\}$, and set the latest allowed arrival time of each feeder i to $\bar{E}_i = S_i + 12$. Since deep-sea vessels have a higher service priority than feeders, we set the value of C_{1i} to 5 for each $i = 1, \ldots, N_1$, and set the value of C_{2i} to 1 for each $i = 1, \ldots, N_2$.

We consider five different feeder queue length limits by setting $\bar{Q}$ to 5, 10, 15, 20, and 25. As mentioned above, we consider three different feeder service time variances. Thus, there are 15 problem sets. For each problem set, we generate 10 random instances. Hence, there are 150 instances in total. Table 3 summarizes the configurations of the problem sets used in our computational experiments. We have conducted a preliminary computational study to test the convergence of our simulation optimization method on various instances of problem sets L3, M3, and H3 using different combinations of parameter values. For example, we have tested the simulation optimization method by setting s_1 and s_2 between 60 and 140; and setting s_3 between 10 and 35. The parameter values that enable the simulation optimization method to achieve a relatively good balance between effectiveness and efficiency are chosen for the computational experiments. The values of the parameters used in our simulation optimization method are shown in Table 4.

4.2 Benchmark Methods

To evaluate the computational performance of our simulation optimization method, we introduce three benchmark methods and compare the performance of our method with those of the benchmark methods. The first benchmark method, denoted BM1, executes only the global phase of our simulation method. Thus, BM1 is a pure genetic algorithm. Using BM1 as benchmark method enables us to evaluate how much the local phase of our simulation optimization method can improve

Table 4: Parameter values used in the simulation optimization method.

Phase	Parameter	Value
Global phase	s_1	100
	s_2	100
	μ	0.1
	$\bar{\ell}$	60
	n_1	20
	n_2	5
	λ_0	100
	ζ	0.5
Local Phase	s_3	20
	s_4	85
	h	10
	n_3	50
	$\bar{\lambda}$	10^5
Clean-up phase	δ	1
	ϵ	0.05
	n_4	3000

the solutions obtained by the global phase. The second benchmark method, denoted BM2, ignores uncertainties by making use of the mean service times of feeders. Specifically, BM2 executes only the global phase of our simulation method, but it evaluates the performance of each solution using the mean service times of feeders rather than the sampled service times. The purpose of using BM2 as benchmark method is to investigate whether it is important to consider uncertainties of feeders' service times when evaluating the performance of each solution. The third benchmark method, denoted BM3, also executes only the global phase of our simulation method, but it evaluates solutions accurately by consuming a large amount of simulation budget. We use the same simulation budget as in Han et al. (2010), and allocate 750 simulation replications to each solution that needs to be evaluated by BM3. The purpose of using BM3 as benchmark method is to investigate whether the simulation budget allocation strategy (i.e., allocating different amounts of simulation budget to different search phases) of our simulation optimization method is more effective than the traditional one. For each benchmark method, we execute a clean-up phase to choose the best solution among the solutions generated at the last iteration.

4.3 Comparison with Benchmark Methods

In this subsection, we evaluate the computational performance of the simulation optimization method for instances with a medium queue length limit. We solve test instances of problem sets L3, M3, and H3 using the simulation optimization method and the three benchmark methods (i.e., BM1, BM2, and BM3). We run each of the solution methods five times for each test instance, and record the computational results obtained in each run. Table 5 summarizes the computational

Table 5: Computational results for instances of problem sets L3, M3, and H3

	Simulation optimization				BM1				BM2				BM3			
Instance	Min	Max	Avg	Time	Min	Max	Avg	Time	Min	Max	Avg	Time	Min	Max	Avg	Time
L3.1	227	257	245.4	1093.7	281	317	285.2	381.6	277	344	287.0	171.6	231	282	240.2	29859.8
L3.2	217	244	232.8	1710.1	255	304	270.4	458.0	273	342	295.4	106.1	227	270	243.8	28342.0
L3.3	173	199	186.2	1639.2	187	234	212.2	455.0	212	282	225.4	162.8	175	184	179.2	26385.8
L3.4	201	222	211.0	1271.5	222	281	240.4	489.1	269	363	296.8	145.5	198	231	222.0	26180.9
L3.5	294	315	306.4	1555.8	296	381	348.6	364.4	327	479	360.8	97.0	302	352	326.2	27289.3
L3.6	151	186	160.2	1605.8	175	213	185.2	495.4	177	289	218.6	165.3	166	187	181.6	28657.1
L3.7	201	231	217.0	1334.7	215	263	232.6	441.1	216	312	280.4	197.6	195	223	203.4	25809.8
L3.8	145	184	160.4	1376.1	169	257	214.8	525.2	197	272	219.0	199.5	171	238	200.8	26349.4
L3.9	226	252	235.6	1565.6	211	288	257.0	459.3	260	378	340.2	100.7	217	267	240.2	31285.9
L3.10	160	194	179.4	1480.4	180	228	204.6	435.6	171	322	213.6	103.3	168	207	189.4	25525.7
Avg	199.5	228.4	213.4	1463.3	219.1	276.6	245.1	450.5	237.9	338.3	273.7	144.9	205.0	244.1	222.7	27568.6
M3.1	327	376	354.2	1275.9	354	434	412.0	427.9	440	493	456.2	197.0	337	388	377.4	30084.5
M3.2	255	303	277.4	1558.2	280	345	309.4	446.2	300	450	368.0	135.0	257	343	292.2	27420.4
M3.3	286	305	299.0	1683.2	292	351	318.6	442.1	343	423	360.6	154.7	284	329	287.6	24654.0
M3.4	299	319	312.8	1390.3	320	383	335.8	391.5	373	491	417.8	128.0	288	374	312.6	25551.4
M3.5	357	396	368.2	1592.6	386	467	418.0	359.3	388	532	424.6	103.5	351	424	371.2	27359.9
M3.6	252	288	272.6	1691.0	298	340	306.4	524.0	249	341	287.8	165.3	264	313	281.0	27730.6
M3.7	225	265	242.8	1531.2	247	343	271.8	473.9	245	397	330.4	184.0	224	303	251.8	25891.4
M3.8	280	310	297.6	1565.9	309	386	346.0	465.0	343	417	352.2	200.2	313	358	338.4	26420.9
M3.9	238	271	263.2	1377.2	264	336	305.2	451.1	295	384	346.4	125.7	253	303	285.6	31152.8
M3.10	257	286	272.4	1411.3	269	320	294.2	405.7	277	403	317.8	140.2	257	282	274.6	27863.7
Avg	277.6	311.9	296.0	1507.7	301.9	370.5	331.7	438.7	325.3	433.1	366.2	153.4	282.8	341.7	307.2	27413.0
H3.1	488	552	538.2	1593.6	525	628	589.4	472.1	634	671	651.6	149.9	518	557	546.2	29756.8
H3.2	351	423	367.4	1613.5	372	465	404.8	569.0	395	576	455.0	105.4	348	448	390.8	28422.8
H3.3	427	447	436.4	1745.6	440	495	466.2	457.3	462	577	513.2	199.4	415	464	429.4	26520.5
H3.4	497	522	515.8	1922.7	497	604	512.6	480.2	572	725	626.0	132.3	473	562	501.4	26106.6
H3.5	484	556	506.6	1632.2	496	591	523.4	550.3	532	668	552.2	118.8	493	584	507.8	28578.7
H3.6	357	398	385.4	1736.4	377	452	420.2	459.8	355	419	398.8	152.6	366	426	394.4	29151.4
H3.7	342	401	375.8	1880.7	372	521	400.6	463.4	371	536	462.2	180.7	322	454	387.2	24283.9
H3.8	400	434	431.2	1858.0	355	508	478.8	416.1	459	546	477.4	213.5	416	458	429.4	27789.3
H3.9	376	432	415.2	1695.9	419	476	461.4	496.8	463	555	486.4	136.5	397	470	442.6	30654.1
H3.10	423	471	439.4	1670.5	438	501	457.4	542.6	432	549	473.2	119.4	421	462	456.4	26106.7
Avg	414.5	463.6	441.1	1734.9	429.1	524.1	471.5	490.8	467.5	582.2	509.6	150.9	416.9	488.5	448.6	27737.1

results. For each solution method and each test instance, the "Min," "Max," and "Avg" columns in Table 5 report the minimum solution value, the maximum solution value, and the average solution value, respectively, among the five independent runs. The "Time" column reports the average computation time (in seconds) of the five runs.

From Table 5, we observe that the average solution values of the simulation optimization method are smaller than those of BM1. This indicates that the local phase of the simulation optimization method improves the solutions generated by the global phase. We also observe from the "Time" columns that the computation time of the simulation optimization method is much longer than that of BM1. This is due to the fact that in the simulation optimization method, the local phase uses a larger simulation budget than the global phase in order to evaluate the performance of solutions more accurately. The running time of BM2 is much shorter than that of the simulation optimization method. However, the average solution values of BM2 are consistently larger than

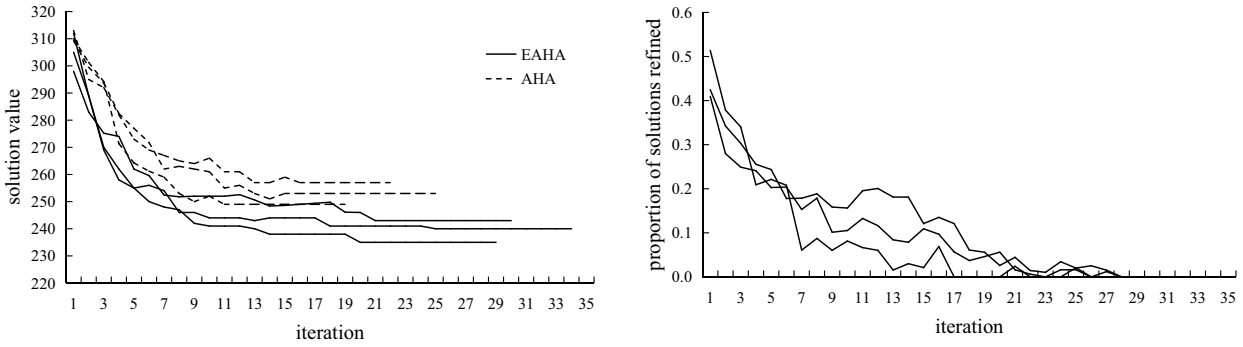
those of the simulation optimization method, indicating that the solutions generated by BM2 are inferior to those generated by the simulation optimization method. This is expected, since the short computation time of BM2 is achieved at the cost of ignoring the uncertainties of feeder service times, which makes the evaluation of solution performance efficient, but which can result in significant errors in the estimation of solution values. On the other hand, BM3 generates better solutions than BM1 and BM2 (see the "Avg" values in Table 5). Since BM3 generates solutions using the same genetic algorithm as that used by BM1 and BM2 but consumes substantially larger computation budget than BM1 and BM2, BM3 estimates solution values more accurately than BM1 and BM2 in each iteration, and is therefore able to generate better solutions. However, because BM3 consumes a large amount of simulation budget, its computation usually requires several hours. Compared to BM3, the simulation optimization method generates better solutions with a considerably smaller amount of computation effort. The reason for the superiority of the proposed simulation optimization method over BM3 is that BM3 is a pure genetic algorithm that searches solutions within a truncated solution space defined by the encoding scheme, whereas the simulation optimization method enhances the genetic algorithm by applying a local search phase that extends the search space using a hyperbox. Hence, the simulation optimization method outperforms the benchmark methods in minimizing the tardiness and displacement cost of vessels.

Another observation from Table 5 is that the solution values obtained by the simulation optimization method in multiple runs do not show high variations, whereas the solution values obtained by the benchmark methods in multiple runs vary considerably. For example, the gap between the "Min" and "Max" values of the simulation optimization method for instance L3.10 is 34, whereas the gaps resulting from BM1, BM2, and BM3 for the same instance are 48, 151, and 39, respectively. This indicates that the simulation optimization method is more robust than the benchmark methods in terms of solution quality. Recall that problem sets L3, M3, and H3 differ from each other only in the variance of feeder service times. Comparing the results obtained for different problem sets, we observe that the solution values tend to increase as the feeder service time variance increases. This is because when the feeders have a higher uncertainty in service time, the solutions tend to reserve longer berthing times for feeders, incurring larger departure tardiness for deep-sea vessels, which increases the overall solution values. The difference in solution values between instances with different feeder service time variances reflects the value of information generated by more accurate feeder service time estimates. Hence, one possible way for the port operator to improve the vessel service is to increase the accuracy of the feeder service time. For example, the port operator can

invest in advanced information technologies, so that more demand information about the feeders can be obtained before the berth planning takes place.

4.4 Effectiveness of the Solution Refinement Subroutine

In the local phase of the simulation optimization method, we enhance the adaptive hyperbox algorithm using a solution refinement subroutine, which aims to restore feasibility of any infeasible solution generated by the adaptive hyperbox algorithm. In this subsection, we compare the performance of the enhanced adaptive hyperbox algorithm (EAHA) with that of the unenhanced adaptive hyperbox algorithm (UAHA). Figure 2 shows the convergence behavior of three EAHA runs and three UAHA runs when solving problem instance M3.1 and the proportion of solutions refined by EAHA in each iteration. In Figure 2, each run of EAHA and UAHA is initialized with a solution cluster generated by the global phase of the simulation optimization method. When using UAHA,



(a) Convergence behavior of EAHA and UAHA

(b) Solutions refined in each iteration of EAHA

Figure 2: EAHA versus UAHA.

each infeasible solution generated by the adaptive hyperbox algorithm is assigned an infinite cost. From Figure 2(a), we observe that EAHA takes more iterations to stop than UAHA, but the solution values generated by EAHA are better than those generated by UAHA. The reason for the relatively poor performance of UAHA is that UAHA frequently generates infeasible solutions in each iteration. These infeasible solutions provide little information useful for constructing a hyperbox that contains good solutions in the next iteration. As can be observed from Figure 2(b), a large proportion of infeasible solutions are generated from the hyperbox at the first iteration, and thus the solution refinement subroutine needs to be executed frequently to improve the solutions. In the later iterations of EAHA, the proportion of infeasible solutions generated from the hyperbox decreases gradually. Hence, the solution refinement subroutine is effective in improving the perfor-

mance of the adaptive hyperbox algorithm when solving the berth allocation and arrival scheduling problem.

4.5 Quay Utilization Rate

In this subsection, we analyze the quay utilization rates of the solutions generated by the simulation optimization method. The quay utilization rate, which is the ratio of the amount of occupied quay space to the amount of available quay space, is an important performance metric for evaluating the productivity of a container port. We compute the average quay utilization rate for each time period of the planning horizon using the solutions generated by simulation optimization for problem set M3. Note that our model adopts a discretized quay space setting and assumes that the length of each vessel is a multiple of the length of each berth segment. This quay space discritization may lead to overestimating the utilization of the quay, since the actual length of each vessel may be shorter than the total length of the berth segments that the vessel needs to occupy. Recall that the length of each berth segment is σ and the number of available berth segments is B. Thus, at any point in time, the amount of available quay space is $B\sigma$. Consider any solution generated by the simulation optimization method. Let $R(t)$ denote the number of berth segments occupied by vessels during the time period $[t, t+1]$. We refer to the ratio $R(t)/B$ as the *nominal quay utilization rate* in period $[t, t+1]$. Let $\rho(t)$ denote the amount of actual quay space occupied by vessels during time period $[t, t+1]$ in this solution, where $\rho(t)$ is obtained by using the vessels' original lengths (i.e., letting the length of each deep-sea vessel i be α_i instead of $\lceil \frac{\alpha_i}{140} \rceil \cdot 140$, and letting the length of each feeder be uniformly distributed in $[75, 140]$). We refer to the ratio $\rho(t)/B\sigma$ as the *actual quay utilization rate* in period $[t, t+1]$. For each instance of M3, we compute the nominal and actual quay utilization rates for the solution. The average nominal (respectively actual) quay utilization rate in each time period $[t, t+1]$ is then obtained by taking the average of the nominal (respectively actual) quay utilization rates in $[t, t+1]$ among all those instances of M3. Figure 3 depicts the computational results.

From Figure 3, we observe that after an initial warm-up period and before the operation approaches the end of the planning horizon, there is a "productive period," during which the average nominal quay utilization rate varies between 0.85 and 1, while the average actual quay utilization rate is about 20% lower than the average nominal quay utilization rate. This indicates that the solutions obtained by discretizing the quay space may lead to overestimating quay utilization. However, the actual quay utilization rate is quite high during the productive period. This implies

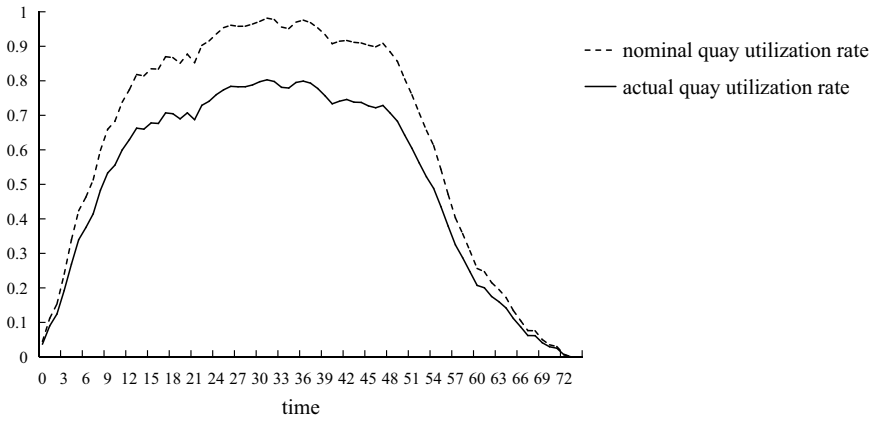


Figure 3: Variation of average quay utilization rate over time.

that substantial queueing of feeders exists in the container port, and thus our model that optimizes the berthing and arrivals of the vessels is beneficial to the port.

4.6 Varying the Queue Length Limit

Next, we investigate how the computational results of the simulation optimization method and the benchmark methods are affected as the upper limit on expected queue length of feeders varies. We consider the test instances with different $\bar{Q}$ values, and we run each of the solution methods five times for each instance. Table 6 summarizes the computational results, where each row reports the average computational results of ten instances in a problem set.

From Table 6, we observe that the average solution values obtained by the simulation optimiza-

Table 6: Results for problem sets with different $\bar{Q}$ values.

Problem set	$\bar{Q}$	Simulation optimization		BM1		BM2		BM3	
		Avg	Time	Avg	Time	Avg	Time	Avg	Time
L1	5	305.3	1445.4	357.8	440.4	384.3	126.3	324.0	27172.4
L2	10	248.0	1477.2	297.4	455.2	409.3	105.4	290.3	25470.7
L3	15	213.4	1463.3	245.1	450.5	273.7	144.9	222.7	27568.6
L4	20	159.2	1486.0	164.3	451.5	232.3	158.2	159.7	28825.9
L5	25	116.8	1504.5	126.3	442.1	201.8	173.2	109.1	28678.3
M1	5	529.8	1503.1	590.2	422.2	655.2	141.6	560.3	28353.8
M2	10	400.0	1521.0	436.0	430.7	650.2	141.1	398.7	25825.2
M3	15	296.0	1507.7	331.7	438.7	366.2	153.4	307.2	27413.0
M4	20	228.6	1495.5	255.5	426.3	285.8	151.1	213.5	28230.0
M5	25	151.4	1462.6	157.2	399.0	228.3	173.2	147.6	24964.7
H1	5	749.7	1819.5	787.8	479.7	859.6	142.1	771.9	30808.7
H2	10	545.5	1775.1	575.3	487.2	779.4	148.7	526.1	29837.5
H3	15	441.1	1734.9	471.5	490.8	509.6	150.9	448.7	27737.1
H4	20	366.3	1706.7	396.7	478.0	434.5	142.3	373.9	28573.2
H5	25	291.1	1486.4	321.5	372.9	359.2	169.9	298.6	29792.8

tion method are smaller than those obtained by BM1 for different $\bar{Q}$ values. This shows that the local phase of the simulation optimization method improves the solutions obtained by the global phase for various queue length limits. However, as the value of $\bar{Q}$ increases, the improvement in the solution value generated by the local phase tends to diminish. For example, the improvement in the average solution value obtained by the local phase is 52.5 for problem set L1, where $\bar{Q}$ is set equal to 5. However, the improvement is only 9.5 for problem set L5, where $\bar{Q}$ is set equal to 25. One possible reason for this tendency is that as the queue length limit increases, constraint (7) of problem **P** becomes less restrictive, increasing the chance of identifying good solutions in the global phase of the simulation optimization method. The average solution values obtained by BM2 are consistently worse than those obtained by the simulation optimization method, showing that the solutions obtained by ignoring uncertainties are inferior under different queue length limits. On the other hand, the average solution values obtained by BM3 are worse than those obtained by the simulation optimization method when $\bar{Q}$ is small, but are comparable to those obtained by the simulation optimization method when $\bar{Q}$ becomes larger. However, the computation time of BM3 is much longer than that of the simulation optimization method. Overall, the simulation optimization method outperforms the benchmark methods for different queue length limits.

Figure 4 depicts the average objective values obtained by the simulation optimization method for problem sets with different queue length limits and different feeders' service time variances. From Figure 4, we observe that the average objective value decreases nearly linearly as the queue length limit increases, and that the rate of decrease is more significant when the feeders' service time variance is higher. Thus, the port operator can improve the vessel service by increasing the queue length limit through enlarging the capacities of the feeder waiting areas. This improvement can be significant when the feeders' service times have a high variation.

4.7 Benefits of Controlling the Queue Length of Feeders

As mentioned in Section 1, port operators usually serve deep-sea vessels by ignoring the uncertainties of feeder service times (i.e., they develop detailed berth plans for deep-sea vessels based on deterministic arrival and service times of deep-sea vessels) and allocate berths to feeders dynamically when feeders arrive at the port. This service strategy can achieve good service levels for deep-sea vessels, as berth space is reserved exclusively for deep-sea vessels. However, this strategy cannot control the waiting times of feeders and is unable to mitigate port congestion incurred by feeder traffic. Unlike the service strategy adopted by port operators, our simulation optimization

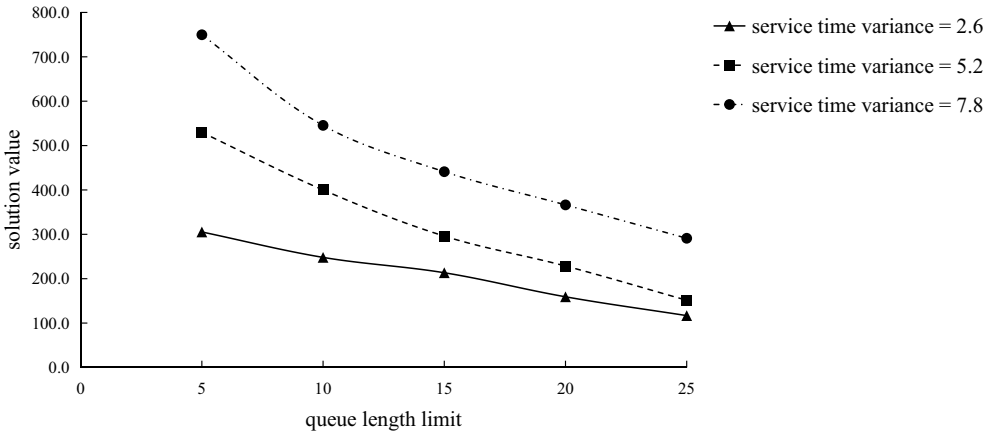


Figure 4: Average objective values obtained by simulation optimization.

method generates berth plans for deep-sea vessels by taking into account feeder service times, and controls both port congestion and feeder waiting times by restricting the expected queue length of feeders. In this subsection, we compare the performance of the simulation optimization method with the current practice of port operators, and reveal the benefits of controlling the queue length of feeders.

We mimic the current practice of port operators using a sequential decision heuristic. In the current practice, the planners ignore the queue length limit of feeders, which is equivalent to setting the value of $\bar{Q}$ in constraint (7) to infinity. Note that after setting $\bar{Q}$ equal to infinity, problem **P** decomposes into two subproblems. One subproblem involves only the deep-sea vessels, while the other subproblem involves only the feeders. The sequential decision heuristic first solves a deterministic berth allocation model of the deep-sea vessels, which involves only the x_{ibt} and l_{1i} variables, and then allocates berths to feeders according to the first-come first-served discipline described in Section 2. Note that in the first-come first-served discipline, a berth segment can be allocated to a feeder only if the berth segment is not reserved for deep-sea vessels. Hence, there are no stochastic delays on the deep-sea vessels; that is, the departure tardiness cost of deep-sea vessels is determined solely by the x_{ibt} values. A detailed description of the sequential decision heuristic is provided in the Appendix. We solve problem sets M1, M2, M3, M4, and M5 using the sequential decision heuristic, and compare the solutions obtained by the sequential decision heuristic with those obtained by the simulation optimization method using the following performance measures: (i) average departure tardiness of deep-sea vessels (AT); (ii) average schedule displacement of feeders (AD); (iii) peak expected queue length of feeders (PQL); (iv) average waiting time of

feeders (AWT); and (v) objective value (OV). Note that since the port is set to be empty at the beginning of the planning horizon, there may exist a warm-up period in the statistics of the feeder waiting times. We apply Fishman's graphical method (Fishman 2001) to identify the warm-up period; that is, we choose a suitable warm-up period by visual inspection on the variation of feeder waiting times. The warm-up period identified is $[0, 15]$. When computing the AWT value, we exclude the statistics in this warm-up period. Table 7 compares the computational results obtained by the simulation optimization method with those obtained by the sequential decision heuristic for the instances in problem set M3. All time-related values reported in Table 7 are in hours.

Table 7: Comparison of simulation optimization and sequential heuristic.

	Simulation optimization					Sequential heuristic				
Instance	AT	AD	PQL	AWT	OV	AT	AD	PQL	AWT	OV
M3.1	1.8	1.6	11.2	2.8	354.2	0.4	0.0	35.4	10.2	50.0
M3.2	1.4	1.3	14.5	4.2	277.4	0.6	0.0	42.1	12.7	75.0
M3.3	1.2	1.8	13.3	2.9	299.0	0.4	0.0	39.9	10.9	50.0
M3.4	1.8	1.1	12.1	2.5	312.8	0.2	0.0	42.5	15.7	25.0
M3.5	2.0	1.4	14.8	4.3	368.2	0.0	0.0	36.2	10.3	0.0
M3.6	1.4	1.2	14.8	5.2	272.6	0.6	0.0	45.7	15.5	75.0
M3.7	1.2	1.2	13.8	3.1	242.8	0.2	0.0	32.8	13.5	25.0
M3.8	1.4	1.5	12.4	2.2	297.6	0.8	0.0	57.6	16.3	100.0
M3.9	1.0	1.7	14.1	3.6	263.2	0.0	0.0	50.9	15.1	0.0
M3.10	1.4	1.2	14.2	3.4	272.4	0.2	0.0	48.3	16.0	25.0
Average	1.5	1.4	13.5	3.4	296.0	0.3	0.0	43.1	13.6	42.5

From Table 7, we observe that the average AT and AD values of the sequential decision heuristic are 0.3 and 0.0, respectively, indicating that the sequential decision heuristic performs well in minimizing the departure tardiness of deep-sea vessels and the schedule displacements of feeders. The zero AD values are due to the fact that the sequential decision heuristic reserves berth space to deep-sea vessels without considering the berth utilization of feeders, and thus the heuristic does not need to alter the initial arrival plans of feeders (i.e., it simply assigns each feeder i the arrival time S_i). Without reserving berth space to feeders, the heuristic can obtain solutions with small objective values; see the OV values of the sequential heuristic in Table 7. However, since the sequential decision heuristic ignores feeders' queuing behavior, the service plan generated by the heuristic can lead to large feeder queue length and long waiting times of feeders, as indicated by the PQL and AWT values of the heuristic. The PQL values of the sequential decision heuristic exceed the value of $\bar{Q}$, indicating that the sequential decisions generate solutions that consistently violate the maximum expected queue length constraint. Compared to the sequential decision heuristic, the simulation optimization method has larger AT and AD values, and thus larger OV values. However, the PQL and AWT values of the simulation optimization method are much smaller than

those of the sequential decision heuristic. This shows that the simulation optimization method generates solutions that strike a balance between congestion mitigation and deep-sea vessel service quality, and performs much better than the sequential decision method in reducing feeder waiting times.

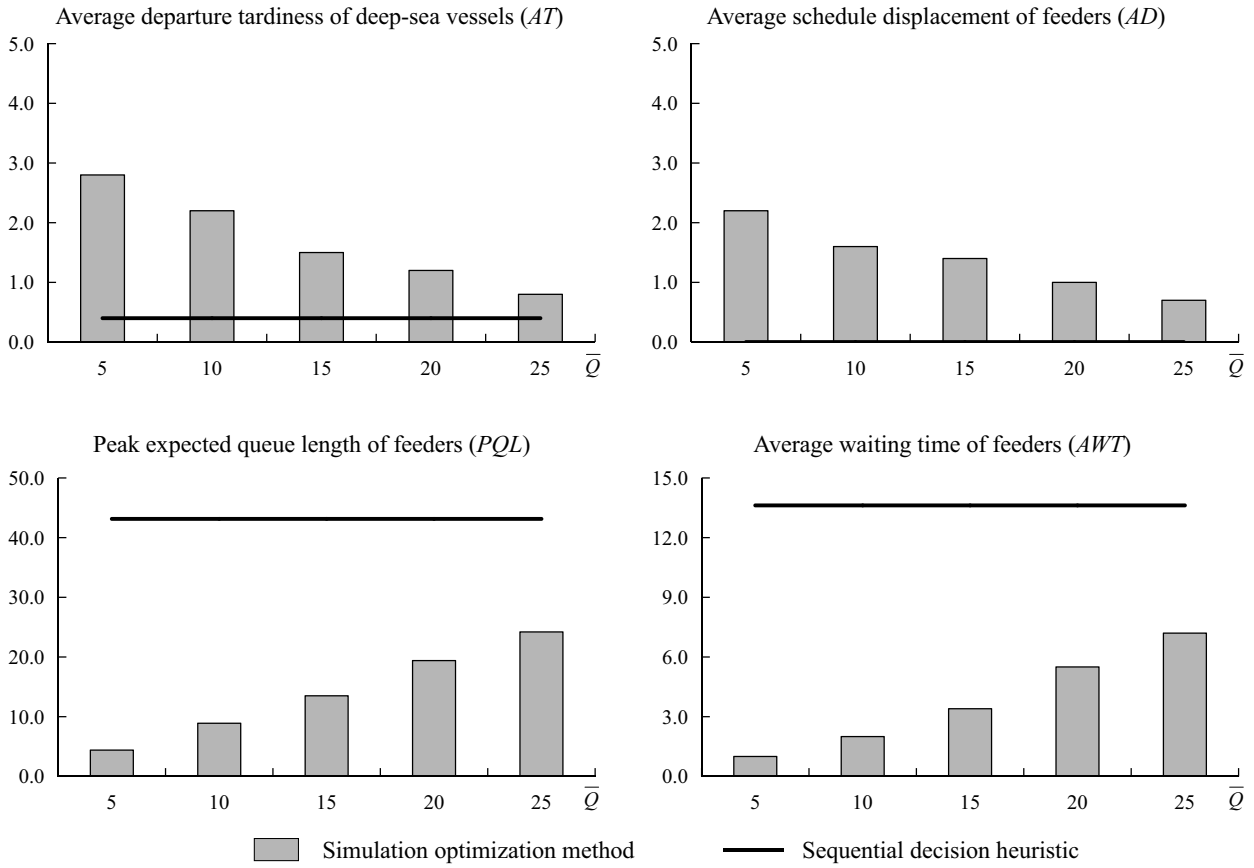


Figure 5: Performance measures of solutions obtained for different $\bar{Q}$ values.

Figure 5 depicts the average performance measures obtained by the simulation optimization method for different $\bar{Q}$ values, as well as performance measures obtained by the sequential decision heuristic. From Figure 5, we observe that the average departure tardiness of deep-sea vessels and the average schedule displacement of feeders generated by the simulation optimization method tend to decrease as $\bar{Q}$ increases, whereas the peak expected queue length of feeders and the average waiting time of feeders tend to increase as $\bar{Q}$ increases. On the other hand, since the sequential decision heuristic ignores the queue length restriction of feeders, its computational results are independent of the $\bar{Q}$ values. The comparisons show that the simulation optimization method is more flexible than the sequential decision heuristic in controlling the expected queue length of feeders, enabling

terminal operators to quantify the trade-off between alleviating port congestion and enhancing vessel service quality. For container ports that serve a large number of feeders (e.g., the Port of Shanghai), long waiting lines of feeders can result in severe congestion in the port basin, impeding the service of vessels and creating a high risk of vessel collisions. Hence, the simulation optimization method would be more preferable than the current practice of port operators for berth allocation and congestion mitigation.

5 Conclusions

This paper studies the problem of how to optimize the berth allocation of deep-sea vessels and schedule arrivals of feeders for a container port where the service times of feeders are uncertain and port congestion needs to be kept under control. We develop a stochastic optimization model for the problem. Our model allocates berth space to deep-sea vessels and schedules the arrivals of feeders, subject to a constraint on the expected feeder queue length. We solve the stochastic optimization model using a simulation optimization method that searches the solution space via a global phase, a local phase, and a clean-up phase, and allocates different amounts of simulation budget to different search phases in order to balance the effort spent on solution evaluation and solution sampling. Unlike the existing simulation optimization methods for solving berth allocation problems, which either allocate a large simulation budget to each solution and incur a heavy computation burden or attempt to explore the solution space efficiently by ignoring uncertainties and then use simulation to evaluate only a limited number of the visited solutions, our simulation optimization method is designed to strike a balance between exploration and exploitation, and is able to generate a locally optimal solution with a reasonable amount of computation effort.

We generate problem instances based on the operational data of the Yangshan Deep-water Port in Shanghai, and compare the computational performance of the simulation optimization method with those of three benchmark methods that only make use of genetic algorithm for searching solutions but apply different simulation budget allocation strategies. The computational results indicate that for the solution sampling strategy employed by the benchmark methods, allocating a larger simulation budget to solution evaluation improves the capability of generating good solutions but leads to considerably stronger demand for computation effort, and vice versa. On the other hand, our simulation optimization method finds good solutions with a reasonable amount of computation effort and thus outperforms the benchmark methods. We also compare the solutions obtained by the simulation optimization method with those obtained by a sequential decision

heuristic that mimics the current practice of port operators. The comparisons show that the simulation optimization method provides the flexibility of allocating berth space to deep-sea vessels and scheduling the arrivals of feeders under different queue length limits, whereas the sequential decision heuristic is incapable of controlling the queue length of feeders. Hence, the simulation optimization method would be a promising decision support tool for berth allocation and congestion mitigation in a port that serves a large number of feeders.

Our solution method can be modified easily to handle different extensions of the problem. First, consider the situation where some vessels are being served and some feeders are waiting for service at the beginning of the planning horizon. Having some vessels being served at time 0 implies that some berth segments are unavailable for some initial time periods. We can modify our simulation optimization method to handle this situation. This is done by modifying the decoding scheme in such a way that allocating berth segments to vessels during the berth segments' unavailable period is disallowed. Having some feeders waiting for service at time 0 implies that those arrived feeders have known service times, and thus the simulator must be modified accordingly. Second, our model assumes that the planning horizon is long enough so that all vessels can complete service during the planning horizon. In practice, however, the planning horizon considered by berth planners may not be sufficiently long, and thus, some vessels may need to be rejected for service during the planning horizon. A straightforward extension of our solution method to account for service rejections is to generate berth plans of all vessels using the extended planning horizon, and then reject those vessels whose expected service completion times exceed the "actual" planning horizon. Third, our model assumes that each feeder occupies one berth segment and that each deep-sea vessel's length is a multiple of the length of a berth segment. Consider the situation where the feeder lengths are non-identical and a feeder can occupy multiple berth segments. In this situation, the first-come first-served service rule described in Section 2 needs to be modified to ensure that a feeder can only be served when there are sufficient consecutive berth segments available to accommodate the feeder. Our simulation optimization method can be applied to this situation if we modify the decoding scheme by allocating berth segments to a feeder only when there are sufficient consecutive berth segments available, and modify the simulator so that the feeders are assigned to berth segments according to the adapted first-come first-served service rule. Fourth, our model assumes that there are no space restrictions; that is, each berth segment is compatible with all vessels, regardless of the vessels' draft and width. For solving a problem with space restrictions, our simulation optimization method should be modified so that each vessel is

assigned to compatible berth segments only. Specifically, in Step 2 of the decoding scheme, a deep-sea vessel p_i can be assigned to berth segments $b, b+1, \ldots, b+R_{p_i}-1$ only if these berth segments are all compatible with vessel p_i. In the adaptive hyperbox algorithm, if a solution generated from the hyperbox violates the compatibility constraint, then the solution is infeasible and assigned an infinite cost. In the simulator, each feeder can be assigned to an available berth segment only if the berth segment is compatible with the feeder. Fifth, in our model an upper limit is imposed on the expected queue length of feeders. Another way to control the port traffic is to set an upper limit on the probability that the queue length exceeds a risk limit; that is, replace constraint (7) by the chance constraint "$Prob(Q_t(\mathbf{x},\mathbf{y}) \geq \bar{Q}) \leq \varrho$," where ϱ is the risk limit. For solving the chance constrained problem, we can introduce an indicator function $\Phi_t(\mathbf{x},\mathbf{y})$ for each $t = 0, 1, \ldots, T-1$, where $\Phi_t(\mathbf{x},\mathbf{y}) = 1$ if $Q_t(\mathbf{x},\mathbf{y}) \geq \bar{Q}$, and $\Phi_t(\mathbf{x},\mathbf{y}) = 0$ otherwise. Then, the chance constraint can be transformed into the constraint $E[\Phi_t(\mathbf{x},\mathbf{y})] \leq \varrho$, which is of the same form as constraint (7). Through this transformation, the chance constrained problem can also be solved by our simulation optimization method. Finally, we remark that by modifying the implementation of the simulator, our simulation optimization method is also applicable to the situation where feeders are served according to particular service rules other than the first-come first-served rule.

Future research could focus on developing adaptations of the simulation optimization method for solving a variety of port operation management problems with uncertainties. For example, in a yard crane scheduling problem, the yard cranes need to serve container trucks that pick up containers from or deliver containers to the yard. Since the arrival times of the container trucks are uncertain to the port operator, it would be important to take into account the stochastic arrivals of container trucks when scheduling the yard cranes. To solve different port operation management problems with uncertainties, the solution sampling strategy used in the global phase and the local phase can be tailored to the particular problem in order to explore the solution space efficiently. The simulation budget allocation strategy (i.e., allocating different amounts of simulation budget to solutions in different search phases) will therefore still be useful for enhancing the computational efficiency.

Acknowledgments

The authors thank Professor Jeff L. Hong and four anonymous referees for their valuable comments that improved the paper. This work was done when the first author was a Ph.D. student at the Department of Logistics and Maritime Studies, The Hong Kong Polytechnic University.

References

Alvarez J.F., Longva T., Engebrethsen E.S., 2010. A methodology to assess vessel berthing and speed optimization policies. Maritime Economics & Logistics 12 (4), 327–346.

Amaran S., Sahinidis N.V., Sharda B., Bury S.J., 2016. Simulation optimization: A review of algorithms and applications. Annals of Operations Research 240 (1), 351–380.

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

Arango C., Cortés P., Onieva L., Escudero A., 2013. Simulation-optimization models for the dynamic berth allocation problem. Computer-Aided Civil and Infrastructure Engineering 28 (10), 769–779.

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

Bierwirth C., 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), 675–689.

Boesel J., Nelson B.L., Kim S.-H., 2003. Using ranking and selection to "clean up" after simulation optimization. Operations Research 51 (5), 814–825.

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

De A., Pratap S., Kumar A., Tiwari M.K., 2020. A hybrid dynamic berth allocation planning problem with fuel costs considerations for container terminal port using chemical reaction optimization approach. Annals of Operations Research 290, 783–811.

Dragović B., Park N.K., Radmilović Z., 2006. Ship-berth link performance evaluation: Simulation and analytical approaches. Maritime Policy & Management 33 (3), 281–299.

Dragović B., Park N.K., Radmilović Z., Maraš V., 2005. Simulation modelling of ship-berth link with priority service. Maritime Economics & Logistics 7 (4), 316–335.

Du Y., Chen Q., Lam J.S.L., Xu Y., Cao J.X., 2015. Modeling the impacts of tides and the virtual arrival policy in berth allocation. Transportation Science 49 (4), 939–956.

Easa S.M., 1987. Approximate queueing models for analyzing harbor terminal operations. Transportation Research Part B 21 (4), 269–286.

Emde S., Boysen N., 2016. Berth allocation in container terminals that service feeder ships and deep-sea vessels. Journal of the Operational Research Society 67 (4), 551–563.

Fishman G.S., 2001. Discrete-Event Simulation: Modeling, Programming, and Analysis. (Springer,

New York).

Fu M.C., Glover F.W., April J., 2005. Simulation optimization: A review, new developments, and applications. Kuhl M.E., Steiger N.M., Armstrong F.B., Joines J.A., eds. Proceedings of the 2005 Winter simulation conference, Orlando, FL, pp. 83–95.

Garey M.R., Johnson D.S., 1979. Computers and Intractability: A Guide to the Theory of NP-Completeness. (Freeman, New York).

Gharehgozli A.H., Roy D., de Koster R., 2016. Sea container terminals: New technologies and OR models. Maritime Economics & Logistics 18 (2), 103–140.

Golias M.M., 2011. A bi-objective berth allocation formulation to account for vessel handling time uncertainty. Maritime Economics & Logistics 13 (4), 419–441.

Golias M.M., Saharidis G.K., Boile M., Theofanis S., Ierapetritou M.G., 2009. The berth allocation problem: Optimizing vessel arrival time. Maritime Economics & Logistics 11 (4), 358–377.

Han X.-L., Lu Z.-Q., Xi L.-F., 2010. A proactive approach for simultaneous berth and quay crane scheduling problem with stochastic arrival and handling time. European Journal of Operational Research 207 (3), 1327–1340.

Hong L.J., Nelson B.L., 2006. Discrete optimization via simulation using COMPASS. Operations Research 54 (1), 115–129.

Iris Ç., Lalla-Ruiz E., Lam J.S.L., Voß S., 2018. Mathematical programming formulations for the strategic berth template problem. Computers & Industrial Engineering 124, 167–179.

Iris Ç., Lam J.S.L., 2019. Recoverable robustness in weekly berth and quay crane planning. Transportation Research Part B 122, 365–389.

Jacquillat A., Odoni A.R., 2015. An integrated scheduling and operations approach to airport congestion mitigation. Operations Research 63 (6), 1390–1410.

Jin J.G., Lee D.-H., Hu H., 2015. Tactical berth and yard template design at container transshipment terminals: A column generation based approach. Transportation Research Part E 73, 168–184.

Karafa J., Golias M.M., Ivey S., Saharidis G.K.D., Leonardos N., 2013. The berth allocation problem with stochastic vessel handling times. The International Journal of Advanced Manufacturing Technology 65 (1–4), 473–484.

Kim K.H., Lee H., 2015. Container terminal operation: Current trends and future challenges. Lee C.-Y., Meng Q., eds. Handbook of Ocean Container Transport Logistics (Springer, Switzerland), pp. 43–73.

Lang N., Veenstra A., 2010. A quantitative analysis of container vessel arrival planning strategies. OR Spectrum 32 (3), 477–499.

Lee D.-H., Jin J.G., 2013. Feeder vessel management at container transshipment terminals. Transportation Research Part E 49, 201–216.

Lee L.H., Chew E.P., Manikam P., 2006. A general framework on the simulation-based optimization under fixed computing budget. European Journal of Operational Research 174 (3), 1828–1841.

Legato P., Mazza R.M., 2001. Berth planning and resources optimisation at a container terminal via discrete event simulation. European Journal of Operational Research 133 (3), 537–547.

Legato P., Mazza R.M., Gullì D., 2014. Integrating tactical and operational berth allocation decisions via Simulation-Optimization. Computers & Industrial Engineering 78, 84–94.

Li C.-L., Pang K.-W., 2011. An integrated model for ship routing and berth allocation. International Journal of Shipping and Transport Logistics 3 (3), 245–260.

Li Q., Lam J.S.L., 2017. Conflict resolution for enhancing shipping safety and improving navigational traffic within a seaport: Vessel arrival scheduling. Transportmetrica A: Transport Science 13 (8), 727–741.

Liu C., Xiang X., Zheng L., 2020. A two-stage robust optimization approach for the berth allocation problem under uncertainty. Flexible Services and Manufacturing Journal 32 (2), 425–452.

Moorthy R., Teo C.P., 2006. Berth management in container terminal: The template design problem. OR Spectrum 28 (4), 495–518.

Pang K.-W., Liu J., 2014. An integrated model for ship routing with transshipment and berth allocation. IIE Transactions 46 (12), 1357–1370.

Pang K.-W., Xu Z., Li C.-L., 2011. Ship routing problem with berthing time clash avoidance constraints. International Journal of Production Economics 131 (2), 752–762.

Radmilovich Z.R., 1992. Ship-berth link as bulk queueing system in ports. Journal of Waterway, Port, Coastal, and Ocean Engineering 118 (5), 474–495.

Shang X.T., Cao J.X., Ren J., 2016. A robust optimization approach to the integrated berth allocation and quay crane assignment problem. Transportation Research Part E 94, 44–65.

Steenken D., Voß S., Stahlbock R., 2004. Container terminal operation and operations research—a classification and literature review. OR Spectrum 26 (1), 3–49.

Umang N., Bierlaire M., Erera A.L., 2017. Real-time management of berth allocation with stochastic arrival and handling times. Journal of Scheduling 20 (1), 67–83.

Ursavas E., Zhu S.X., 2016. Optimal policies for the berth allocation problem under stochastic

nature. European Journal of Operational Research 255 (2), 380–387.

Venturini G., Iris Ç., Kontovas C.A., Larsen A., 2017. The multi-port berth allocation problem with speed optimization and emission considerations. Transportation Research Part D 54, 142–159.

Xiang X., Liu C., Miao L., 2017. A bi-objective robust model for berth allocation scheduling under uncertainty. Transportation Research Part E 106, 294–319.

Xiang X., Liu C., Miao L., 2018. Reactive strategy for discrete berth allocation and quay crane assignment problems under uncertainty. Computers & Industrial Engineering 126, 196–216.

Xu J., Huang E., Chen C.-H., Lee L.H., 2015. Simulation optimization: A review and exploration in the new era of cloud computing and big data. Asia-Pacific Journal of Operational Research 32 (3), 1550019 (34 pages).

Xu J., Nelson B.L., Hong L.J., 2010. Industrial strength COMPASS: A comprehensive algorithm and software for optimization via simulation. ACM Transactions on Modeling and Computer Simulation 20 (1), 3:1–3:29.

Xu J., Nelson B.L., Hong L.J., 2013. An adaptive hyperbox algorithm for high-dimensional discrete optimization via simulation problems. INFORMS Journal on Computing 25 (1), 133–146.

Xu Y., Chen Q., Quan X., 2012. Robust berth scheduling with uncertain vessel delay and handling time. Annals of Operations Research 192, 123–140.

Zhen L., Lee L.H., Chew E.P., 2011. A decision model for berth allocation under uncertainty. European Journal of Operational Research 212 (1), 54–68.

Zrnić D.N., Dragović B.M., Radmilović Z.R., 1999. Anchorage-ship-berth link as multiple server queuing system. Journal of Waterway, Port, Coastal, and Ocean Engineering 125 (5), 232–240.

Appendix

Proof of Property 1:

Consider the situation where problem **P** is feasible. Note that in each problem instance of **P**, there are finitely many possible $(\mathbf{x}, \mathbf{y})$ values. Let

$$\Delta_{\min} = \min_{(\mathbf{x},\mathbf{y}) \text{ s.t. } \kappa(\mathbf{x},\mathbf{y})>\bar{Q}} \big\{\kappa(\mathbf{x}, \mathbf{y}) - \bar{Q}\big\},$$

where

$$\kappa(\mathbf{x}, \mathbf{y}) = \max_{t=1,\ldots,T-1} \big\{E[Q_t(\mathbf{x}, \mathbf{y})]\big\}.$$

Then, $\Delta_{\min} > 0$. Let Z^* denote the optimal solution value of problem **P**, and let

$$\hat{\lambda} = \frac{Z^*}{\Delta_{\min}} + 1.$$

Because any feasible solution of **P** is a feasible solution of $\mathbf{P}'(\lambda)$ with $\Delta = 0$, the optimal solution value of $\mathbf{P}'(\lambda)$ is at most Z^*, for any $\lambda \geq 0$.

Consider a feasible solution of $\mathbf{P}'(\lambda)$ with $\lambda \geq \hat{\lambda}$ and $\Delta > 0$. In this feasible solution, $\Delta \geq \Delta_{\min}$. The objective function value of this feasible solution is at least $\lambda\Delta \geq \hat{\lambda}\Delta_{\min} > Z^*$. Thus, this feasible solution is not optimal. Hence, any optimal solution of $\mathbf{P}'(\lambda)$ with $\lambda \geq \hat{\lambda}$ must satisfy the condition that $\Delta = 0$. Therefore, an optimal solution of $\mathbf{P}'(\lambda)$ with $\lambda \geq \hat{\lambda}$ is also optimal to **P**. ∎

Solution Refinement Subroutine:

In the following, we provide details of the solution refinement subroutine used in our implementation of the adaptive hyperbox algorithm. The subroutine requires a solution $(\mathbf{x}, \mathbf{y})$ as input. For ease of presentation, we make use of the following notations:

$\mathcal{P}$: Set of (b, t) pairs for which constraint (4) is violated.

$\mathcal{N}_{bt}$: Set of deep-sea vessels that occupy berth segment b during time period $[t, t + 1]$.

ω_{bt}: Number of deep-sea vessels that occupy berth segment b during time period $[t, t + 1]$.

$\bar{b}_i$: The first berth segment allocated to deep-sea vessel i.

$\bar{t}_i$: Service start time allocated to deep-sea vessel i.

The solution refinement subroutine is presented as follows:

Step 1 (Initialization): For each $i = 1, \ldots, N_1$, set $\bar{b}_i \leftarrow \min\big\{b \mid \sum_{t=0}^{T-H_i} x_{ibt} = 1;\ b = 1, \ldots, B - R_i + 1\big\}$ and $\bar{t}_i \leftarrow \min\big\{t \mid x_{i\bar{b}_i t} = 1;\ t = 0, 1, \ldots, T - 1\big\}$. Set $\mathcal{P} \leftarrow \emptyset$. For each $b = 1, \ldots, B$ and $t = 0, 1, \ldots, T-1$, set $\mathcal{N}_{bt} \leftarrow \emptyset$; in addition, if constraint (4) is violated for b and t, then set $\mathcal{P} \leftarrow \mathcal{P} \cup \{(b, t)\}$.

Step 2 (Resolving infeasibility): If $\mathcal{P} = \emptyset$, then stop. Otherwise, randomly select an element from $\mathcal{P}$, with all elements having equal probability of being selected. Let (b, t) be the selected element.

Step 2.1: For each $i = 1, \ldots, N_1$ such that $\bar{b}_i \leq b \leq \bar{b}_i + R_i - 1$ and $\bar{t}_i \leq t \leq \bar{t}_i + H_i - 1$, set $\mathcal{N}_{bt} \leftarrow \mathcal{N}_{bt} \cup \{i\}$.

Step 2.2: Randomly select an element from $\mathcal{N}_{bt}$, with all elements of $\mathcal{N}_{bt}$ having equal probability of being selected. Let i be the selected element. For each $b = 1, \ldots, B$ and $t = 0, 1, \ldots, T-1$, set $\omega_{bt} \leftarrow \sum_{i' \in \{1,\ldots,N_1\} \setminus \{i\}} \sum_{b'=\max\{b-R_{i'}+1,1\}}^{b} \sum_{t'=\max\{t-H_{i'}+1,0\}}^{t} x_{i'b't'}$.

Step 2.3: Set $\bar{t}_i \leftarrow \min\big\{t \mid \sum_{b'=\bar{b}_i}^{\bar{b}_i+R_i-1} \sum_{t'=t}^{t+H_i-1} \omega_{b't'} = 0;\ t = A_i, A_i+1, \ldots, T-H_i+1\big\}$. For each $t = 0, 1, \ldots, T-1$, if $t = \bar{t}_i$, then set $x_{i\bar{b}_i t} \leftarrow 1$; otherwise, set $x_{i\bar{b}_i t} \leftarrow 0$. Set $\mathcal{N}_{bt} \leftarrow \mathcal{N}_{bt} \setminus \{i\}$. If $|\mathcal{N}_{bt}| \leq 1$, then set $\mathcal{P} \leftarrow \mathcal{P} \setminus \big\{(b, t)\big\}$ and repeat Step 2. Otherwise, go to Step 2.2.

Step 1 initializes the set $\mathcal{P}$, and Step 2 handles the (b, t) pairs in $\mathcal{P}$ one at a time. Step 2.1 determines the set $\mathcal{N}_{bt}$ in the current solution. Steps 2.2 and 2.3 iteratively re-determine the start times of the deep-sea vessels in $\mathcal{N}_{bt}$ one at a time until the set $\mathcal{N}_{bt}$ contains no more than one element.

Sequential Decision Heuristic:

In the following, we describe the sequential decision heuristic used in Section 4.7. The sequential decision heuristic first solves a deterministic berth allocation model of the deep-sea vessels, which involves only the x_{ibt} and l_{1i} variables, and then allocates berths to feeders according to the first-come first-served discipline described in Section 2. The deterministic berth allocation model of the deep-sea vessels is presented as follows:

$$\text{minimize} \quad \sum_{i=1}^{N_1} C_{1i} l_{1i}$$

$$\text{subject to (2), (4), (5), (8)}$$

We solve the deterministic berth allocation model using CPLEX to obtain the total tardiness cost of deep-sea vessels and the values of the x_{ibt} variables. To generate a service plan $(\mathbf{x}, \mathbf{y})$, we need the values of the y_{it} variables. For each $i = 1, \ldots, N_2$ and $t = 0, 1, \ldots, T-1$, we set $y_{it} = 1$ if $t = S_i$, and set $y_{it} = 0$ otherwise. With the value of $(\mathbf{x}, \mathbf{y})$, we run n_4 simulation replications to evaluate the expected queue lengths of feeders and the average waiting times of feeders.