Flex Serv Manuf J (2013) 25:576–608
DOI 10.1007/s10696-012-9135-6

A scheduling problem for a novel container transport system: a case of mobile harbor operation schedule

Hochang Nam · Taesik Lee

Published online: 26 January 2012

Abstract Mobile Harbor (MH) is a movable floating platform with a container handling system on board so that it can load/discharge containers to/from an anchored container ship in the open sea. As with typical quay crane operation, an efficient schedule for its operation is a key to enhancing its operational productivity. A MH operation scheduling problem is to determine a timed sequence of loading/discharging tasks, assignment of MH units to each task, and their docking position, with an objective of minimizing the makespan of a series of incoming container ships. A mixed integer programming model is formulated to formally define the problem. As a practical solution method to the problem, this paper proposes a rule-based algorithm and a random key based genetic algorithm (rkGA). Computational results show that the rkGA method produces a better-quality solution than the rule-based method, while requiring longer computation time.

Keywords Random key based genetic algorithm · Mobile harbor · Operation scheduling problem · Quay crane scheduling · Vehicle routing problem · Genetic algorithm · Mixed integer programming

1 Introduction

1.1 Open sea container loading and discharging operation

In response to the continued increase in the volume of worldwide maritime container transport, container ports face an increasing demand for their service capacity. Many terminal operators respond to this challenge by extending their

H. Nam · T. Lee (✉)
Department of Industrial and Systems Engineering, Korea Advanced Institute of Science and Technology, 335 Gwahangno, Yuseong-Gu, Taejon, Republic of Korea
e-mail: Taesik.lee@kaist.edu

infrastructure (Meisel 2009). The approaches include building new terminals, expanding the existing terminals, constructing or upgrading to faster equipment and optimizing port operations.

As an alternative to these approaches, various non-traditional offshore container port service concepts have been proposed (Kim and Morrison 2011). One such alternative solution is loading and discharging in the open sea. This type of solution is called *mid-stream operation*. Mid-stream operation has been in service at Hong Kong container port for many years. It uses barges with its own derrick crane (Fig. 1) to provide offshore container handling services. Mid-stream operation is reported to handle about 10% of the total container volume at Hong Kong port (HKMOA 2009). While mid-stream operation's utility is acknowledged—especially under severe congestion, this system has been criticized for its looser safety standard and lower quality of container handling than container terminals do. Recently, a group of researchers at Korea Advanced Institute of Science and Technology (KAIST) has developed a new system, called Mobile Harbor (MH) (Suh 2008). MH resolves the weaknesses of existing mid-stream operation by deploying advanced technologies. Similar to the mid-stream operation, MH units go out to a container ship anchoring in the open sea, to (un)load containers on sea and take them to their destination.

A basic MH model has the maximum container carrying capacity of 250 TEU. This model, called MH-250, is shown in Fig. 2, and Table 1 lists its physical dimensions. A MH unit is a specially designed floating platform equipped with a

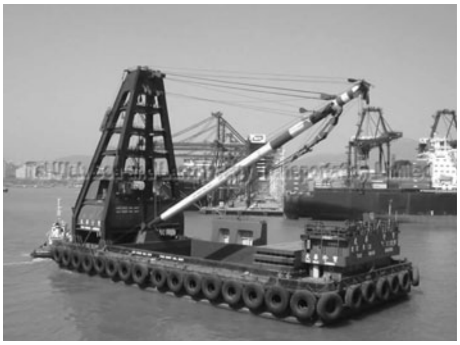


Fig. 1 An example of Mid-stream operation (TS & LHT 2011)

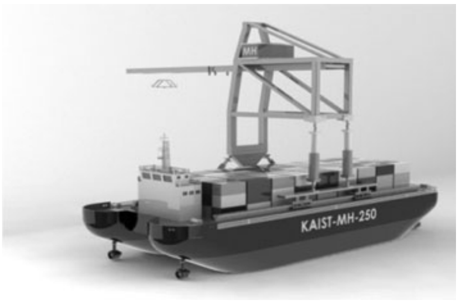


Fig. 2 A 250 TEU Mobile Harbor unit (MH-250)

Table 1 Physical dimension of 250 TEU MH unit

	Length (m)	Width (m)	Height (m)	Mooring depth (m)	Crane height (m)	Crane outreach (m)	Container storage area (40 ft)	Maximum container carrying capacity (TEU)
Design value	77	33	11	5–7	35	37	10 rows × 4 bays	240–250

modern container handling system, so it offers more efficient and safer services than the current mid-stream operation. Major technological developments include a crane balance technology (Jung and Kwak 2009; Kwak and Oh 2009), spreader stabilization technology (Han and Lee 2009; Kim et al. 2009), and robot arm docking system (Lee et al. 2009; Shin et al. 2009). Technical feasibility of these new technologies has been successfully demonstrated in a recent open-sea test (MHBT 2011).

An illustrative operational scenario of a MH is as follows:

- A container ship calls at a port, and instead of berthing at a terminal, it anchors at an anchorage, remotely located from the terminal.
- For discharging operation, an empty MH unit travels to anchorage, and docks with the container ship (port or starboard side).
- Once the docking operation is completed, the MH unit starts to unload containers from the container ship on the deck (container storage area) of the unit.
- The MH unit may move to another ship-bay, re-dock with the container ship, and continue to unload containers at the new docking position.
- It continues to unload containers until the maximum container storage capacity is reached or there is no more container to be unloaded from the container ship.
- The MH unit travels back to the terminal, docks at a berth, and unloads containers using its own container crane.
- If there are more containers to be unloaded in the container ship, it travels back to the ship to unload another round of containers.
- For loading operation, containers are loaded on to a MH unit at a berth in a terminal.
- The loaded MH travels to the container ship, establishes a docking position, and loads containers on to the container ship.
- When all loading operations are carried out, the container ship departs the anchorage and the MH unit returns to the terminal.

According to the study by Kim and Morrison (2011)—also published in this special issue—MH-based container handling system can be an economically viable solution in spite of its apparent disadvantage of double-handling. A MH-based system consists of a fleet of MH units and, possibly, some number of small berths dedicated for MH operation. Kim and Morrison (2011) investigate the economic feasibility of a MH-based system across a variety of operational environments, and conclude that a 250 TEU MH system is favorable compared to other options in a few cases. Although the MH only proves competitive for limited applications, they do not include the indirect benefits by MH's flexibility. For example, a MH system can be used for direct ship-to-ship container transfer. In addition, MH can deliver containers to locations where container handling infrastructure does not exist. It can also offer a solution to respond to a temporary congestion at a busy port. Increasing productivity of MH operation is a key to improving its economic feasibility. As with conventional container terminals, planning and scheduling of the system's resources is a very important problem for enhancing the system productivity.

While various off-shore container operation concepts have been suggested and may seem promising, there has been little research on scheduling of off-shore operations. Therefore, our work on the MH system operations scheduling goes beyond its direct applications to the MH system, and provides a useful understanding for operations of off-shore container port service concepts.

1.2 Literature reviews

This paper focuses on a detailed work schedule for each MH unit for a MH fleet. In this MH scheduling problem, we determine a sequence of discharging and loading tasks, their start time, and a docking position of each MH unit for the tasks. The objective of this scheduling problem is to minimize total ship staying time for a series of incoming container ships. This problem shares common features with quay crane scheduling problem, which is an important research topic in the field of maritime operations research.

In a quay crane scheduling problem, it is common to define a task as a basic unit for jobs that a quay crane has to conduct. A task is a set of loading or discharging operations for a group of containers, where this group of containers is called a cluster. A task may be simply defined for the collocated containers—that is, containers that are stacked in the same bay always belong to one task (Peterkofsky and Daganzo 1990; Liu et al. 2006). Some researchers define a task in more detail by including such attributes as destination ports, stack locations, and others (Kim and Park 2004; Moccia et al. 2006; Sammarra et al. 2007). In this case, there can be more than one task in a single bay. In this study, we follow the latter type of task definition as it offers flexibility to capture unique characteristics of MH operations. With the definition, individual quay cranes are assigned to the tasks. Various types of constraints are considered depending on the specifics of an individual problem environment. Commonly found constraints from the literature include non-preemptable task, interference between quay cranes, and precedence relationships between tasks.

A quay crane scheduling problem is often formulated based on a more general type of scheduling problems. For example, it has been modeled as a parallel uniform machine scheduling problem with precedence constraints (Kim and Park 2004). Alternatively, it can be viewed as a variant of a vehicle routing problem. Moccia et al. (2006) apply solution techniques for the precedence constrained traveling salesman problem to crane scheduling. Additional constraints such as crane interference condition can be included to modify a basic vehicle routing problem

(Sammarra et al. 2007). Problem formulation and constraints expressed in these models provide useful insight for building a MH scheduling model.

One of the operational features of a MH system is that multiple number of MH units travel between the origin of a cluster and its destinations. This feature is found in a pickup-delivery vehicle routing problem with a single depot, finite number of vehicles of a limited capacity, and a number of customers. Each customer must be picked up at his origin and delivered to his destination with the goal of minimizing total travel cost. Solution methods to this problem are available in the literature. Casco et al. (1988) developed a load-based insertion procedure, where they insert pickup customers into the routes initially formed by delivery customers. This method is improved by Salhi and Nagy (1999) by allowing pickup customers to be inserted in clusters. Solutions to a vehicle routing problem can be modified to find a solution to a corresponding pickup/delivery vehicle routing problem (Nagy and Salhi 2005).

While many previous researches share some of the common features with a MH system scheduling problem, those models and solution methods are not directly transferrable to the MH scheduling problem, mainly because of the unique operational features of MH. For example, the MH problem has a concurrent operation constraint, i.e. some operations cannot be carried out simultaneously, and this constraint is coupled with a MH unit's docking position selection. More of these unique operational features are discussed in detail in Sect. 2.2.

This article is structured as follows. In Sect. 2, we define a MH operation scheduling problem in detail, and examine the characteristics of this problem. Section 3 presents a mathematical formulation as a mixed integer programming problem. As the mathematical model becomes intractable as the size of a problem gets bigger, a rule-based algorithm and a meta-heuristic approach based on genetic algorithm are developed in Sect. 4. Computational results from the proposed methods are compared and discussed in Sect. 5. Summary and conclusions follow in Sect. 6.

2 Problem descriptions

2.1 MH operation scheduling problem

A MH system operates a multiple number of MH units as a MH fleet, to serve a series of incoming container ships. For such a system, a MH operation scheduling problem is to determine an optimal sequence of discharging and loading operations, their start time, and a docking position of each MH unit for the specific task. The objective of this scheduling problem is to minimize total ship staying time, makespan, for a series of incoming container ships.

In this study, we use the MH-250 model's design information to determine necessary input values for the problem. The MH-250 model is designed to cover the width dimension of a (up to) 5,000 TEU container ship, as shown in Table 1. For this problem, we assume that the container ship to be handled by a MH unit is a 4,000 TEU ship. In a typical 4,000 TEU ship, the number of bays (forty-foot) is 14. Given the relative dimensions and configurations of a container ship and a MH unit, maximum of four MH units can concurrently operate on a container ship with two MH units docked at starboard and the other two at port side. The target maneuvering speed of a MH-250 unit is 8 knot, and we assume that an anchorage point for container ships is located 12 nautical miles away from a container terminal. This gives 90 min for a MH unit to make a one-way trip between a container ship and a berth at a terminal. Docking operation involves a fine maneuver for a MH unit at the proximity of a container ship and steps of mechanical operations (Lee et al. 2009; Shin et al. 2009), and it takes 30 min to complete a docking operation. The target value for on-board crane's container handling speed is approximately 60 TEU/h.

As a pre-process to solving a MH operational scheduling problem, we first define a *cluster*. A cluster is a group of containers, and it is represented as a set of container slot positions (bay-row-tier) on a container ship. A cluster is a basis to define a unit task to be executed by a MH unit. There are two types of a cluster: a discharging cluster and a loading cluster. A discharging cluster is a group of inbound containers to be unloaded from a container ship, and a loading cluster is a group of outbound containers to be loaded on a ship. Given a discharging and loading stowage plan, containers are grouped into a set of clusters as follows. For discharging containers:

- A cluster consists of containers from a single ship-bay. That is, containers from two or more ship-bays are not grouped as a single cluster, even when they are adjacent bays.
- The number of containers of a cluster cannot exceed 250 TEU (a MH-250 unit's maximum capacity).
- If there are more than 250 TEU containers in a single ship-bay, they are evenly divided into two or more clusters (e.g. 300 TEU containers are stacked in a single bay, two clusters are formed, each with 150 TEU containers). In this case, we assume that these clusters are located above and below each other, creating a precedence relationship between the two clusters.

Clusters for loading containers are grouped in the same way.

Actual operation on a cluster, discharging or loading, is referred to as a *task*. Due to the double-handling feature of MH, each cluster requires two tasks executed during its operational cycle. A discharging cluster is unloaded from a container ship to a MH unit, and later unloaded from the MH unit at a container terminal (MH berth). Thus, there are two tasks for a single discharging cluster: a *discharging task on sea* and *discharging task at berth*. Likewise, a loading cluster requires a *loading task at berth*, where a loading cluster is first loaded on to the deck of a MH, and a *loading task on sea* to load the cluster on to a container ship. Figure 3 illustrates the four types of tasks for a MH unit operation.

With a set of tasks as inputs, a MH operation scheduling problem is to assign each task to a set of available MH units, determine its docking position/side and the start time for each task.

There are a few assumptions on the MH operation scheduling problem:

- Arrival time of each container ship is a constant and known beforehand.
- Processing time of a task (loading/discharging time by onboard crane of a MH unit) is proportional to the number of containers in the cluster.

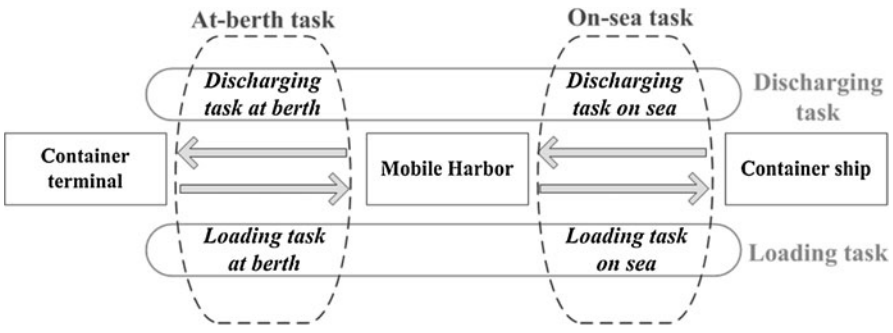


Fig. 3 The four types of tasks in the context of MH unit operation

- Docking time is a constant regardless of docking position.
- Travel time between a container terminal and a container ship (moored at a remote location) is a constant.
- Dual cycle operation is allowed; that is, a MH unit carries a loading cluster, loads them on to a container ship, and then works on a discharging cluster to carry them to a terminal.
- MH system's container terminal has unlimited resources (berths, yard truck, etc.) so that it does not cause any delay.

2.2 Unique operational features of MH operation

As with quay crane operation, a few fundamental constraints exist in the MH operation schedule problem. First and obvious is that there are precedence relationships among on-sea tasks. When more than one discharging clusters exist in a single ship-bay, the task for a cluster located above must be performed before the cluster below. Likewise, for single ship-bay loading tasks, the task for clusters located below must be conducted first. In addition, if one ship-bay hosts both discharging and loading clusters, the discharging task must precede the loading task. While MH operation is in general similar to quay crane operation, there are a few unique features specific to the MH operation scheduling problem. Unlike a quay crane working at a berth, a MH unit has a limit on the number of containers to handle in a single task due to its container carrying capacity limit. The number of containers to be stacked in the deck of a MH unit cannot exceed the maximum carrying capacity for both discharging and loading operation. Another feature that needs to be reflected in the modeling is that a cluster always generates a pair of tasks, and they must be operated by an identical MH unit. These pairs of tasks have a precedence relationship: a loading task at berth must precede its pair task—loading task on sea, and vice versa for discharging tasks.

Physical configurations and dimensions of a MH unit also need to be taken into account in the modeling. MH's on-board crane can move only within a range of the length dimension of MH unit. For MH-250 model, this length dimension is approximately 70 m, and the crane can travel for about 50 m in the aft to bow

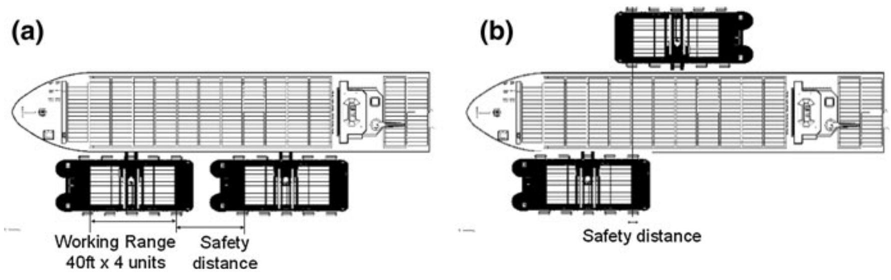


Fig. 4 The minimum safety distance for MH units (Sung et al., accepted). **a** Docking on the same side of a container ship. **b** Docking on the opposite side of a container ship

direction. This is equivalent to four 40-ft ship-bays length. The implication of this range restriction is that, once a MH unit docks at a certain position alongside a container ship, the crane's operating range is limited to a four 40-ft bay range with respect to the docking position.[1] Thus, to conduct a task for a cluster in a bay outside of the range, a MH unit has to disengage from its current docking position, maneuver to a new docking position, and conduct another docking at the new position. Additional implication of this restriction is that, in executing an on-sea task for a cluster, there are total of eight possible docking positions, four on each side of container ship.

It is important to properly determine a docking position for a MH unit because it can affect the interference condition. Just as a quay crane requires a minimum safety distance with its neighboring quay crane working on a same container ship, MH units require a safety distance. It is assumed here that the minimum safety distance between the crane working ranges of two MH units is three 40-ft ship-bays. This applies when two MH units dock on the same side of a container ship as shown in Fig. 4a. If two MH units dock on the opposite side (Fig. 4b), the two units can dock much closer to each other—up to the point their working ranges do not overlap. Thus, two tasks located nearby—within the three 40-ft bays—may be operated simultaneously by two MH units if the units can dock at an opposite side to each other. Whether this opposite-side-docking and simultaneous execution of the two tasks indeed improves the overall productivity is determined as part of the optimization problem.

Table 2 shows an example of a MH operation schedule for illustration purpose. It shows the sequence of tasks to be performed by each MH units, the time schedule for performing each task, and the docking position for the on-sea tasks.

3 Mathematical formulation

In this section, we present a mixed integer programming model for the MH operation scheduling problem where there are total of n clusters (n_l loading clusters and n_d discharging clusters).

[1] A docking position is defined as a ship-bay number to which the first MH deck-bay will be aligned when a MH unit docks with a container ship.

Table 2 An example of a MH operation schedule

MH index	Operation sequence	Task index	Information of cluster			Time schedule		Docking position	
			Location	Type	Number of containers	Start	Finish	Docking bay	Docking side
MH1	1	1	2, C1[a]	D[b]	60	02:00	03:00	3	S[d]
	2	3	3, C1	D	160	03:00	05:40	3	S
	3	8	–	(D)[c]	(160)[c]	07:40	10:20	–	–
	4	6	–	(D)	(60)	10:20	11:20	–	–
	5	10	–	(L)	(250)	11:20	15:30	–	–
	6	5	3, C1	L[c]	250	17:30	21:40	4	P[d]
MH2	1	9	–	(L)	(120)	00:00	02:00	–	–
	2	2	12, C1	D	100	04:00	05:40	12	P
	3	4	12, C1	L	120	05:40	07:40	12	P
	4	7	–	(D)	(100)	09:40	11:20	–	–

[a] (2, C1) = (ship-bay number, container ship ID)

[b] D: discharging task, L: loading task

[c] Parenthesis indicates the task is an at-berth task

[d] P: port of container ship, S: starboard of container ship

This model builds upon the constraints commonly used in a quay crane scheduling problem (Kim and Park 2004; Moccia et al. 2006; Sammarra et al. 2007) and a pickup-delivery vehicle routing problem (Desrochers et al. 1988). Some of those constraints are modified and additional constraints are added to express the unique features of MH operation schedule.

The notation to be used in a mixed integer programming is followed below: *Indices:*

i, j	Tasks Index, where $i, j \in \{1, \ldots, 2n\} \cup \{0, T\}$. Task 0 and T are dummy tasks to indicate an initial task and a final task, respectively
k	MH unit Index, where $k = 1, \ldots, K$ (K is the total number of MH units in a fleet)
v	Possible docking positions for a (on-sea) task, $v = 1, \ldots, 8$
c	Container ships Index, where $c = 1, \ldots, C$ (C is the number of container ships to be served)

Set of indices:

M	The set of discharging tasks ($M = M^+ \cup M^-$). It also consists of two subsets: a subset of discharging tasks on sea, $M^+ = \{j \mid j \in \{1, \ldots, n_d\}\}$, a subset of discharging tasks at berth, $M^- = \{j \mid j \in \{n+1, \ldots, n+n_d\}\}$
N	The set of loading tasks ($N = N^+ \cup N^-$). It consists of two subsets: a subset of loading tasks at berth, $N^+ = \{i \mid i \in \{n_d+1, \ldots, n_d+n_l\}\}$, and a subset of loading tasks on sea, $N^- = \{i \mid i \in \{n+n_d+1, \ldots, n+n_d+n_l\}\}$
A	The set of all tasks $(= N \cup M = \{1, \ldots 2n\})$

Φ	The set of task pairs that cannot be operated concurrently due to closeness of their positions. It has a pair of on-sea tasks and their docking positions as its member: (i, v_1, j, v_2)
Ω	The set of pairs of tasks between which there is a precedence relationship. When task i must precede task j: (i, j)
Ψ_c	The set of on-sea tasks which belong to container ship c

Problem data:

q_i	Changes in the number of containers on a MH unit after a task i is completed. If a task i is a loading-at-berth or discharging-on-sea task $(i \in N^+ \cup M^+)$, q_i is a positive value. If a task j is a loading-on-sea or a discharging-at-berth task $(j \in N^- \cup M^-)$, q_i is a negative value
Q	Maximum container carrying capacity of a MH unit (=250 TEU)
p_i	The processing (container loading/discharging) time for task i
t_{ij}	The travel time from the location of task i to task j. If task i is an on-sea task and task j is a at-berth task, or vice versa, t_{ij} is 120 min (90 min for one-way trip plus 30 min for docking). Otherwise, $t_{ij} = 0$
$ts_{iv_1jv_2}$	Re-docking time between task i and task j. If a docking position v_1 for task i is different from a docking position v_2 for task j, $ts_{iv_1jv_2}$ is 30 min. Otherwise, $ts_{iv_1jv_2} = 0$
r_i	The earliest possible start-time of task i. When task i is on-sea task, it is determined by arrival time of container ship. When task i is at-berth task, $r_i = 0$
U	A sufficiently large constant

Decision variables:

X_{ij}^k	1, if MH k performs task j immediately after completing task i; 0, otherwise
D_i	The completion time of task i
Y_c	The makespan of container ship c
Z_{ij}	1, if task j starts later than the completion time of task i; 0, otherwise
S_{iv}	1, if task i is performed at the docking position v; 0, otherwise
$Mjob_i$	The number of containers currently stacked in a MH unit immediately before task i is performed by a MH unit

The MH operation scheduling problem can be formulated as follows:

$$\text{Minimize} \sum_{c=1}^{C} Y_c \tag{1}$$

subject to

$$Y_c \geq D_i \quad \forall i \in \Psi_c,\ \forall c = 1, \ldots, C \tag{2}$$

$$\sum_{j \in A \cup \{T\}} X_{0j}^k = 1 \quad \forall k = 1, \ldots, K \tag{3}$$

$$\sum_{i \in A \cup \{0\}} X_{iT}^{k} = 1 \quad \forall k = 1, \ldots, K \tag{4}$$

$$\sum_{k=1}^{K} \sum_{i \in A \cup \{0\}} X_{ij}^{k} = 1 \quad \forall j \in A \tag{5}$$

$$\sum_{j \in A \cup \{T\}} X_{ij}^{k} - \sum_{j \in A \cup \{0\}} X_{ji}^{k} = 0 \quad \forall i \in A, \ \forall k = 1, \ldots, K \tag{6}$$

$$D_i - p_i \geq r_i \quad \forall i \in A \tag{7}$$

$$D_i \leq D_j - p_j \quad \forall (i,j) \in \Omega \tag{8}$$

$$D_i + p_j - D_j \leq U * \left(1 - Z_{ij}\right) \quad \forall i,j \in A \tag{9}$$

$$D_j - p_j - D_i \leq U * Z_{ij} \quad \forall i,j \in A \tag{10}$$

$$\sum_{v=1}^{8} S_{iv} = 1 \quad \forall i \in (N^{-} \cup M^{+}) \tag{11}$$

$$Z_{ij} + Z_{ji} - 1 \geq U * \left(S_{iv_1} + S_{jv_2} - 2\right) \quad \forall (i, v_1, j, v_2) \in \Phi \tag{12}$$

$$\sum_{j \in A \cup \{0\}} X_{ji}^{k} - \sum_{j \in A \cup \{T\}} X_{i+n,j}^{k} = 0 \quad \forall i \in (N^{+} \cup M^{+}), \quad \forall k = 1, \ldots, K \tag{13}$$

$$D_i + t_{i,i+n} \leq D_{i+n} - p_{i+n} \quad \forall i \in (N^{+} \cup M^{+}) \tag{14}$$

$$\begin{gathered} D_i + t_{ij} + p_j - D_j \leq U * \left(1 - X_{ij}^{k}\right) \\ \forall i \in (A \cup \{0\}), \ \forall j \in (A \cup \{T\}), \ \forall k = 1, \ldots, K \end{gathered} \tag{15}$$

$$\begin{gathered} D_i + ts_{iv_1jv_2} + p_j - D_j \leq U * \left(3 - \sum_{k \in K} X_{ij}^{k} - S_{iv_1} - S_{jv_2}\right) \\ \forall i,j \in (N^{-} \cup M^{+}), \ \forall v_1, v_2 = 1, \ldots, 8 \end{gathered} \tag{16}$$

$$\begin{gathered} Mjob_i + q_i - Mjob_j \leq U * \left(1 - X_{ij}^{k}\right) \\ \forall i \in (A \cup \{0\}), \ \forall j \in (A \cup \{T\}), \ \forall k = 1, \ldots, K \end{gathered} \tag{17}$$

$$0 \leq Mjob_i \leq Q \quad \forall i \in A \tag{18}$$

$$X_{ij}^{k} = X_{j-n,i+n}^{k} = X_{j-n,i}^{k} = X_{i,j-n}^{k} = 0 \quad \forall i \in M^{+}, \ \forall j \in N^{-}, \ \forall k = 1, \ldots, K \tag{19}$$

$$X_{ij}^{k} = 0 \text{ or } 1 \quad \forall i \in (A \cup \{0\}), \ \forall j \in (A \cup \{T\}), \ \forall k = 1, \ldots, K \tag{20}$$

$$Z_{ij} = 0 \text{ or } 1 \quad \forall i,j \in A \tag{21}$$

$$S_{iv} = 0 \text{ or } 1 \quad \forall i,j \in (N^{-} \cup M^{+}), \ \forall v = 1, \ldots, 8 \tag{22}$$

$$Y_c, D_i \geq 0 \quad \forall c = 1, \ldots, C, \ \forall i \in A \tag{23}$$

The objective function (1) minimizes the sum of the makespans for each container ship. Constraint (2) defines the final completion time to be the completion time of the last on-sea task. Constraints (3) and (4) set the first and last task of each

MH. Task 0 and T are dummy tasks which represent the initial state and final state, respectively. Constraint (5) guarantees that each task is performed by exactly one MH unit. Constraint (6) is a flow balance constraint that ensures the continuity of a task sequence assigned to a MH unit. Constraint (7) ensures that each task cannot be started before its earliest available time. Constraint (8) states that task j should not start before the completion of task i if $(i,j) \in \Omega$. Constraints (9) and (10) define Z_{ij} such that $Z_{ij} = 1$ when the operation for task j starts after the operation for task i is completed; $Z_{ij} = 0$, otherwise. If $Z_{ij} + Z_{ji} = 0$, it means that task i and task j can be performed simultaneously. If $Z_{ij} + Z_{ji} = 1$, it means that either task i or task j gets started first and the other waits until the preceding task is completed. $Z_{ij} + Z_{ji}$ cannot be 2 by definition of Z_{ij}. Constraint (11) ensures that all on-sea tasks are assigned one of the docking positions. Constraint (12) prevents interference between MH units. If task i, performed at docking position v_1, and task j, at docking position v_2, cannot be operated simultaneously, it makes $Z_{ij} + Z_{ji} \geq 1$. Constraint (13) ensures that two tasks representing the pickup and delivery request of the same cluster must be assigned to the same MH unit. Constraint (14) represents the precedence relationship between pickup and delivery points. Constraints (15) and (16) determine the completion time for each task and eliminate sub-tours. Constraint (15) determines the task completion time including the docking time and the travel time between a container terminal and container ship. Constraint (16) supplements constraint (15), and calculates the completion time when re-docking occurs. Constraints (17) and (18) guarantee that the number of containers stacked in a MH unit does not exceed MH's maximum container carrying capacity. Constraint (19) requires that, if a MH deck is holding containers that need to be unloaded at the current position (e.g. loading-on-sea task), these containers must be unloaded from the MH first before other containers can be loaded on to the MH. This is to ensure there is no need for a remarshaling operation on a MH unit.

It is well known that both a parallel uniform machine problem with precedence constraints and a vehicle routing problem with pickups and deliveries are NP-hard problem. It is also known that a quay crane scheduling problem is an NP-hard problem. Since this MH operation scheduling problem possesses their characteristics as well as its own additional complicating features, this problem is also deemed as an NP-hard problem, albeit we do not provide a formal proof in this work.

4 Proposed algorithm

Since solving the mixed integer programming model requires a considerable amount of computation time, it is practically infeasible to solve a problem with a reasonably large number of clusters. Thus, it is necessary to develop a heuristic algorithm to obtain a near-optimal solution within a reasonable amount of computation time. At first, a rule-based algorithm is developed. Then, a meta-heuristic method based on genetic algorithm is developed as an alternative.

4.1 A rule-based algorithm

A rule-based algorithm proposed in this section uses a relatively simple set of rules with a goal of obtaining a reasonable MH operation schedule in a short computation time. This algorithm consists of six steps, from forming a group of on-sea tasks to computing the objective value. Basic idea of each step is briefly discussed below, and the details of this algorithm procedure can be found in "Appendix".

Step 1. Form task groups out of all on-sea tasks The first step is to form task groups out of all on-sea tasks. A task group here refers to a set of tasks that will be carried out by a MH unit at a single docking position. On-sea tasks located within a neighboring four 40 ft-bay range can be handled by a MH unit at a single docking position. As such, it may be advantageous to group them as one task group, as it will possibly reduce the number of re-docking operation.

To form task groups, we first generate a randomly ordered sequence of all "on-sea tasks". Starting from the very first task, say task 1, in the sequence, we examine the second task, task 2, to see if the two tasks can be grouped together. Conditions (a)–(e) are checked in this step. If the conditions are satisfied, move on to task 3 to see if it satisfies the conditions. If task 3 does not satisfy any of the five conditions, the first task group is determined as task 1 and task 2. Then, starting from task 3, the above procedure is repeated. This process continues until all on-sea tasks belong to a task group.

(a) Only tasks within four 40 ft-bay apart can be included in a same group.
(b) Number of containers in a task group cannot exceed a MH unit's maximum capacity.
(c) Tasks included in a same task group do not violate any precedence relationship between them.
(d) No remarshaling operation is required (recall the explanation on constraint (19) in Sect. 3).
(e) A random number generated between (0,1) is greater than 0.3.

Step 2. Determine a docking position for each task group The second step is to determine a docking position for each task group. Recall that a docking position is defined as a ship-bay number at which the first MH deck-bay will be aligned to when a MH unit docks. For a task group whose target clusters are confined to one bay, say *k*th bay, there are four docking positions—*k*-3, *k*-2, *k*-1 and *k*—at which a MH unit can execute the task. Likewise, for a task group with its target clusters spread through two consecutive bays, three docking positions are feasible. If target clusters are distributed over three consecutive bays, two docking positions are possible, and for four consecutive bays, there is only one docking position for the task group. In determining a docking position for each task group, we would want to minimize the number of conflicts—a conflict being an overlap between two MH units' length range. To achieve this, a docking position for each task group is determined as follows:

(a) For each of the possible docking positions, count the number of clusters contained in the corresponding four-bay range, and choose a docking position with the least number of such clusters.

(b) If there is a tie for more than two docking positions, count the additional number of clusters in a six-bay range (extended by one bay in both direction from the four-bay range), and choose the one with a smaller count.

The counting scheme is based on a simple rationale. If there exists a cluster from other task group in the working range by a docking position, it necessarily means there is a conflict between the two task groups. Higher number of such clusters implies a longer duration for the conflict. Thus, we count the number of clusters from other tasks in the range, and select the position that has the minimum number of such clusters. This scheme does not give an optimal set of docking positions (that minimize the number of conflict). But, for the sake of simplicity for the rule-based approach, we use this method as an alternative.

Step 3. Assign task groups to MH units Now that task groups and their docking positions have been determined, each task group is assigned to a MH unit. As there are typically more task groups to be handled than the number of MH units in a fleet, one MH unit will be assigned multiple task groups. As with the first step, this step starts with randomly ordering task groups to generate an arbitrary sequence. Given a sequence, each group is assigned to a MH unit starting from the first group in the sequence. A basic idea behind determining which MH unit takes on a task group is that a task group is assigned to a MH unit that allows the earliest starting time for the first task in the task group. This starting time depends on quite a few factors, and Table 3 summarizes how the starting time is computed.

In addition to the earliest starting time, precedence conditions between the groups are checked to ensure that the task groups are ordered properly.

Step 4. Insert at-berth tasks into the groups Up to Step 3, at-berth tasks have not been explicitly considered in constructing an operations sequence for task groups. In this step, these at-berth tasks are inserted to an appropriate on-sea task group. Two conditions are considered in this process: (1) on-sea discharging tasks or at-berth loading tasks—in which MH unit is being loaded—are scheduled consecutively as long as a MH unit has sufficient space to store them all, (2) no remarshaling is allowed on a MH unit.

Step 5. Determine a completion time of each task In the fifth step, a true completion time for the above-obtained sequence of all tasks is computed. In this process, a docking side for a docking position needs to be determined. As mentioned in Sect. 2.2, whether two tasks can be executed concurrently by two MH units is affected by docking side selection, and this in turn affects the task completion time. In this sense, the completion time estimated by step 4 is a lower bound for a true completion time. Starting from the very first task in the sequence obtained from the fourth step, the earliest possible starting time is assigned to the task. For this first task, docking side is determined arbitrarily as there is no restriction. For the second task and on, the true earliest possible starting time may depend on whether concurrent operations by two MH units are allowed or not. Thus, a docking side is chosen such that a task can begin at the earliest possible time.

Step 6. Calculate the objective value and update the best schedule With a complete sequence of tasks and MH assignment determined, the objective value,

Table 3 Table for determining the earliest starting time of task group

MH status	1st task of next group					
	Discharging (ΣD_{nxt})			Loading (ΣL_{nxt})		
Non-empty (Discharging, ΣD_i)	$\Sigma D_i + \Sigma D_{\mathrm{nxt}} \geq 250$	$\Sigma D_i + \Sigma D_{\mathrm{nxt}} < 250$		$t^* + 2 \cdot 120$ + Emptying time + Charging time		
	$t^* + 2 \cdot 120$ + Emptying time	Same DP	Different DP			
		t^*	$t^* + 30$			
Empty (Loading, ΣL_i)	Same DP	Different DP		$\Sigma L_i + \Sigma L_{\mathrm{nxt}} \geq 250$	$\Sigma L_i + \Sigma L_{\mathrm{nxt}} < 250$	
	t^*	$t^* + 30$		$t^* + 2 \cdot 120$ + Charging time	Same DP	Different DP
					t^* + Charging time	t^* + Charging time + 30

Emptying time: time to make MH deck empty, charging time: time to load containers to MH deck

t^*: completion time of MH unit in current situation

ΣD_i: number of discharging containers in current MH deck, ΣL_i: number of loading containers before emptying MH deck

ΣD_{nxt}, ΣL_{nxt}: total number of loading or discharging containers used for processing next group

DP: Docking position

sum of makespans of all container ships, is computed for the schedule. Since this rule-based algorithm has some random factors, we repeat steps 1–6 to generate 10,000 schedules for a given set of tasks. Through the replication, the best schedule is updated as a better schedule is generated.

4.2 A random key based genetic algorithm

As an alternative to the rule-based algorithm, a meta-heuristic method using random key based genetic algorithm (rkGA) is developed. rkGA, first introduced by Bean (1994), uses indirect representation of permutations. A random key representation encodes a solution with random numbers, and these numbers are later used as a sorting key to decode a solution. The advantage of this algorithm is that no infeasible offspring is produced even with the traditional crossover operators. This advantage makes rkGA an attractive meta-heuristic method for various scheduling problems. (Goncalves et al. 2005; Bean 1994; Norman and Bean 1999; Okada et al. 2010; Xu and Bean 2007).

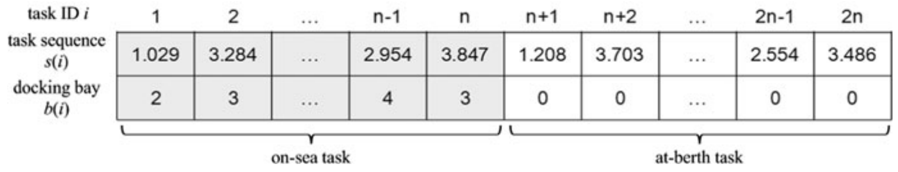
The overall procedure of rkGA for the MH operation scheduling problem is represented in Fig. 5. $P(t_{gen})$ and $C(t_{gen})$ are parents and offspring respectively, at the current generation, t_{gen}. The procedure starts with constructing initial population by using a random key based encoding routine. A decoding routine, then, evaluates each member in the initial population. In the subsequent steps, elitist reproduction, crossover, post tournament selection and immigration are sequentially carried out until a termination condition is met.

4.2.1 Random key based encoding process

The genetic representation (s, b) of a solution is composed of two parts of chromosomes, as shown in Fig. 6. The first part of chromosome, s, shows the sequence of each task along with an index number of MH units to take the task. The second part of chromosome, b, represents a docking position for each task. Recall

```
Procedure: random key based genetic algorithm (rkGA)
Input: Information of each containership, the number of MH units, GA parameters
Output: The best MH operation schedule
Begin:
  tgen ← 0;
  initialize P(tgen) by random key based encoding routine;
  evaluate P(tgen) by random key based decoding routine;
  while(not terminating condition) do
    reproduce P(tgen+1) from P(tgen) by elitist reproduction routine;
    for (remainder of P(tgen+1)) do
      create C(tgen) from P(tgen) by crossover routine;
      evaluate C(tgen) by random key based decoding routine;
      select P(tgen+1) from C(tgen) by post-tournament selection routine;
    end for
    replace P(tgen+1) by immigration routine;
    tgen ← tgen + 1;
  end while
end
```

Fig. 5 The overall procedure of random key based genetic algorithm



task ID i	1	2	...	n-1	n	n+1	n+2	...	2n-1	2n
task sequence $s(i)$	1.029	3.284	...	2.954	3.847	1.208	3.703	...	2.554	3.486
docking bay $b(i)$	2	3	...	4	3	0	0	...	0	0

on-sea task
at-berth task

Fig. 6 The genetic representation of an individual solution

that one cluster requires two tasks, one on sea and the other at berth. Thus, when there are n clusters, both parts of chromosomes have $2n$ genes, where each gene represents a task. The first n genes represent the information for the on-sea tasks, and the second n genes represent the information for at-berth tasks. Task j and task $j + n$ are the two tasks associated with one cluster—one at berth and the other on sea.

Each gene in a task sequence chromosome, $s(i)$, is given a real number. Integer part of this number is interpreted as the index number for a MH unit the task is assigned to. The integer part is randomly generated from $\{1,\ldots,K\}$, where K is the total number of MH units. It should be noted that $s(j)$ and $s(j + n)$ must have the same integer value, which indicates that a pair of tasks associated with an identical cluster should be assigned to the same MH unit. Decimal part of $s(i)$ is used as a sorting key when the sequencing and time schedule of each task is determined. It is generated randomly from a uniform distribution, *uniform*(0, 1).

Recall that in performing an on-sea task, a MH unit can dock at one of the four candidate docking positions (this is not considering its docking side). A docking position chromosome, $b(i)$, contains docking position information. Entries to the chromosome, $b(i)$ is generated randomly from $\{1, 2, 3, 4\}$ to indicate the four candidate docking position, 1 being the leftmost position. Since at-berth tasks do not need a docking position, this part of the chromosome, $b(n + 1)$ to $b(2n)$, is set to 0.

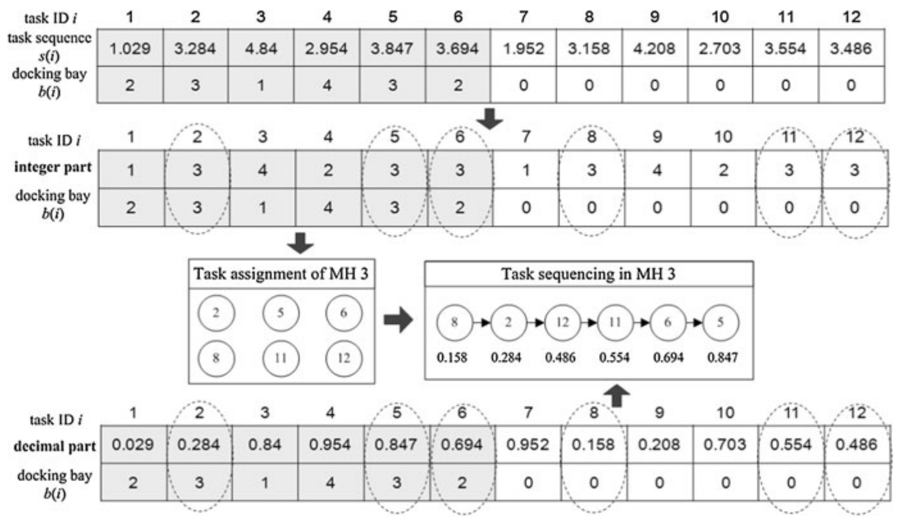
Later in the experiments, we use the rule-based algorithm to generate a subset of an initial population. The use of the rule-based algorithm for generating initial population aims to improving the quality of an initial population.

4.2.2 Random key based decoding process

To decode a solution from a chromosome, we first extract the integer part from a task sequence chromosome, $s(i)$, to group tasks by each MH unit. Integer part of $s(i)$ represents the index number of MH that will perform task i. A group of tasks that have the same integer value are then assigned to the MH whose index number is the integer value from $s(i)$.

Next, for the tasks in a group, decimal part of $s(i)$ determines a sequence of the tasks. A lower decimal value means the higher priority, and thus earlier position in the sequence. A task may lose its priority to a task with the next-highest priority if:

(a) A precedence relationship requires other task be executed ahead of it.
(b) Adding the task results in the number of containers on the MH unit greater than its maximum carrying capacity.
(c) Adding the task causes remarshaling operation on the MH unit.



task ID *i*	1	2	3	4	5	6	7	8	9	10	11	12
task sequence *s*(*i*)	1.029	3.284	4.84	2.954	3.847	3.694	1.952	3.158	4.208	2.703	3.554	3.486
docking bay *b*(*i*)	2	3	1	4	3	2	0	0	0	0	0	0

task ID *i*	1	2	3	4	5	6	7	8	9	10	11	12
integer part	1	3	4	2	3	3	1	3	4	2	3	3
docking bay *b*(*i*)	2	3	1	4	3	2	0	0	0	0	0	0

task ID *i*	1	2	3	4	5	6	7	8	9	10	11	12
decimal part	0.029	0.284	0.84	0.954	0.847	0.694	0.952	0.158	0.208	0.703	0.554	0.486
docking bay *b*(*i*)	2	3	1	4	3	2	0	0	0	0	0	0

Fig. 7 An example of random key based decoding routine

If a task loses its priority, then a task with the next highest priority takes the position in the sequence. Figure 7 illustrates a simple example of this procedure.

Note that a task sequence determined in the above step is within the group of tasks by each MH unit, and we need to determine the complete sequence across all MH task groups. As with the sequencing within a MH task group, a decimal part of *s*(*i*) is used to provide a default sequence for all tasks. For this cross-group task sequencing, there is only one condition by which a task loses its default priority: a precedence relationship for the current task requires other task (from other MH task group) be executed ahead of it. Now that a complete sequence for entire set of tasks is determined, the final task completion time can be computed. Here, a key consideration is a possibility for concurrent execution of on-sea tasks by multiple MH units. Whether concurrent execution is allowed or not depends on the choice of a docking side for the tasks in question. Determining a docking side and computing task completion times can be done in the same way as the fifth step of the rule-based algorithm (see Sect. 4.1).

Once all tasks are scheduled, sum of the completion times of the last on-sea task for each container ship. The result is then used as a fitness value for this solution.

4.2.3 Generation of next population

Four strategies are applied to construct a new population for a next generation: elitist reproduction, uniform crossover, post tournament selection and immigration.

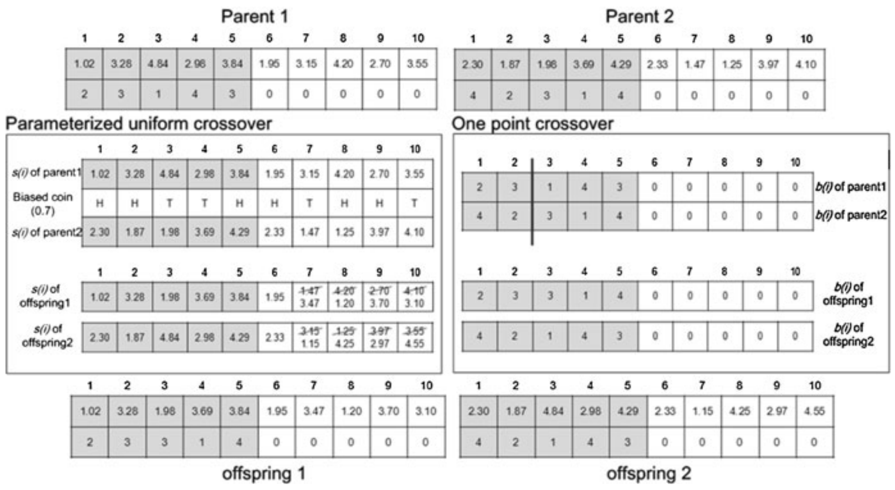


Fig. 8 Crossover operator (parameterized uniform crossover + one-point crossover)

First, the top 15% of the current population is passed down to the next generation. This procedure is called *elitist reproduction*, which is to copy the best individual chromosome from one generation to the next (Goldberg 1989). The advantage of the elitist strategy is that the best solution is monotonically improving from one generation to the next.

Next step is to apply a crossover operator. To do this, two chromosomes are randomly selected from the current population as parents. Then, for a task sequence chromosome, $s(i)$, a parameterized uniform crossover (Spears and DeJong 1991) is applied. For each gene of $s(i)$, we toss a biased coin to select which parent will contribute the allele (mid-left of Fig. 8). Amount of this bias indicates the degree of change for the chromosomes. Preliminary experiments show that bias probability of 0.7 works well for our scheduling problem. Note that, in this crossover operation, we still keep the integer value of $s(i + n)$ to be the same integer value of $s(i)$. For a docking position chromosome, $b(i)$, one point crossover is employed, as shown on the mid-right of Fig. 8.

Once two offspring chromosomes are generated as a result of the crossover operation, a post-tournament selection (Norman and Bean 1999) is applied. The two offspring chromosomes are evaluated, and the one with a better objective value enters the next generation. The other offspring is discarded. This process is repeated until the rest of the population (85%) is filled up.

The final step is to replace some percentage of the chromosomes with a low fitness value with randomly generated chromosomes. This is the concept of immigration (Bean 1994). An immigration operation is better than a traditional mutation operation in terms of preventing premature population convergence due to the elitist reproduction. From preliminary experiments, it is found that replacing the bottom 3% yields good results.

5 Computational results

To evaluate the performance of the proposed methods for the MH operation scheduling problem, we conduct a number of computational experiments using various problem instances. For small sized problems where the exact solution is available from solving MIP problem, we compare results from the rule-based algorithm and rkGA to the exact solution. For larger-sized problems, we develop a heuristic to estimate a lower bound for the problem, and use the results to evaluate the performance of the rule-based algorithm and the rkGA-based method.

The mixed integer programming is coded in CPLEX 11.0. The rule-based algorithm and rkGA are coded in C#. We run on a desktop computer with Core 2 Quad 2.40 GHz CPU and 3.00 GB RAM. The rkGA uses four parameters: the ratio of elitist reproduction, the ratio of immigration, the population size, and the number of generation. The ratio of elitist reproduction and immigration are set to 15 and 3% respectively. Population size is 100, and the number of generations is 1,000. For initial population of the rkGA method, about 10% of the initial population is generated from the solutions by the rule-based algorithm. The other 90% of the initial population are randomly generated.

5.1 Computational results for small sized problem instances

For a relatively small-sized problem instances, we conduct comparison between the solutions from the mixed integer programming, rule-based algorithm, and the rkGA. As test cases, we consider two factors to generate problem instances: the number of clusters and distribution of clusters across ship-bays. The number of clusters determines the size of the scheduling problem, and we use four levels for the cluster size: 4, 5, 6, and 7 clusters. The spread of clusters across ship-bays is also likely to affect the difficulty of obtaining a scheduling solution. For this, we use two levels: clusters are concentrated at certain ship-bays (C), and uniformly distributed across ship-bays (U). In addition, to account for other factors such as variance of the number of containers across clusters, four different stowage plans–a, b, c, d–are used for each {number of clusters–spread type} combination. Finally, the number of MH units is varied from 1 to 4 units. So there are total of 128 problem instances.

As shown in Table 4, the mixed integer programming is solved to provide the optimal solution for 107 tested problem instances, and for the remaining 21 instances, a lower and upper bound at the program termination is reported. With the rkGA, it turns out that the method finds the optimal solution for the 107 instances. For the 21 cases, where a lower and upper bound are given, the objective value found by rkGA lies inside the lower and upper bound. Thus, it seems that rkGA performs very well in terms of its solution quality for small sized problem instances. The rule based algorithm also finds the optimal solution for majority of the problem instances, but it does not find the optimal solution in 19 cases. For the 21 lower- and upper-bound ranges, two cases lie outside the bounds.

As expected, the computation time of the mixed integer programming tends to increase as the size of a problem. The computation time of rkGA also increases, but not as significantly as the MIP case. It does not change much for the rule-based algorithm.

Table 4 Comparison between mixed integer programming, rule-based algorithm, and the rkGA for a relatively small-sized problem instances

Problem		MIP		RBA			rkGA		
Attri.	MH	Obj.	Sec.	Obj.	Gap[a](%)	Sec.	Obj.	Gap(%)	Sec.
5Ca	1	1,412	23.4	1,412	0	0.9	1,412	0	29.1
	2	688	14.6	688	0	1.0	688	0	31.0
	3	604	61.1	604	0	1.1	604	0	32.4
	4	501	10.9	501	0	1.3	501	0	33.2
5Cb	1	1,811	27.1	1,811	0	1.2	1,811	0	29.1
	2	824	25.9	883	7.2	1.3	824	0	30.8
	3	597	23.0	597	0	1.3	597	0	30.6
	4	519	13.2	519	0	1.6	519	0	31.7
5Cc	1	1,855	10.6	1,855	0	1.0	1,855	0	27.8
	2	917	11.2	917	0	1.2	917	0	29.6
	3	722	10.3	722	0	1.3	722	0	30.4
	4	637	18.9	637	0	1.4	637	0	30.8
5Cd	1	1,822	10.6	1,822	0	1.0	1,822	0	28.8
	2	1,036	23.5	1,036	0	1.1	1,036	0	29.3
	3	666	1.5	697	4.7	1.2	666	0	29.4
	4	666	1.7	666	0	1.3	666	0	31.4
5Ua	1	1,532	170.2	1,532	0	1.3	1,532	0	25.3
	2	740	33.5	740	0	1.4	740	0	26.0
	3	552	10.2	552	0	1.7	552	0	26.8
	4	552	7.8	552	0	1.7	552	0	25.8
5Ub	1	1,866	32.9	1,866	0	1.0	1,866	0	25.4
	2	1,036	24.0	1,036	0	1.1	1,036	0	26.9
	3	584	18.8	584	0	1.3	584	0	26.8
	4	536	1.4	536	0	1.7	536	0	28.5
5Uc	1	1,378	126.4	1,378	0	1.4	1,378	0	24.1
	2	597	19.4	597	0	1.5	597	0	24.9
	3	484	8.7	491	1.4	1.7	484	0	25.0
	4	456	1.3	456	0	2.0	456	0	25.5
5Ud	1	1,808	41.9	1,808	0	1.2	1,808	0	29.2
	2	796	17.9	796	0	1.3	796	0	30.3
	3	612	8.5	612	0	1.2	612	0	31.7
	4	598	1.9	598	0	1.6	598	0	29.3
6Ca	1	1,922	733.7	1,922	0	1.5	1,922	0	44.9
	2	934	729.2	934	0	1.8	934	0	45.4
	3	741	152.5	832	12.3	1.8	741	0	46.7
	4	603	13.6	690	14.4	2.6	603	0	46.7

Table 4 continued

Problem		MIP		RBA			rkGA		
Attri.	MH	Obj.	Sec.	Obj.	Gap[a](%)	Sec.	Obj.	Gap(%)	Sec.
6Cb	1	2,213	274.0	2,213	0	1.5	2,213	0	46.7
	2	1,183	56.5	1,183	0	1.7	1,183	0	48.1
	3	817	156.8	817	0	1.7	817	0	50.9
	4	698	17.0	777	11.3	1.7	698	0	52.6
6Cc	1	2,007	205.1	2,007	0	1.4	2,007	0	47.4
	2	1,069	207.5	1,069	0	1.8	1,069	0	47.1
	3	722	102.8	722	0	1.9	722	0	48.5
	4	637	74.2	646	1.4	2.2	637	0	49.9
6Cd	1	1,856	542.9	1,856	0	1.4	1,856	0	44.6
	2	1,036	271.1	1,036	0	1.9	1,036	0	46.5
	3	674	50.0	700	3.9	2.0	674	0	47.1
	4	666	12.6	674	1.2	2.1	666	0	50.5
6Ua	1	1,926	1,881.7	1,926	0	1.5	1,926	0	43.5
	2	799	205.3	799	0	1.7	799	0	44.2
	3	699	1,029.8	699	0	2.1	699	0	43.4
	4	552	16.8	581	5.3	2.5	552	0	44.0
6Ub	1	2,162	1,385.9	2,162	0	1.5	2,162	0	40.7
	2	1,076	237.0	1,084	0.7	1.9	1,076	0	46.3
	3	673	87.3	710	5.5	1.9	673	0	42.7
	4	536	20.1	543	1.3	2.3	536	0	46.8
6Uc	1	1,647	2,630.3	1,647	0	1.5	1,647	0	45.1
	2	683	378.7	683	0	2.0	683	0	46.3
	3	560	1,630.4	597	6.6	1.9	560	0	48.8
	4	456	46.9	461	1.1	2.1	456	0	50.3
6Ud	1	2,412	387.0	2,412	0	1.8	2,412	0	45.4
	2	1,166	2,363.1	1,166	0	2.1	1,166	0	47.1
	3	760	439.1	760	0	1.7	760	0	46.2
	4	612	40.3	612	0	1.8	612	0	49.4
7Ca	1	(1,377, 2,414)[b]	Memory	2,414	–	2.2	2,414	–	66.4
	2	(724, 1,322)[b]	Memory	1,263	–	2.3	1,263	–	70.2
	3	902	3,267.5	902	0	2.5	902	0	69.3
	4	(603, 832)[b]	Memory	832	–	2.7	832	–	75.5
7Cb	1	(1,148, 2,245)[b]	Memory	2,245	–	1.7	2,245	–	58.8
	2	(698, 1,215)[b]	Memory	1,183	–	1.8	1,183	–	63.4
	3	(698, 849)[b]	Memory	817	–	1.9	817	–	63.7
	4	698	18.48	730	4.6	2.0	698	0	66.3
7Cc	1	2,691	2,159.8	2,691	0	1.7	2,691	0	58.3
	2	1,279	7,726.4	1,279	0	1.9	1,279	0	61.9
	3	891	4,824.3	891	0	1.8	891	0	60.1
	4	722	2,630.9	722	0	1.9	722	0	61.4

Table 4 continued

Problem		MIP		RBA			rkGA		
Attri.	MH	Obj.	Sec.	Obj.	Gap[a](%)	Sec.	Obj.	Gap(%)	Sec.
7Cd	1	(1,062, 2,269)[b]	Memory	2,269	–	1.5	2,269	–	54.8
	2	(666, 1,688)[b]	Memory	1,202	–	1.8	1,202	–	59.2
	3	(676, 954)[b]	Memory	868	–	1.8	868	–	59.3
	4	700	337.5	706	0.9	2.0	700	0	61.3
7Ua	1	(1,040, 2,397)[b]	Memory	2,368	–	1.5	2,363	–	65.8
	2	(552, 1,241)[b]	Memory	1,162	–	1.7	1,162	–	65.8
	3	(552, 800)[b]	Memory	737	–	2.3	730	–	68.6
	4	(552, 615)[b]	Memory	615	–	2.3	615	–	64.3
7Ub	1	(1,039, 2,397)[b]	Memory	2,342	–	1.8	2,342	–	57.8
	2	(552, 1,241)[b]	Memory	1,169	–	2.1	1,169	–	59.3
	3	(536, 710)[b]	Memory	837	–	2.2	710	–	59.4
	4	543	11.3	680	25.2	2.3	543	0	61.8
7Uc	1	(978, 2,385)[b]	Memory	2,332	–	1.6	2,332	–	57.6
	2	(587, 1,222)[b]	Memory	1,185	–	1.8	1,185	–	60.7
	3	661	482.2	661	0	1.9	661	0	52.4
	4	579	4,847.6	611	5.5	2.3	579	0	61.0
7Ud	1	(1,073, 2,451)[b]	Memory	2,451	–	2.5	2,451	–	69.5
	2	(598, –)[b]	Memory	1,166	–	1.7	1,166	–	59.3
	3	(598, 760)[b]	Memory	799	–	1.8	760	–	59.1
	4	598	40.8	598	0	1.9	598	0	61.4

4-cluster cases are not shown in the table: both rkGA and rule based algorithm find the optimal solution for those cases

[a] Gap = (final solution of proposed algorithm − final solution of MIP) · 100/final solution of MIP

[b] (x, y): the value of x and y is a lower and upper bound for the problem instance

5.2 Computational results for large sized problem instances

For the problem instances with more than eight clusters, it turns out that a mixed integer programming problem cannot be solved with the solver and a meaningful lower and upper bound is not obtainable.

In order to evaluate the quality of the solutions of the proposed algorithms for these problems, a heuristic method is developed to estimate a lower bound as a reference for comparison. As developing a tight lower bound for general cases turns out to be difficult, we design 1-container ship problem instances where we can derive a reasonable lower bound by introducing a set of relaxing assumptions:

- There is no interference between any set of on-sea tasks, and thus multiple units of MH can carry out on-sea tasks concurrently.
- Time for re-docking operation is ignored.

With the above assumptions, we decompose a lower bound into two components: container handling time and traveling time. Suppose there are L [TEU] containers to be loaded, and D [TEU] containers to be discharged. The number of MH units is K. For the container handling time component, the best way to utilize K MH units is to evenly distribute the workload across them. So, each MH unit has to load $\lceil L/K \rceil = w_L$ containers and discharge $\lceil L/K \rceil = w_D$ containers, respectively. $\lceil r \rceil$ denotes a function to return the minimum integer that is greater or equal to r. Since a container, either loading or discharging, involves two handling steps—one at berth and the other on sea, total container handling time for this MH unit is $2 \cdot w_L \cdot t_C + 2 \cdot w_D \cdot t_C$, where t_C is a container handling time per TEU. Note that a container ship can depart once the last on-sea discharging task is complete. Thus, we need to deduct one at-berth discharging task time from the above container handling time. For this, we assume the last discharging task has 250 TEU containers (maximum carrying capacity, Q), and so this amount of container handling time is subtracted to give the following lower bound for the container handling time component:

$$2 \cdot w_L \cdot t_C + 2 \cdot w_D \cdot t_C - Q \cdot t_C \qquad (24)$$

For the traveling time components, we estimate a lower bound by counting the minimum number, H, of one-way trips required to complete all tasks. When L loading containers and D discharging containers are ideally clustered, we have $\lceil L/Q \rceil = N_L$ loading clusters and $\lceil D/Q \rceil = N_D$ unloading clusters. Each cluster requires one round trip. Note that by the dual cycle operation assumption, a pair of loading-discharging cluster can be handled in one round trip. There are Min(N_L, N_D) loading-discharging cluster pairs, and the rest are either loading-only or discharging-only clusters. Total number of round trips to handle all clusters is Max(N_L, N_D), say R. Number of one-way trips is $2R$. Recall that a container ship can depart as soon as the last on-sea task is complete. Thus, some of the $2R$ one-way trips do not contribute to the container ship staying time, and are irrelevant. We determine the number of such one-way trips by assigning K MH units to R round trips. Let us take an example of $R = 5$ and $K = 2$. The two MH units, $\mathrm{MH_A}$ and $\mathrm{MH_B}$, will make their first round trip. In the next round, $\mathrm{MH_A}$ and $\mathrm{MH_B}$ make another round trip. By then, four out of five ($R = 5$) round trips have been completed. For the last round trip, only $\mathrm{MH_A}$ will need to make a trip. In the second round trip, $\mathrm{MH_B}$'s return trip is irrelevant. Likewise, the return trip for $\mathrm{MH_A}$ is irrelevant as well. Therefore, the minimum number of one-way trips, H, is $10 - 2 = 8$. This procedure can be expressed by the following equation:

$$H = \begin{cases} 2 \cdot K \cdot (\lfloor R/K \rfloor - 1) + K + 2 \cdot \mathrm{mod}(R, K) = 2R - K, & K \le R, \\ R, & K > R \end{cases} \qquad (25)$$

Thus, total minimum time for traveling component is $H \cdot t_{tr}$, where t_{tr} is a one-way travel time for a MH unit. Then, a lower bound for the traveling time component is obtained by simply dividing this total time by the number of MH units: $(H \cdot t_{tr})/K$.

The overall lower bound value is the sum of the two components, container handling time component and traveling time component, and it is

Table 5 Comparison between lower bound, rule-based algorithm, and rkGA for a relatively larger sized problem instances

Problem		LB	RBA			rkGA		
Attri.	MH	Obj.	Obj.	Gap[a](%)	Sec.	Obj.	Gap[a](%)	Sec.
12_1	1	5,514	5,893	6.9	6.5	5,786	4.9	302.4
	2	2,572	2,928	13.8	5.1	2,747	6.8	293.1
	3	1,592	1,941	21.9	6.2	1,727	8.5	242.9
12_2	1	4,724	5,097	7.9	4.2	5,062	7.2	277.5
	2	2,178	2,461	13.0	5.3	2,436	11.8	327.6
	3	1,328	1,706	28.5	6.3	1,546	16.4	284.0
12_3	1	4,992	5,432	8.8	7.2	5,420	8.6	318.4
	2	2,312	2,711	17.3	8.6	2,564	10.9	333.4
	3	1,418	1,825	28.7	8.3	1,788	26.1	306.5
12_4	1	5,150	5,550	7.8	4.2	5,520	7.2	187.3
	2	2,390	2,640	10.5	6.6	2,610	9.2	231.1
	3	1,470	1,830	24.5	8.7	1,650	12.2	278.9
12_5	1	6,830	6,870	0.6	6.8	6,840	0.1	289.1
	2	3,230	3,270	1.2	6.1	3,270	1.2	314.1
	3	2,030	2,100	3.4	7.7	2,070	2.0	319.6
12_6	1	5,030	5,495	9.2	4.7	5,435	8.1	287.7
	2	2,330	2,760	18.5	5.7	2,585	10.9	321.3
	3	1,430	1,820	27.3	7.6	1,790	25.2	309.4
15_1	1	6,312	6,958	10.2	7.9	6,898	9.3	505.4
	2	2,972	3,436	15.6	7.5	3,340	12.4	455.0
	3	1,860	2,295	23.4	10.3	2,196	18.1	558.3
15_2	1	5,802	6,540	12.7	8.2	6,432	10.9	526.5
	2	2,716	3,304	21.6	11.2	3,198	17.7	604.5
	3	1,688	2,184	29.4	12.2	2,108	24.9	550.2
15_3	1	8,254	8,394	1.7	8.7	8,334	1.0	462.5
	2	3,944	4,274	8.4	11.0	4,164	5.6	514.9
	3	2,506	2,876	14.8	8.3	2,829	12.9	391.9
15_4	1	6,380	7,023	10.1	8.4	7,023	10.1	376.0
	2	3,006	3,525	17.3	7.6	3,525	17.3	358.8
	3	1,880	2,253	19.8	11.3	2,223	18.2	563.8
15_5	1	6,110	6,852	12.1	6.0	6,732	10.2	450.3
	2	2,970	3,450	16.2	10.8	3,390	14.1	553.4
	3	1,790	2,310	29.1	8.6	2,280	27.4	401.5
15_6	1	8,300	8,475	2.1	6.5	8,355	0.7	350.9
	2	3,966	4,305	8.5	11.4	4,245	7.0	593.4
	3	2,520	2,910	15.5	11.6	2,880	14.3	602.5
18_1	1	8,392	8,911	6.2	9.8	8,761	4.4	769.3
	2	4,012	4,538	13.1	11.2	4,320	7.7	832.4
	3	2,552	3,124	22.4	16.6	2,905	13.8	725.0

Table 5 continued

Problem		LB	RBA			rkGA		
Attri.	MH	Obj.	Obj.	Gap[a](%)	Sec.	Obj.	Gap[a](%)	Sec.
18_2	1	8,060	8,986	11.5	14.8	8,676	7.6	945.3
	2	3,846	4,394	14.2	16.7	4,318	12.3	900.0
	3	2,442	3,070	25.7	20.4	3,064	25.5	998.9
18_3	1	8,142	9,127	12.1	11.1	8,887	9.2	890.3
	2	3,888	4,412	13.5	18.7	4,393	13.0	1,053.6
	3	2,468	2,969	20.3	18.5	2,969	20.3	992.3
18_4	1	7,790	8,550	9.8	13.5	8,400	7.8	917.0
	2	3,710	4,380	18.1	13.9	4,170	12.4	1,012.7
	3	2,350	2,850	21.3	15.7	2,820	20.0	904.8
18_5	1	10,430	10,800	3.5	14.5	10,530	1.0	852.3
	2	5,030	5,250	4.4	11.1	5,220	3.8	764.8
	3	3,230	3,720	15.2	10.9	3,510	8.7	700.4
18_6	1	8,750	9,460	8.1	14.5	9,250	5.7	856.7
	2	4,190	4,600	9.8	18.9	4,600	9.8	971.0
	3	2,670	3,100	16.1	14.4	3,090	15.7	876.4
Average				14.1	10.2		11.3	555.7

[a] Gap = (final solution of proposed algorithm − Lower bound) · 100/Lower bound

$$\{2 \cdot w_L \cdot t_C + 2 \cdot w_D \cdot t_C - Q \cdot t_C\} + \{(H \cdot t_{tr})/K\} \tag{26}$$

54 problem instances in total are tested. We use three levels for the cluster size: 12, 15 and 18 clusters. Also, six different stowage plans are used for each level of cluster size. These stowage plans are randomly generated. Finally, the number of MH units is varied from 1 to 3 units.

Table 5 shows that the average gap between the lower bound and the rkGA solution is 11.3%. Out of 54 tested instances, rkGA produces a solution within 10% of the lower bound value in 26 cases. For the rule-based algorithm, the average gap is 14.1%, and there are 18 within-10% cases. Thus, for this limited set of test cases, we see that the rkGA produces a reasonably good solution compared to the lower bound, and generates better quality solutions than the rule-based algorithm.

6 Conclusions

Mobile Harbor is a specially designed floating platform equipped with a modern container handling system that can load/discharge containers to/from an anchored container ship in the open sea. With its unique operational features, a MH system introduces novel scheduling issues. A MH unit has a limited capacity, and tasks may have conflicts with each other depending on its docking position.

A mixed integer programming model for this scheduling problem is presented. The mathematical formulation builds upon a quay crane scheduling problem and vehicle routing problem with pickups and deliveries. As this problem is only solvable for very small-sized problem instances and quickly becomes intractable, two alternative solution methods have been proposed: a rule-based algorithm and a random key based genetic algorithm (rkGA). A rule-based algorithm is based on the rational that reducing the number of re-docking operations and possibility for inter-MH interference would result in a reasonably good schedule. rkGA uses a random key concept to represent an order schedule, and it offers a convenience in handling an order-based scheduling problem.

For small-sized problem instances where the exact optimal solution or reasonable bound information is available from the mixed integer problem, rkGA is shown to be able to produce the optimal solution or a solution within the bounds. The rule-based algorithm finds optimal solutions and solutions within the bounds for majority of cases, but there are 21 cases out of 128 test problems that the method does not perform well.

For large-sized problem instances, we calculate an approximate lower bound for a comparison purpose. The method we develop to compute a lower bound cannot be applied for a general case, but in a single container ship case, it provides a reasonable lower bound. Experimental results show that the proposed rkGA works reasonably well, with about 11% gap from the lower bound. The rule-based algorithm's performance is worse than the rkGA, showing about 14% gap. Overall, quality of the solutions obtained from the rkGA based method is better than the rule-based algorithm.

Computation time for the rkGA based method is in the order of tens of seconds for small-sized problems and hundreds of seconds for large-sized problems (one container ship cases). On the other hand, the computation time for the rule-based algorithm is roughly an order of magnitude smaller than the rkGA method. Though not presented here, we have tested the rkGA method to solve for more than one container ship cases, and the computation time dramatically increases as high as over an hour ($\sim$3,900 s for two container ships each with 16 clusters). With the rule-based algorithm, it is in the order of a minute. Whether 10 min to 1 h of extra time to compute a near optimal schedule is acceptable or not depends on a practical operational environment of MH-based system. If a quick response is required, the rule-based algorithm will be appropriate, and if a better solution is desired and 10–15 min response time is tolerable, the rkGA method is recommended.

There are a few areas that future study may further investigate. First, computing more reliable lower bound from the exact algorithm such as branch and bound will be desirable. Also, more realistic and dynamic operational environments, such as dynamic arrival time of container ships and a hatch cover constraint, can be incorporated in the modeling and solution procedure.

Acknowledgments This study was supported by Mobile Harbor Research Grant by Korea Ministry of Knowledge Economy. This support is gratefully acknowleged.

Appendix: Detail explanation of rule-based algorithm

Form task groups out of all on-sea tasks

Notation

Seq_i: i-th on-sea task or i-th task group in random sequence

$MaxBay_g$, $MinBay_g$: maximum and minimum bay number within task group g, respectively

Bay_i: bay number of on-sea task i

LCon, DCon: the sum of number of loading containers and discharging containers in task group g or on-sea task i or MH unit k, respectively

```
Input data: all on-sea tasks $\in (N^{-} \cup M^{+})$
i=1; g=1;
randomly generate a sequence ($seq_i$)
while(i $\leq$ the number of on-sea tasks){
  if(i=1){insert $seq_i$ to $G_g$ ; i++;}
  else{
    if($MaxBay_g$-3$\leq Bay_i \leq MinBay_g$+3 and ($seq_i$, j) $\notin \Omega$, for all on-sea tasks j in $G_g$){
      check that the precedence is tangled with other groups when $seq_i$ is in $G_g$;
      if(constraint above is feasible){
        if($LCon_g$>0 and $DCon_g$=0){
          if($Seq_i \in$M+ or ($Seq_i \in$N- and $LCon_g$+$Con_i$ $\leq$250)){
            generate random number rd in (0, 1);
            if(rd$\leq$0.7) {insert $seq_i$ to $G_g$; i++;}
            else{ g++; insert $seq_i$ to $G_g$; i++;}
          }else{
            g++; insert $seq_i$ to $G_g$; i++;
          }
        }else{
          if($Seq_i \in$M+ and $DCon_g$+$Con_i \leq$250){
            generate random number rd in (0, 1);
            if(rd$\leq$0.7) {insert $seq_i$ to $G_g$; i++;}
            else {g++; insert $seq_i$ to $G_g$; i++;}
          }else{
            g++; insert $seq_i$ to $G_g$; i++;
          }
        }
      }
    }else{
      g++; insert $seq_i$ to $G_g$; i++;
    }
  }
}

Output data: task groups $G_{gr}$ (gr=1,...,g)
```

Determine a docking position for each task group

```
Input data: task groups $G_{gr}$ ($gr=1,...,g$)
$gr=1$;
calculate the number of task groups ($N_v$) in ship-bay v
while($gr \le g$){
  if($MaxBay_{gr}$-$MinBay_{gr}$=3){
    docking bay of $G_{gr}$ = $MinBay_{gr}$; gr++;
  }else if($MaxBay_{gr}$-$MinBay_{gr}$=2){
    candidate docking bays: $MinBay_{gr}$-1 (cdb1), $MinBay_{gr}$ (cdb2)
    compare $\sum_{v=MinBay_{gr}-1}^{MinBay_{gr}+2} N_v$ (of cdb1) with $\sum_{v=MinBay_{gr}}^{MinBay_{gr}+3} N_v$ (of cdb2);
    if(only one is the minimum value){
      docking bay of $G_{gr}$ = $MinBay_{gr}$-1 or $MinBay_{gr}$;
    }else{
      case1: compare $\sum_{v=MinBay_{gr}-1-i}^{MinBay_{gr}+2+i} N_v$ (of cdb1) and $\sum_{v=MinBay_{gr}-i}^{MinBay_{gr}+3+i} N_v$ (of cdb2) as i is
             increased by 1 until one of them is smaller than the other. Candidate docking bays
             having smaller value is determined as the docking bay of $G_{gr}$
      case2: if more than two candidate docking bays have same value yet case1 is proceeded,
             one of them is randomly selected as a docking bay of $G_{gr}$
    }
    gr++;
  }else{
    apply to the same procedures used in ($MaxBay_{gr} - MinBay_{gr}$ = 2) about more candidate
     docking bays and select the docking bay of $G_{gr}$;
    gr++;
  }
}

Output: docking bay of each task group $G_{gr}$
```

Assign task groups to MH units

```
Input data: task groups $G_{gr}$ and docking bay of each task group

give a priority for task groups $G_{gr}$;
make a sequence of task groups $G_{gr}$ using priorities and precedence relationships similar to
  Graham's list scheduling algorithm;

count=1;
while(count $\le$ the total number of task groups){
  select the first group( j) of sequence above;
  calculate the fastest start time of group j as max(condition1, condition2, condition3)
    condition1: the earliest start time among MH units to operate group j
                = Min(completion time of MH unit k + travel time (as shown in table3))
    condition2: max(completion time of a set of predecessor groups of group j)
    condition3: ready time of group j

  assign the group j to the MH unit of resulting condition1 and update the completion time of MH
    unit as (max(condition1, condition2, condition3) + processing time of the group);
  add the group j to order list O;
  delete the group j out of sequence;
  count++;
}

Output: sequence of groups in each MH unit, order list (O) of task groups
```

Insert at-berth tasks into the groups

```
Input data: sequence of groups in each MH unit

for(k is 1 to the number of MH units){
    i=1; pl=0; pd=0;(as shown in Fig. 9)
    $LCon_k$=0; $DCon_k$=0; tempL=empty; tempD=empty;
    while(i<=the number of groups in MH unit k){
        if(i=1){
            if($LCon_i$>0 and $DCon_i$>0){
                insert loading tasks at berth of $G_i$ to position pl; pl++;
                $DCon_k$=$DCon_i$; tempD={discharging tasks at berth of $G_i$}; pd++;
            }else if($LCon_i$>0 and $DCon_i$=0){
                $LCon_k$=$LCon_i$; tempL={loading tasks at berth of $G_i$}; pd++;
            }else{     //LConi=0 and DConi>0
                $DCon_k$=$DCon_i$; tempD={discharging tasks at berth of $G_i$}; pd++; pl++;
            }
            i++;
        }else{
            if($LCon_i$>0 and $DCon_i$>0){
                if($DCon_k$>0){
                    insert tempD to position pd; tempD=empty;
                    insert loading tasks at berth of $G_i$ to position pl; pl++;
                }else{
                    if($LCon_k$+$LCon_i$≤250){
                        insert loading tasks at berth of $G_i$ to tempL;
                        insert tempL to position pl; pl=i; $LCon_k$=0; tempL=empty;
                    }else{
                        insert tempL to position pl; pl=i-1; $LCon_k$=0; tempL=empty;
                        insert loading tasks at berth of $G_i$ to position pl; pl++;
                    }
                }
                $DCon_k$=$DCon_i$; tempD={discharging tasks at berth of $G_i$}; pd++; i++;
            }else if($LCon_i$>0 and $DCon_i$=0){
                insert tempD to position pd; $DCon_k$=0; tempD=empty; pd++;
                if($LCon_k$+$LCon_i$≤250){
                    insert loading tasks at berth of $G_i$ to tempL;
                }else{
                    insert tempL to position pl; pl=i-1; tempL=empty;
                    $LCon_k$=$LCon_i$; tempL={loading tasks at berth of $G_i$}
                }
                i++;
            }else{ //$LCon_i$=0 and $DCon_i$>0
                insert tempL to position pl; pl=i; tempL=empty; $LCon_k$=0;
                if($DCon_k$+$DCon_i$<=250){
                    insert discharging tasks at berth of $G_i$ to tempD; pd++;
                }else{
                    insert tempD to position pd; tempD=empty;
                    $DCon_k$=$DCon_i$; tempD={discharging tasks at berth of $G_i$}; pd++;
                }
                i++;
            }
        }
    }
    insert tempD to position pd; insert tempL to position pl;
}

Output data: sequence of all tasks in each MH unit
```

See Fig. 9.

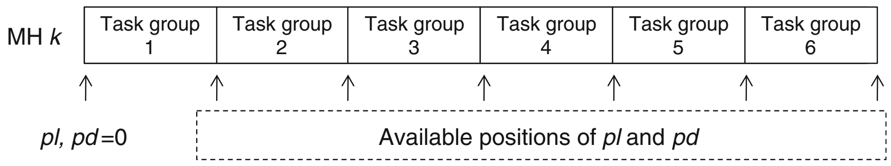


Fig. 9 Explanation of variable pl and pd for STEP 4 in rule-based algorithm

Determine a completion time of each task

```
Input data: sequence of all tasks in each MH unit, order list (O) of groups from output of step 3

setfinish=empty;
while(the completion time of all tasks is not determined){
  for(k is 1 to the number of MH units){
    while(the first task is at-berth task in sequence of tasks of MH unit k){
      calculate the completion time of at-berth task
                    = the completion time of MH unit k + travel time + processing time;
      add the at-berth task to setfinish;
      delete the at-berth task in the sequence of tasks of MH unit k;
    }
  }
  choose the on-sea task i including the group having the highest priority among the set of first
task of the sequence of tasks of each MH unit;
  if(on-sea task i is the first task of the group){
    calculate temporary start time of on-sea task i = max(condition1, condition2, condition3);
      condition1: the earliest start time of MH = the completion time of MH + travel time
      condition2: max(completion time of a set of predecessor tasks of task i)
      condition3: ready time

    arrange on-sea tasks of setfinish to an increasing order of their completion time;
    if(docking side of on-sea task i is port){
      while(arranged on-sea task j of setfinish is not checked){
        if(on-sea task i and j are simultaneously performed at certain time slot when on-sea task
i is started at temporary start time){
          if(docking position of on-sea task i and j cannot satisfy the safety distance){
            temporary start time of on-sea task i = completion time of on-sea task j;
          }
        }
      }
    }else{ // docking side of on-sea task i is starboard;
      same procedure is applied;
    }
    compare the temporary start times to be performed at each side of container ship (port,
starboard);
    determine earliest start time and docking side of task i among the temporary start times to be
performed at each side of container ship (port, starboard);
    calculate the completion time of task i = start time + processing time;
  }else{
    determine docking side of the on-sea task i as the docking side of first on-sea task included in
same group;
    calculate temporary start time of on-sea task i = max(condition1, condition2, condition3);
      condition1: the earliest start time of MH = the completion time of MH + travel time
      condition2: max(completion time of a set of predecessor tasks of task i)
      condition3: ready time
```

```
        arrange on-sea tasks in setfinish to an increasing order of their completion time;
        while(arranged on-sea task j in setfinish is not checked){
            if(on-sea task i and j are simultaneously performed at certain time slot when on-sea task i is started at temporary start time){
                if(docking position of on-sea task i and j cannot satisfy the safety distance){
                    temporary start time of on-sea task i = completion time of on-sea task j;
                }
            }
        }
        Determine the start time of on-sea task i and calculate the completion time;
    }
    add on-sea task i to setfinish;
}

Output data: the completion time of all tasks in each MH unit
```

References

Bean JC (1994) Genetic algorithms and random keys for sequencing and optimization. Oper Res Soc Am J Comput 6:154–160. doi:10.1287/ijoc.6.2.154

Casco DO, Golden BL, Wasil EA (1988) Vehicle routing with backhauls: models, algorithms, and case studies. In: Golen BL, Assad AA (eds) Vehicle routing: methods and studies. Elsevier, Amsterdam, pp 127–147

Desrochers M, Lenstra JK, Savelsbergh MWP, Soumis F (1988) Vehicle routing with time windows: optimization and approximation. In: Golen BL, Assad AA (eds) Vehicle routing: methods and studies. Elsevier, Amsterdam, pp 65–84

Goldberg DE (1989) Genetic algorithm in search optimization and machine learning. Addison-Wesley, Boston

Goncalves JF, Mendes JJM, Resende MGC (2005) A hybrid genetic algorithm for the job shop scheduling problem. Eur J Oper Res 167:77–95. doi:10.1016/j.ejor.2004.03.012

Han SH, Lee JH (2009) Container docking system, container crane, and container docking method, container crane and container docking method. South Korean patent application, filing number: 1020090082529, filing date: 2 Sept 2009

Hong Kong Mid-Stream Operators Association (HKMOA) (2009) http://www.hkmoa.com/Statistics.aspx?lang=E

Jung H, Kwak BM (2009) Balance keeping crane and vessel with the crane. South Korean patent application, filing number: 10-2009-0074380, filing date: 12 Aug 2009

Kim JH, Morrison JR (2011) Offshore port service concept: classification and economic feasibility. Flex Serv Manuf J 156:752–768. doi:10.1007/S10696-011-9100-9

Kim KH, Park Y (2004) A crane scheduling method for port container terminals. Eur J Oper Res 156:752–768. doi:10.1016/S0377-2217(03)00133-4

Kim SH, Kim UH, Hong YS, Ju HJ, Kim J, Kwak YG, Kwak BM (2009) Auto landing, location, locking device for spreader of crane and method thereof. South Korean patent application, filing number: 1020090074305, filing date: 12 Aug 2009

Kwak BM, Oh JH (2009) Balance keeping crane and vessel with the crane. South Korean patent application, filing number: 1020090070799, filing date: 31 July 2009

Taiwah Sea & Land Heavy Transport Ltd (TS & LHT) (2011) Tai Wah Jumbo. http://www.taiwahhk.com/Equippage/equippage.htm

Lee PS, Jung H, Lee DY, Kim SI (2009) Docking system for a ship and docking method using the same. South Korean patent application, filing number: 1020090074208, filing date: 12 Aug 2009

Liu J, Wan YW, Wang L (2006) Quay crane scheduling at container terminals to minimize the maximum relative tardiness of vessel departures. Nav Res Logist 53:60–74. doi:10.1002/nav.20108

Meisel F (2009) Seaside operations planning in container terminals. Physica-Verlag, Berlin. doi:10.1007/978-3-7908-2191-8

Mobile Harbor Business Team (MHBT) (2011) An introduction material about the mobile harbor project. http://www.mobileharbor.or.kr/index.html → English → Public Relation → Brochure. Accessed 13 Jan 2011

Moccia L, Cordeau J-F, Gaudioso M, Laporte G (2006) A branch-and-cut algorithm for the quay crane scheduling problem in a container terminal. Nav Res Logist 53:45–59. doi:10.1002/nav.20121

Nagy G, Salhi S (2005) Heuristic algorithms for single and multiple depot vehicle routing problems with pickups and deliveries. Eur J Oper Res 162:126–141. doi:10.1016/j.ejor.2002.11.003

Norman BA, Bean JC (1999) A genetic algorithm methodology for complex scheduling problems. Nav Res Logist 46:199–211. doi:10.1002/(SICI)1520-6750(199903)46:2<199:AID-NAV5>3.0.CO;2-L

Okada I, Zhang XF, Yang HY, Fujimura S (2010) A random key-based genetic algorithm approach for resource-constrained project scheduling problem with multiple modes. In: Proceedings of the international multiconference of engineers and computer scientists, vol 1

Peterkofsky RI, Daganzo DF (1990) A branch and bound solution method for the crane scheduling problem. Transp Res B 24:159–172. doi:10.1016/0191-2615(90)90014-P

Salhi S, Nagy G (1999) A cluster insertion heuristic for single and multiple depot vehicle routing problems with backhauling. J Oper Res Soc 50:1034–1042. doi:10.1057/palgrave.jors.2600808

Sammarra M, Cordeau J-F, Laporte G, Monaco MF (2007) A tabu search heuristic for the quay crane scheduling problem. J Sched 10:327–336. doi:10.1007/s10951-007-0029-5

Shin HK, Shin JW, Kim MS, Jung WJ (2009) Docking system of ship. South Korean patent application, filing number: 1020090093030, filing date: 30 Sept 2009

Spears WM, Dejong KA (1991) On the virtues of parameterized uniform crossover. In: Proceedings of the fourth international conference on genetic algorithms, San Diego, pp 230–236

Suh NP (2008) Mobile harbor to improve ocean transportation system and transportation method using the same. Korean patent no. 100895604, granted date: 23 Apr 2008

Sung IK, Nam HC, Lee TS (accepted) Scheduling algorithm for mobile harbor: an extended m-parallel machine problem. Int J Ind Eng Theory

Xu S, Bean JC (2007) A genetic algorithm for scheduling parallel non-identical batch processing machines. In: Proceeding of the IEEE symposium on computational intelligence in scheduling (SCIS 07), Honolulu, HI, pp 143–150

Author Biographies

Hochang Nam received B.S. and M.S. degrees in Industrial and Systems Engineering from KAIST (Korea Advanced Institute of Science and Technology), South Korea. He is currently a Ph.D. student in the Department of Industrial and Systems Engineering, KAIST, South Korea. His research interest is system design operations management using system simulation and optimization.

Taesik Lee is an Assistant Professor in the Department of Industrial and Systems Engineering at KAIST. He obtained his M.S. and Ph.D. in Mechanical Engineering at MIT. Before joining KAIST, he worked as an Assistant Director at Park Center for Complex Systems at MIT, where he carried out system design and analysis research in aerospace applications. His current research interest is in system design operations management using system simulation and optimization.