Noname manuscript No.
(will be inserted by the editor)

A Quay Crane System That Self-Recovers From Random Shocks

Yun Fong Lim • Yan Zhang • Chen Wang

Abstract The main challenge for a container terminal is to maximize its throughput using limited resources subject to various operational constraints under uncertainty. Traditional methods try to achieve this through an optimized plan by solving a quay crane scheduling problem; but the plan may become obsolete or infeasible after shocks (changes in the system due to uncertainty). To respond to shocks these methods require frequent re-planning, which increases the operations cost. We propose a new method to counter this. Instead of creating plans, we develop an operating protocol to respond to shocks without re-planning. Under this protocol, each quay crane along a berth follows simple rules to serve vessels that arrive continuously in time. If the system is configured properly, it always spontaneously recovers to its efficient form after a random shock. The average throughput of the system operating on its efficient form is very near its full capacity if the crane travel time per bay is relatively short. This self-recovery is robust even under a sequence of shocks as the system persistently restores its throughput after each shock. Most importantly, this is accomplished without complex computation.

Keywords: quay cranes; container terminals; shocks; self-organizing systems

1 Introduction

As globalization shapes the world rapidly, deep-sea maritime transportation becomes increasingly important as a key component in the global supply chain (UNCTAD 2009). The top three busiest container ports in the world: Shanghai, Singapore, and Hong Kong handle 31,740, 29,940, and 24,380 thousand TEUs (Twenty-foot Equivalent Units), respectively, in 2011. These ports compete not only with each other, but also with new ports from emerging economies with significantly lower operations costs. As a result, container port operators are keen to improve their productivity by adopting new operations methods to increase their competitiveness.

Typically, each container port consists of several terminals. Each terminal comprises several berths and each berth is equipped with quay cranes to handle containers. For example, the port of Singapore has 4 terminals, 54 berths (with a total quay length of 52,480 feet), and 190 quay cranes. In contrast, the port of Rotterdam, which is the busiest port in Europe, has 9 terminals, 23 berths (with a total quay length of 291,920 feet), and 103 quay cranes.

Within a container terminal there are three types of container movements: import, export, and transshipment. Figure 1 shows that after a vessel arrives at a container terminal, quay cranes transfer import containers to internal trucks (operated by the terminal), which carry them to their

Y.F. Lim, Y. Zhang, C. Wang
Tel.: +65-6828-0774
Fax: +65-6828-0777
E-mail: yflim@smu.edu.sg, yan.zhang.2008@mom.smu.edu.sg, cwang@smu.edu.sg
Lee Kong Chian School of Business, Singapore Management University, Singapore 178899

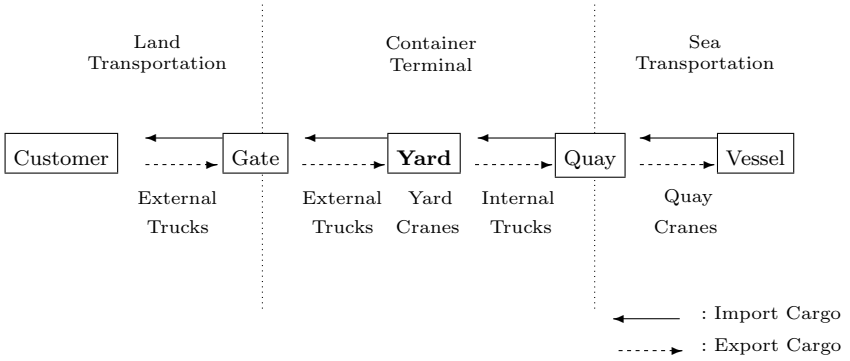


Fig. 1 Typical import and export operations in a container terminal.

assigned storage locations at the yard. These containers are then unloaded to the yard by yard cranes. The import containers stay in the yard until they are transported to customers by external trucks (operated by the customers).

On the other hand, export containers arrive at the terminal by external trucks. These containers are stored in the yard until the vessels that transport them to their next destination ports arrive. The export containers are then loaded by yard cranes to internal trucks that carry them to a berth, where they are loaded to the corresponding vessels by quay cranes.

Transshipment containers arrive at the terminal by sea. They are stored in the yard temporarily before they are shipped to their next destination ports. For modeling purposes, each transshipment container is seen as a pair of import and export containers in this paper.

Quay cranes are standard equipment for handling containers at the berth. A quay crane usually handles one container at a time. A large crane can reach out to serve a vessel that is 22 container rows wide. It is crucial to operate the quay cranes efficiently because they are the most expensive equipment in a container terminal. The operations of quay cranes are subject to two constraints: (1) The *non-crossing constraint* requires quay cranes to remain in a fixed sequence along the berth. This is because cranes along the same berth share a common rail. (2) The *minimum separation constraint* requires quay cranes to keep a minimum distance from each other due to safety reasons.

Each vessel is assigned to a berthing position when it arrives at a container terminal. After they are moored at their assigned positions, they wait to be served by quay cranes. Given limited quay length and resources, the container terminal can only serve a finite number of vessels at a time. For congested ports, it is generally crucial to serve the vessels swiftly. Thus, it is important to coordinate quay cranes effectively to maximize the throughput (number of containers handled per unit time). In fact, the average throughput of a port is one of the most important performance measures for container ports.

Typically, a fixed number of quay cranes along a berth serve a series of vessels that arrive sequentially over time. The problem is to allocate import and export containers to the quay cranes such that the long-run average throughput of the berth is maximized, subject to the non-crossing and the minimum separation constraints. Despite the vessels' arrival times and information on import and export containers are generally given in advance, this problem is still difficult to solve due to uncertainty in terminal operations. Sources of uncertainty include mishandling of containers, breakdowns of equipment, and delay of vessels' arrivals.

Most papers in the literature address the above problem by solving a *quay crane scheduling problem*. The goal is to find a *plan* for the cranes to operate within a finite time horizon so that an objective function is optimized (see Bierwirth and Meisel (2010) for a survey). The plan is typically found through mathematical programming methods or heuristics.

A *shock* is an unexpected event that may cause a plan created by the above approach to become obsolete or infeasible. According to Meisel and Bierwirth (2013), two types of shocks are frequently observed. The first type of shocks lengthens a container's handling time, and the affected quay crane may block the movements of other cranes along the berth. A shock of this type could be caused by a disrupted stowage plan (a predetermined stowage plan differs from the actual stowage of containers) or a breakdown of an internal truck in the yard. The second type of shocks corresponds to late vessel arrivals. Both types of shocks will delay the service for a vessel. This may generate conflicts among services for vessels and cause the original plan infeasible. The problem with the above, traditional approach is that it requires re-planning if its original plan for crane operations

is no longer feasible after a shock. As shocks occur from time to time, the above approach could be onerous to implement in practice as it may require frequent re-planning.

To counter this, we propose a new and potentially applicable approach to address the above problem. Instead of repeatedly finding optimized plans, we develop an *operating protocol* to respond to shocks without re-planning. Under this protocol, each quay crane along the berth follows simple rules to serve vessels that arrive continuously in time. We assume the crane travel time from one point to another along the berth is proportional to the distance traveled. Under the operating protocol, the system can be configured such that it will spontaneously recover to its efficient form after a random shock. If the crane travel time per bay is sufficiently short, then the throughput of the system operating on its efficient form is very near its full capacity. Even under a sequence of shocks (a sequence of unexpected events such as misplacements of containers, breakdowns of internal trucks, etc.), the system is robust to recover from each shock as it persistently restores its throughput. This is especially appealing to terminal operators because the system can spontaneously restore near maximum efficiency after each shock without complex computation or onerous re-planning. Furthermore, the non-crossing and the minimum separation constraints can be easily satisfied under our approach.

To use our approach a terminal operator will have to control the processing rate of each crane (the number of containers that a crane can handle per unit time). In practice, the bottleneck of the unloading and loading processes lies on the internal trucks that transport containers between the berth and the yard. Due to traffic congestion in the terminal, quay cranes often wait for the internal trucks to arrive. By properly allocating the internal trucks to the quay cranes, one can control the processing rates of the cranes.

The contributions of this paper are summarized as follows:

1. We propose an operating protocol to coordinate quay cranes along a berth such that the system spontaneously responds to shocks and constantly maximizes its throughput. This protocol is easy to implement as it only requires the cranes to follow simple rules.
2. We analyze the system's dynamics for a special case where workload is uniformly distributed on the vessels. If the internal trucks are allocated to the cranes such that the cranes' processing rates are from slowest to fastest, then the system always recovers to its efficient form after a random shock. We also show that if the crane travel time per bay is sufficiently short, the throughput of the system operating on its efficient form is very near its full capacity. When the system is subject to a sequence of shocks, it repeatedly recovers from each shock and persistently restores its throughput. Even if workload is not uniformly distributed (which is common in practice), our numerical experiments suggest that the system constantly maintains its throughput near its full capacity.

We organize the paper as follows. After reviewing the related literature, we first analyze the system with uniform workload in Section 3. We assume the port is sufficiently congested so that new jobs are always available. We demonstrate in Section 4 that the system under the proposed protocol can self-recover from shocks and restore its efficiency. We then test the system's performance when workload is nonuniform in Section 5. We consider cases including variable turnover time of internal trucks and nonuniform number of bays per job. Section 6 gives concluding remarks.

2 Related literature

Bierwirth and Meisel (2010) give an excellent survey on seaside operations planning in container terminals. The authors divide the problem into three parts: the berth allocation problem, the quay crane assignment problem, and the quay crane scheduling problem. See references therein for the literature of each problem and of integrated models. For a comprehensive review of seaside operations planning in container terminals, please refer to Meisel (2009).

For the quay crane scheduling problem, many papers in the literature assume deterministic settings and consider various objective functions. Under this approach, a terminal operator may need to re-solve the problem when a shock occurs. Daganzo (1989) and Peterkofsky and Daganzo (1990) are the first to address the quay crane scheduling problem. The authors neglect the crane travel time and the non-crossing and minimum separation constraints. Lim et al. (2004) solve the quay

crane scheduling problem using dynamic programming. Their objective is to maximize throughput subject to the non-crossing, the minimum separation, and the job separation constraints. Kim and Park (2004) consider the non-crossing and the minimum separation constraints with significant crane travel time. Zhu and Lim (2006) minimize the latest completion time subject to the non-crossing constraint with negligible crane travel time. Other variants of the quay crane scheduling problem have been studied by Liu et al. (2006), Lim et al. (2007), Lee et al. (2008), Guan et al. (2010), Meisel (2011), and Legato et al. (2012). Li et al. (2009) study the crane scheduling problem in the yard with inter-crane interference and the minimum separation constraint. Meisel and Bierwirth (2011) present a unified approach to compare different models and solution procedures for quay crane scheduling.

The literature on quay crane scheduling under uncertainty is very limited. Legato et al. (2010) consider unloading and loading containers under uncertainty. They find a schedule by solving the deterministic quay crane scheduling problem and then evaluate the schedule under the impact of uncertainty by simulation. Zeng et al. (2011) study the problem of recovering quay crane schedules when disruptions occur. They find an updated schedule by solving a recovery model such that the deviation of the updated schedule from the original one is minimized.

A closely related problem is the berth allocation problem in which each vessel is assigned a berthing position and a berthing time such that an objective is optimized. See Bierwirth and Meisel (2010) and references therein for models and methods to solve the berth allocation problem. For work related to berth allocation under uncertainty, please see Guan and Yang (2010), Hendriks et al. (2010), Zhen et al. (2011), Golias (2011), and Xu et al. (2012).

Several authors have proposed approaches to solve the berth allocation problem and the quay crane scheduling problem in an integrated manner. Lee et al. (2006) integrate berth allocation and quay crane scheduling using a feedback loop. The solution of the berth allocation problem is input to the quay crane scheduling problem. The latter is then solved to obtain the total processing time of each vessel, which is used to update the berth allocation. The loop is repeated for a number of iterations. Similar approach is proposed by Meier and Schumann (2007). Liu et al. (2006) study an integrated model where quay crane schedules are preprocessed to generate the total processing time of each vessel, which is then used to solve the berth allocation problem. Meisel and Bierwirth (2013) proposes an integrated approach similar to that of Liu et al. (2006) and further adjusts the berth allocation and the crane schedule in a feedback loop. Han et al. (2010) develop a proactive approach to solve the simultaneous berth and quay crane scheduling problem with stochastic arrival and handling time.

A significant amount of work has been done on the integration of berth allocation and the problem of assigning cranes to vessels (without scheduling). For this stream of work, please see Park and Kim (2003), Oğuz et al. (2004), Meisel and Bierwirth (2006, 2009), Theofanis et al. (2007), Giallombardo et al. (2008), Imai et al. (2008), Hendriks et al. (2010), and Blażewicz et al. (2011). The resultant assignment of cranes to vessels is then used to find crane schedules.

The rules that we propose to coordinate quay cranes are closely related to the ideas of self-balancing assembly lines introduced by Bartholdi and Eisenstein (1996, 2005). The authors propose a simple protocol (called "bucket brigades") to coordinate workers along an assembly line such that, if the workers are sequenced appropriately, their workload will be balanced spontaneously. The same ideas can be applied to order-picking in warehouses to boost productivity (Bartholdi et al. 2001), to bus routes to resist bus bunching (Bartholdi and Eisenstein 2012), and to inventory systems to recover a target replenishment schedule after shocks (Eisenstein 2005). For related work in this area, see Bartholdi et al. (2006, 2009, 2010), Lim and Yang (2009), Lim (2011, 2012), and Lim and Wu (2014). However, the technical analysis and proofs of this paper are very different from that of the above papers.

A well-known approach to address optimization problems under uncertainty is robust optimization (see, for example, Ben-Tal and Nemirovski (1999, 2000), Bertsimas and Sim (2003, 2004)). This approach has been implemented in a dynamic setting that involves decision making in stages (Bertsimas and Thiele 2006, Adida and Perakis 2006). Ben-Tal et al. (2004) propose the concept of adjustable robust counterpart to better adapt to a multi-stage decision process. This concept allows decisions to be delayed until information becomes available. Unfortunately, models based on adjustable robust counterpart are generally NP-hard. Although linear (Ben-Tal et al. 2005, Bertsimas et al. 2010) and piecewise linear (See and Sim 2010) decision rules have been proposed

to reduce the complexity, the resultant models generally remain large and finding their solutions often involves heavy computation. In contrast, we propose an operating protocol that enables the system to respond to shocks after they occur. This approach does not require complex computation and is relatively straightforward to implement.

3 The uniform workload case

We first consider an ideal case where workload is uniformly distributed on vessels. Although in practice workload is usually not uniformly distributed, this special case provides useful insights on the system's dynamics. It also allows us to find a lower bound on the average throughput, which can be used to evaluate the system's performance under various parameter settings. Table 1 summarizes the notation used in this paper.

Table 1 Notation used in this paper.

b	Number of bays per job
n	Number of quay cranes in the system
B	Length of the berth (in number of bays)
z_i	Location of crane i along the berth (in number of bays)
d	Minimum distance between two neighboring cranes
μ_i	Processing rate of crane i (in number of containers per unit time)
α	Average time for a crane to travel a bay along the berth
$\bar{\tau}$	Time when all cranes start moving backward simultaneously in a reset
$\sigma(i)$	Rotation function of crane index
x_i	Fraction of workload completed on the job of crane i
$\mathbf{x}$	State of the system, $\mathbf{x} = (x_1, \ldots, x_n)$
m	Number of containers to be handled (unloaded and loaded) per bay
ρ	Average throughput of the system
h	Maximum number of tiers in each stack of containers
θ	Maximum variation of turnover time of internal trucks
δb	Maximum variation of number of bays per job

3.1 Definitions

Consider a berth that serves a sequence of vessels. Containers on each vessel are partitioned into *bays* along the vessel's longitudinal axis. Each bay stretches across the width of the vessel. Figure 2 shows a vessel where containers are stored in 12 bays along its longitudinal axis, in 8 *rows* across its width, and up to 4 *tiers*. After all import containers are unloaded from a bay, export containers are then loaded to the same bay. Both import and export containers of each bay are determined in advance before the vessel arrives at the berth.

Define a *job* as the unloading and loading operations for b adjacent bays on the vessels, where b is a parameter. As shown in Figure 3, a job may comprise bays from different vessels that are adjacent to each other along the berth. We first analyze an ideal case where the workload of each job is a constant, and is uniformly distributed on each bay of the job so that every bay has the same workload. This assumption is aligned with a practical objective that strives to evenly distribute cargo on each vessel to ensure balance of the vessel's center of gravity. We assume each job is preemptive so that a crane can *take over* another crane's job before the job is completed. This assumption is acceptable because each job comprises many unloading and loading operations in practice. Preemptive jobs allow the cranes to dynamically share their workload in a more flexible manner. We will see in our analysis that this flexibility allows the system to effectively absorb the impact of shocks and to constantly maintain its efficiency.

Define the direction from left to right along the berth as *forward* and the reverse direction as *backward*. Each job is processed in the forward direction from its left-most bay to its right-most bay. For each bay, all import cargo is first unloaded and then all export cargo is loaded before the next bay is processed. A vessel leaves the berth after all its jobs are processed. An arriving vessel,

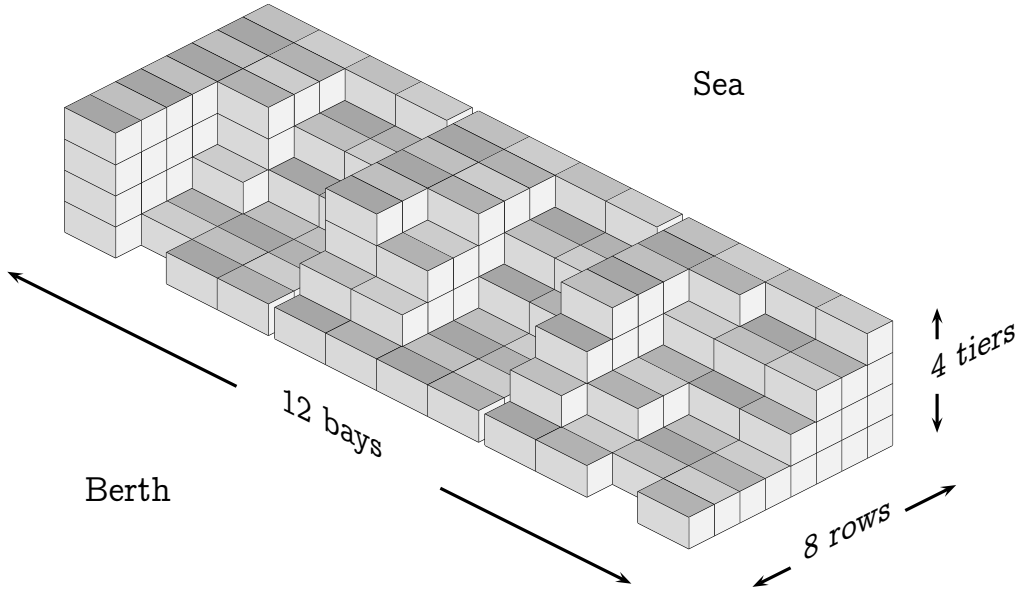


Fig. 2 Containers on a vessel.

if available, is then assigned to the corresponding berth position. Note that a departing vessel may be replaced by a shorter, arriving vessel at a berth position. This causes some empty space along the berth. We will see in the numerical simulations in Section 5 that this will not significantly affect the system's performance.

Cranes of the same berth remain in a fixed sequence because they move on a common rail track. We index the cranes from 1 to n in the forward direction. Cranes $i-1$ and $i+1$ are called the *predecessor* and *successor*, respectively, of crane i. Define the left-most point of the berth as the origin. Let B be the berth's length and z_i be the location of crane i along the berth. Both B and z_i are expressed in number of bays. Due to safety reasons, each crane must maintain a minimum distance d from its immediate neighbors. As a result, the condition $z_j \geq z_i + d$ must hold if crane j is on the right of crane i along the berth. Since the cranes cannot be too close to each other, each job is processed by at most one crane at any time.

Each quay crane handles (unloads or loads) one container at a time. Each crane i processes jobs with a constant rate μ_i (in number of containers per unit time), which is a parameter. We assume the processing rate of a crane is determined by the number of internal trucks supporting the crane. According to port executives, this assumption holds for congested ports such as Singapore because the time to unload or load a container at the berth is generally shorter than the time for the next internal truck to return from the congested yard. For simplicity, we assume the *turnover time* for an internal truck to travel from the berth to the yard and then back to the berth is a constant, which we define as a unit time. Based on the above assumptions, we can control the processing rate of a crane by assigning an appropriate number of internal trucks to it. For example, a crane with a rate of 5 containers per unit time means five internal trucks are assigned to the crane. The *capacity* of the system is the sum of all the cranes' processing rates.

Let α denote the average time for a crane to travel a bay along the berth. For example, a heavy crane moves about 44.4 feet per minute. Assuming a turnover time of 15 minutes and each bay is 40 feet long, we have $\alpha = 0.06$ unit time. According to port executives in Singapore, α is generally less than 0.2 unit time (that is, the average time for a crane to travel a bay is less than 20% of the turnover time of an internal truck).

3.2 Allocation of internal trucks

As mentioned above, we control the cranes' processing rates by allocating the internal trucks to the cranes. For example, if we want to set the processing rate of crane i to 3 containers per unit time,

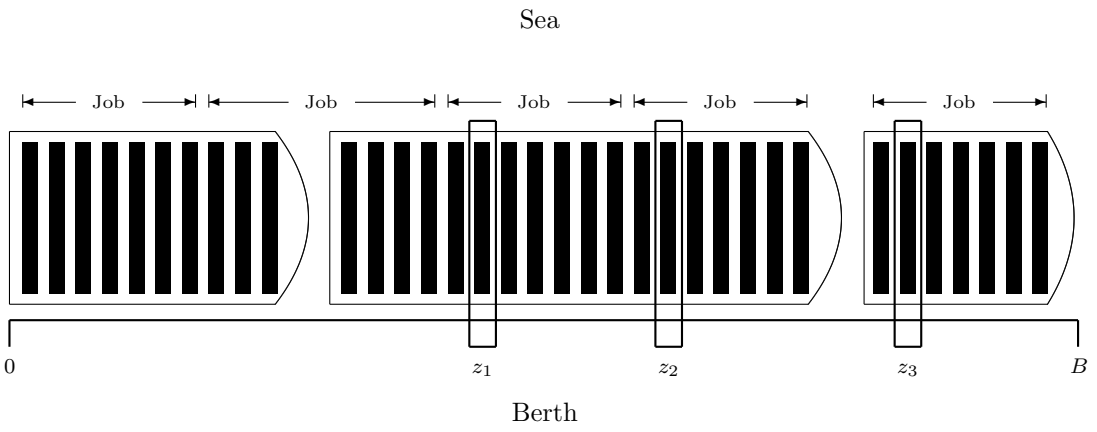


Fig. 3 A job is to handle containers for a set of adjacent bays on the vessels.

we can allocate three internal trucks to it. If any of these trucks are currently assigned to other cranes and are engaged in tasks such as carrying containers to the yard or being unloaded at the berth, then we require the trucks to serve crane i only after they complete their current tasks. We set different processing rates for different cranes such that some cranes can process their workload, *on average*, faster than others. According to port executives in Singapore, the reallocation of trucks to cranes can be done dynamically in a short time and it does not waste any significant capacity.

For congested ports, one may not be able to increase a berth's total processing rate by simply increasing the number of internal trucks. This is because the terminal will be more congested if more trucks are introduced. The turnover time of an internal truck generally increases as there are more trucks in the terminal. As a result, given a limited number of internal trucks, port operators strive to maximize their berths' efficiency through effective use of the quay cranes and internal trucks subject to various uncertain factors in terminal operations. We propose an operating protocol to achieve this in the following section.

3.3 An operating protocol

We initialize the system by assigning n jobs to the n cranes such that the i-th job from the right end of the berth is assigned to crane $n+1-i$. All cranes process their jobs by serving one bay after another in the forward direction. Each crane i continues its job until crane n completes a job and the system *resets* itself: Crane n moves backward to take over work from crane $n-1$, which moves backward to take over work from crane $n-2$ and so on, until crane 1 initiates a new job immediately on its left. Furthermore, if crane $i < n$ completes its job before its successor takes over the job, then crane i remains idle until its job is taken over.

When a crane is handling a container, it cannot be preempted by another crane. A crane can take over a job from its predecessor only after the latter has finished handling its current container. When the system resets itself we adopt a synchronous reset policy such that all cranes start moving backward simultaneously, immediately after all of them have finished handling their current containers. Suppose a reset begins at time s (that is, crane n finishes a job at time s). Let τ_i be the time when crane i finishes handling its latest container that is initiated before time s, for $i = 1, \ldots, n$. By definition, we have $\tau_n = s$. Note that if crane $i < n$ finishes a job before time s, then $\tau_i < s$. Define $\bar{\tau} = \max_{1 \le i \le n} \tau_i$. We require each crane i with $\tau_i < \bar{\tau}$ stays idle in the time interval $[\tau_i, \bar{\tau})$. According to our synchronous reset policy, all cranes start moving backward simultaneously at time $\bar{\tau}$. After all the cranes have reached their respective newly assigned jobs, the reset is completed and the cranes start processing their newly assigned jobs simultaneously. As a result, each reset is not instantaneous.

The system is in a *normal mode* if the cranes are indexed sequentially from 1 to n in the forward direction and there is at least a new job on the left of crane 1. Each crane i follows the *dynamic allocation rules* below to handle containers when the system is in a normal mode:

Process forward:
- If you are crane n, continue to process your job in the forward direction until you complete it. Then a reset begins and you **wait**.
- Otherwise, continue to process your job in the forward direction until
  1. a reset begins, then finish your current container and **wait**; or
  2. you complete your job, then remain idle until a reset begins and **wait**.

Wait: Remain idle until all cranes finished their current containers, then **move backward**.

Move backward: Take over work from your predecessor or initiate a new job immediately on your left if you are crane 1. When all cranes reach their newly assigned jobs then **process forward**.

In each reset crane 1 initiates a new job immediately on its left. What happens when crane 1 reaches the left-most job of the berth and there is no job on the left for crane 1 to initiate? We may allow crane 1 to be idle while other cranes are processing their respective jobs. Apparently, this strategy does not fully utilize the capacity of the system. Since there are new vessels arriving at the right of the berth, we can dispatch some cranes to serve these new vessels while other cranes continue serving the vessels on the left of the berth. We can achieve this by rotating the crane indices.

3.4 Rotation of crane indices

Define a *rotation function* $\sigma(i) = (i \bmod n) + 1$, which performs a cyclic permutation on the crane indices. For example, if $n = 3$, $\sigma(1) = 2$, $\sigma(2) = 3$, and $\sigma(3) = 1$. The system resets itself in a different way whenever crane n finishes a job, while crane 1 is with the left-most job of the berth. Figure 4 illustrates this type of resets for a system with three cranes (represented by $\bullet$, $*$, and $\circ$). Each rectangle in the graph corresponds to a job, which contains columns representing the job's remaining workload.

When crane 3 completes a job while crane 1 is processing or has completed the left-most job of the berth, the system resets itself by first applying the function $\sigma(i)$ on all the crane indices. Thus, crane 1 becomes crane 2, crane 2 becomes crane 3, and crane 3 becomes crane 1. Since the right-most crane becomes crane 1 after the rotation and its job is already completed, it can be dispatched to serve the new vessels (if any) arriving at the right of the berth. It immediately moves forward to initiate a new, right-most job of the berth (if there is no new job available, crane 1 moves to the right-most bay of the berth and remains idle). The reset is completed when crane 1 finishes its travel to the right. After the reset the processing rates of the cranes are adjusted according to their new indices.

Similarly, when crane 3 completes a job while crane 2 is processing or has completed the left-most job of the berth, the system resets itself by applying the function $\sigma(i)$ on all the crane indices. Immediately after the rotation crane 1 moves forward to initiate a new job immediately on the left of crane 2, which is now on the right end of the berth (if there is no new job available for crane 1, it remains idle at the bay immediately on the left of the job of crane 2). The system continues to rotate the crane indices for each reset until a normal mode is restored. Figure 4 shows a series of resets based on rotations until the system returns to a normal mode.

In general, when crane n finishes a job while there is a crane i, $1 \le i \le n$, with the left-most job of the berth (either crane i is still processing or has completed the job) the system resets itself in the following way: It instantaneously rotates crane indices by applying the function $\sigma(i)$ on them. Immediately after the rotation crane 1 moves forward to initiate a new, right-most available job. Thus, each reset is not instantaneous because it takes some time for crane 1 to travel to its new job. After each reset, the cranes' processing rates are adjusted according to their new indices. This can be achieved by reallocating the internal trucks to the cranes. For example, if the processing rate of crane 1 after a reset is 4 containers per unit time, then we assign four internal trucks to the crane indexed as 1 after the reset.

Both types of resets require communication among cranes. Adjusting the cranes' processing rates require reallocation of internal trucks to the cranes. All these can be achieved in practice as modern ports are equipped with integrated computer systems that allow them to coordinate the cranes and the internal trucks.

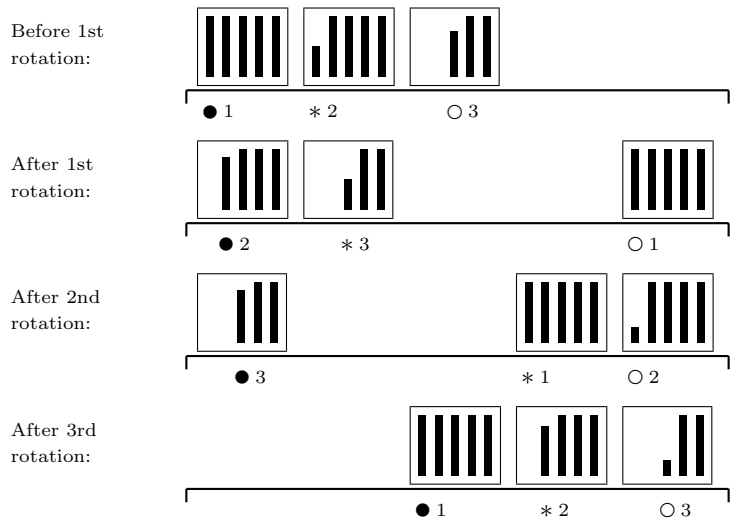


Fig. 4 Rotating crane indices.

3.5 Dynamics and throughput

We consider ports that are sufficiently congested so that new jobs are always available. This situation occurs, for example, at the port of Los Angeles, California, which constantly faces a long queue of vessels waiting to be served. Other major ports in the world such as Shanghai, Singapore, Hong Kong, and Rotterdam also face a similar situation. Due to long queues of vessels, these ports are especially keen in improving the operational efficiency of their berths. We will show analytically that our approach is promising in maximizing the system's throughput. Specifically, we study the dynamics and find a lower bound on the long-run average throughput of the system under the proposed dynamic allocation rules for such congested ports.

Since cranes process jobs in the forward direction along the berth and they must maintain a minimum distance from their immediate neighbors, a crane i is *blocked* by its neighboring crane j on the right when the former finishes processing its current bay and $z_i = z_j - d$. In this case, crane i remains idle until crane j finishes processing a bay. Both cranes then proceed simultaneously in the forward direction to their next respective bays. Note that blocking may occur regardless of whether the system is in a normal mode. If the system is not in a normal mode, crane 1 may be on the right of crane n after a rotation of indices (see Figure 4). We assume a long enough berth such that crane n will never be blocked by crane 1 in such a situation. Thus, crane n is never blocked and crane $i < n$ can only be blocked by its successor.

Let $x_i \in [0, 1]$ denote the fraction of workload completed on the job of crane i. Define $\mathbf{x} = (x_1, \ldots, x_n)$ as the *state* of the system. The following lemma shows that if the internal trucks are allocated such that the processing rates of cranes 1 to n are from slowest to fastest (equivalently, if $\mu_1 \leq \mu_2 \leq \cdots \leq \mu_n$), then the cranes will not be blocked after a transient period. Note that we can always allocate the internal trucks to the cranes such that the condition $\mu_1 \leq \mu_2 \leq \cdots \leq \mu_n$ is satisfied, regardless of whether the system is in a normal mode.

Lemma 1 *If $\mu_1 \leq \mu_2 \leq \cdots \leq \mu_n$, then blocking is transient.*

Proof Assume the system starts from an arbitrary state $\mathbf{x}$. We prove by induction. According to our assumption, crane n is never blocked. If crane $n-1$ is blocked by crane n, the former remains idle until the latter completes its current bay. After that both cranes process their next respective bays with their processing rates. Since $\mu_{n-1} \leq \mu_n$ and the workload of each bay is uniform, crane $n-1$ will never be blocked by crane n again. Therefore, blocking is transient for crane $n-1$.

Assume blocking is transient for crane $j+1$. After the transient period, if crane j is blocked by crane $j+1$, the former remains idle until the latter completes its current bay. After that both cranes process their next respective bays with their processing rates. Since $\mu_j \leq \mu_{j+1}$ and the

workload of each bay is uniform, crane j will never be blocked by crane $j+1$ again. Thus, blocking is also transient for crane j. This completes the proof. □

Although the non-crossing and the minimum separation constraints on the cranes may lead to blocking, Lemma 1 shows that this waste of system capacity disappears after a transient period if the internal trucks are assigned to cranes 1 to n such that their processing rates are from slowest to fastest.

Another type of waste occurs when a crane i completes a job before its successor takes over the job. According to the dynamic allocation rules, crane i remains idle until its job is taken over. We will show that this type of waste ceases after a transient period. We say crane i *overtakes* its successor if $x_i > x_{i+1}$. The following lemma shows that if the processing rates of cranes 1 to n are from slowest to fastest, then overtaking is also transient.

Lemma 2 *If $\mu_1 \le \mu_2 \le \cdots \le \mu_n$, then overtaking will cease after $n-1$ resets.*

Proof Assume the system starts from an arbitrary state $\mathbf{x}$. Since $\mu_1 \le \mu_2$ and the workload of each job is uniform, crane 1 will never overtake crane 2 after the first reset. Assume there is no overtaking among cranes $1, 2, \ldots, j$ after the $(j-1)$-st reset. Now consider the j-th reset, crane $j+1$ takes over a job from crane j. Since there is no overtaking among cranes $1, 2, \ldots, j$ after the $(j-1)$-st reset, we have $x_j \le x_{j+1}$ after the j-th reset. Crane j will not overtake crane $j+1$ afterwards because $\mu_j \le \mu_{j+1}$ and the workload of each job is uniform. Thus, there is no overtaking among cranes $1, 2, \ldots, j+1$ after the j-th reset. This shows that for a system with n cranes, overtaking will cease after $n-1$ resets. □

In the proofs of the above lemmas, we assume the system begins from an arbitrary initial state $\mathbf{x}$. We say the quay crane system attains its *efficient form* when both blocking and overtaking cease. Lemmas 1 and 2 imply that if the processing rates of cranes 1 to n are from slowest to fastest, then independent of its initial state, the system will always attain its efficient form after a transient period of time. When the system operates on its efficient form, all the cranes will be constantly busy serving the vessels. We summarize this result as follows.

Corollary 1 *If $\mu_1 \le \mu_2 \le \cdots \le \mu_n$, then independent of its initial state, the quay crane system always attains its efficient form after a transient period of time.*

After the system attains its efficient form it remains on its efficient form. This is because the processing rates of cranes 1 to n are from slowest to fastest and the workload is uniformly distributed. Upon its efficient form the only waste of the system's capacity occurs during the resets when the cranes move to their newly assigned jobs. Let m denote the number of containers to be handled (unloaded and loaded) per bay. The following theorem determines a lower bound on the average throughput of the system when it operates on its efficient form.

Theorem 1 *The average throughput of the quay crane system when it operates on its efficient form is at least*

$$\frac{\sum_{i=1}^{n} \mu_i}{1 + \max\{1 + (3b-2)\alpha, (B-nb)\alpha\}(\sum_{i=1}^{n} \mu_i)/(bm)}.$$

Proof After the system attains its efficient form, the only waste of system capacity is caused by non-instantaneous resets. We first calculate the maximum idle time during a reset when the system is in a normal mode. According to our synchronous policy, after a reset begins at time s all cranes start moving backward simultaneously at time $\bar{\tau}$. The idling time $\bar{\tau} - s$ is bounded by the turnover time of an internal truck (which equals 1 unit time). After all the cranes have reached their respective newly assigned jobs, which takes at most $(2b-1)\alpha$ unit time, the reset is completed and the cranes start processing their newly assigned jobs simultaneously. Thus, the maximum idling time during a reset is $1 + (2b-1)\alpha$. Since the system operates in its efficient form, it handles containers with the system capacity $\sum_{i=1}^{n} \mu_i$. Thus, the time to process a job with bm containers is bounded by $bm/\sum_{i=1}^{n} \mu_i + (b-1)\alpha$, where the term $(b-1)\alpha$ is the maximum crane travel time along the berth within a job. Thus, the intercompletion time of successive jobs is at most $bm/\sum_{i=1}^{n} \mu_i + (b-1)\alpha + 1 + (2b-1)\alpha = bm/\sum_{i=1}^{n} \mu_i + 1 + (3b-2)\alpha$.

When the system is not in a normal mode, the largest distance for a crane to travel along the berth during a reset is $B-nb-(b-1)$ bays. The intercompletion time of successive jobs is at most $bm/\sum_{i=1}^{n}\mu_i+(b-1)\alpha+[B-nb-(b-1)]\alpha = bm/\sum_{i=1}^{n}\mu_i+(B-nb)\alpha$.

Combining the two cases, the intercompletion time of successive jobs is at most $bm/\sum_{i=1}^{n}\mu_i+\max\{1+(3b-2)\alpha,(B-nb)\alpha\}$. The intercompletion time of successive containers is at most $1/\sum_{i=1}^{n}\mu_i+\max\{1+(3b-2)\alpha,(B-nb)\alpha\}/(bm)$. Thus, the average throughput of the system when it operates on its efficient form is at least

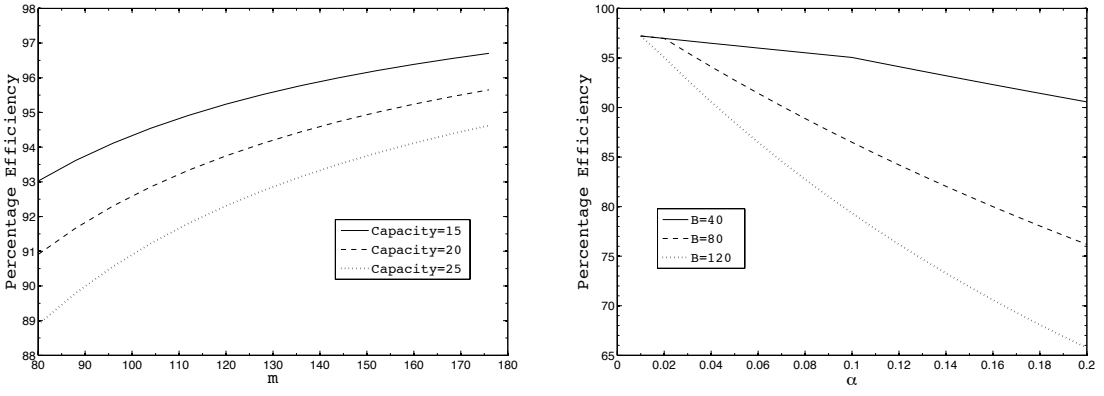
$$\frac{1}{1/\sum_{i=1}^{n}\mu_i+\max\{1+(3b-2)\alpha,(B-nb)\alpha\}/(bm)} = \frac{\sum_{i=1}^{n}\mu_i}{1+\max\{1+(3b-2)\alpha,(B-nb)\alpha\}(\sum_{i=1}^{n}\mu_i)/(bm)}.$$

□

Let ρ denote the average throughput of the quay crane system. Define *percentage efficiency* as $(\rho/\sum_{i=1}^{n}\mu_i)\times 100\%$. Figure 5(a) shows the lower bound on the percentage efficiency when the system operates on its efficient form for various m under different total processing rates. We set $B=40$, $b=4$, $n=5$, and $\alpha=0.06$. The results suggest that our approach is promising as the percentage efficiency is well above 90% for a wide range of m and system capacity. In Figure 5(b) we set $m=144$, $b=4$, $n=5$, and a system capacity of 15 containers per unit time. It shows that although the lower bound decreases with both α and B, the system operating on its efficient form is still highly efficient for small α. For example, for $\alpha\le 0.06$ the percentage efficiency is at least 96%, 91%, and 86% for a berth with 40, 80, and 120 bays respectively. Figure 5 suggests that the average throughput of the system operating on its efficient form is very close to its full capacity if α is sufficiently small (if the crane travel time per bay is sufficiently short compared to the turnover time of an internal truck).

Note that Corollary 1 and Theorem 1 only require *relative* processing rates of the cranes. All we need is a proper allocation of internal trucks to the cranes such that the processing rates of cranes 1 to n are from slowest to fastest. For example, consider a berth with 3 cranes that are supported by 15 internal trucks. Each truck has a processing rate of 1 container per unit time. The berth can allocate 2, 5, and 8 trucks to cranes 1, 2, and 3 respectively. This allocation ensures the system to attain its efficient form. Alternatively, one can allocate 4, 4, and 7 trucks to cranes 1, 2, and 3 respectively to achieve the same effect.

Since the cranes jointly serve from one end of a vessel to its other end, one may be concerned about the vessel's stability. We believe this should not be an issue under our protocol. This is because after each crane unloads import containers from a bay on a vessel, it immediately starts loading export containers to the same bay before it serves another bay.



(a) (b)

Fig. 5 Lower bounds on percentage efficiency of the system operating on its efficient form.

4 Self-recovery from shocks

The above analysis is based on a deterministic setting where the quay crane system starts from an arbitrary initial state. Without any disruptions, the system attains its efficient form and it remains on its efficient form. However, the system may lose its efficient form if it is subject to a shock. For example, a stowage plan could be disrupted, which causes the affected crane to block other cranes. We study how the system recovers from shocks in this section.

4.1 Self-recovery from a single shock

Suppose the system is subject to a single shock, which drives the system to a state $\tilde{\mathbf{x}}$ that leads to blocking or overtaking. In this situation, the system no longer operates on its efficient form. Fortunately, Corollary 1 ensures that the system can always spontaneously recover from state $\tilde{\mathbf{x}}$ to its efficient form after another transient period of time. Once the system recovers to its efficient form, Theorem 1 guarantees that its throughput will restore to a level near the system's full capacity. We say the system *self-recovers* from the shock.

One may argue that a new shock may occur before the system recovers to its efficient form from a previous shock. Is the self-recovery mechanism robust when the system is subject to a sequence of shocks? Will the throughput drop significantly when there are multiple shocks? We demonstrate how the system self-recovers from multiple, random shocks below.

4.2 Self-recovery from multiple shocks

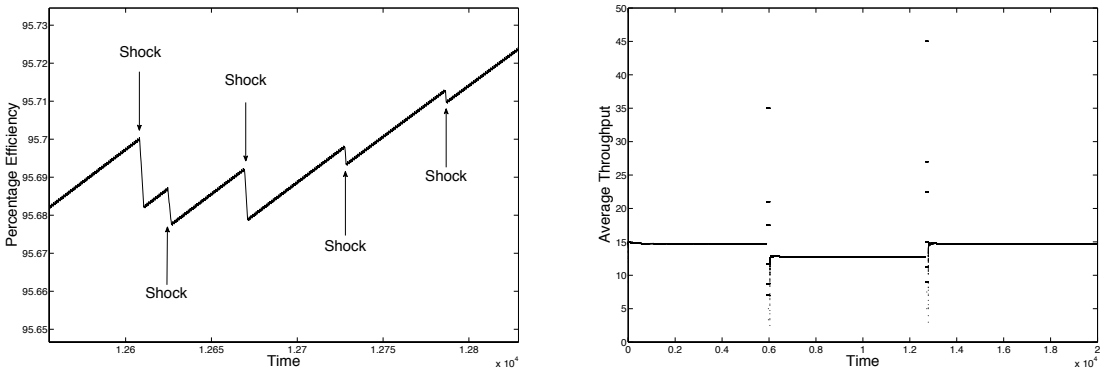
Figure 6(a) shows the percentage efficiency of the system under a sequence of shocks. We set $B = 40$, $b = 4$, $m = 108$, $d = 4$, $n = 3$, and $\alpha = 0.06$. Each shock delays the service of a random container, causing its handling duration to be 20 times of the regular handling duration. In practice, these shocks may represent disrupted stowage plans, which are often caused by misplacements of containers. Figure 6(a) shows that despite multiple perturbations, the system repeatedly self-recovers from the shocks and persistently restores its efficiency. Overall, the system maintains at high efficiency even though it is subject to these random shocks.

The self-recovery mechanism is especially appealing to terminal operators as there are always disruptions due to uncertainty in their operations. By following the simple protocol, the quay crane system can absorb the impacts of these random shocks and persistently restores its efficiency. More importantly, this is accomplished without complex computation and onerous re-planning, which are required by any traditional optimization-based approach. By totally eliminating these tedious procedures, the self-recovery mechanism could potentially save significant costs for terminal operators.

4.3 Responsiveness to longer disruptions

If an internal truck is suddenly and unexpectedly unavailable, for example, due to mechanical breakdown or more urgent assignments, the system's capacity will drop. Under a traditional optimization-based approach, this change may cause the original solution (or plan) to be obsolete or infeasible, which may require re-planning. In contrast, under our approach we only need to ensure that the processing rates of cranes 1 to n remain from slowest to fastest.

Using the same parameter values as in Figure 6(a), Figure 6(b) shows the impact on throughput when two trucks are removed from crane 3 at time 0.6×10^4: We change (μ_1, μ_2, μ_3) from $(1, 5, 9)$ to $(1, 5, 7)$ so that the system's capacity reduces from 15 to 13 containers per unit time. The system responds to this disruption by rebalancing the workload among the cranes according to the new truck allocation. After a transient period of time in which the system's throughput fluctuates drastically (which gives rise to the first peak in Figure 6(b)), the system recovers to its efficient form. The average throughput converges to and maintains at a new, lower value. When the two trucks are returned to crane 3 at about time 1.25×10^4, the system's throughput first fluctuates for a



(a) (b)

Fig. 6 Resilience of the quay crane system.

transient period of time (which gives rise to the second peak in Figure 6(b)) before it spontaneously restores to its original level.

Figure 6 shows that any breakdowns of trucks will affect the crane processing rates but not the applicability of the operating protocol. However, the protocol could no longer work if a quay crane breaks down. This is because the movement of (some of) the remaining cranes may be blocked. In this situation, one can split the system into two subsystems with the remaining cranes and reallocate the internal trucks to the subsystems according to the position of the crane that is down. Each subsystem can run the protocol independently.

In practice, the workload of each job is neither constant nor uniformly distributed on the bays. In this case blocking may persist and the system may never attain its efficient form. However, as we will see in the next section, the system's throughput remains near its full capacity as long as the processing rates of cranes 1 to n are from slowest to fastest.

5 The nonuniform workload case

If workload is not uniformly distributed on the bays, the model is effectively equivalent to the uniform workload case with frequent shocks. The analysis of the dynamics seems intractable if different bays have different workload. Thus, we perform simulation studies to investigate the influence of various parameters on the system's throughput. We study the impact of job size, difference in processing rate, crane travel time, variability in truck turnover time, and variability in number of bays per job.

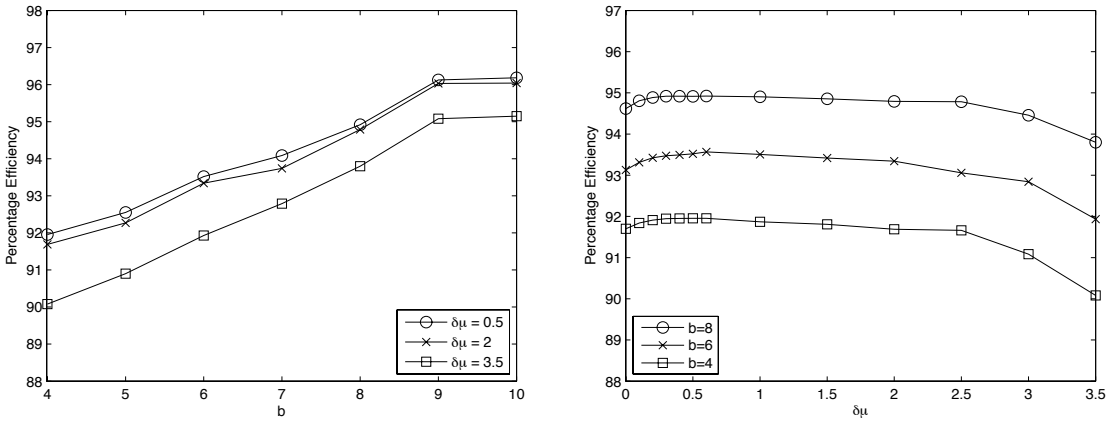
We consider a berth with $B = 100$ and $n = 6$. We set $\mu_i = 10 + (i - 3.5)\delta\mu, i = 1, \ldots, n$, where $\delta\mu$ represents the difference in processing rate between two successive cranes. The processing rates of cranes 1 to n are from slowest to fastest. The total processing rate of the system is 60 containers per unit time. We assume the number of tiers in each stack of containers to be unloaded from (or to be loaded to) a vessel is an integral random variable uniformly distributed in $[0, h]$, where h is a parameter. We set $d = 4$ bays, which is commonly adopted in practice. We observe similar results for $d = 3$ and 5 bays and so they are not reported in this paper. We first set $\alpha = 0.06$. The simulations are implemented in JAVA programming language and are run on a personal computer with a 1.8GHz Intel Core 2 Duo CPU and 2GB of memory. Each data point in the figures below represents the average result of 10 simulation runs. Each run introduces a sequence of vessels with random sizes ranging from 1 to 6 jobs. The total workload to be processed in each run is kept at 4,000 bays. Each bay contains 18 rows of containers. Most of the data points can be obtained within seconds.

5.1 Impact of job size

Figure 7(a) shows the percentage efficiency for various b (the number of bays per job) under different combinations of processing rates. We set $h = 5$. For each combination of processing rates, the percentage efficiency increases with b. As b increases, each crane handles more containers before each reset. Thus, the system's capacity is more utilized and its average throughput increases. Note that the percentage efficiency is at least 90% and can be as high as 96%. We observe similar results when we increase h. The system is more efficient when the average job size gets larger. This is because when each job contains more workload, the cranes spend more time handling containers rather than idling in resets.

5.2 Impact of difference in processing rate

In Figure 7(a), the percentage efficiency increases as the processing rates of different cranes get closer. One might expect to attain 100% efficiency by making the cranes homogeneous. Unfortunately, this is not the case. Figure 7(b) shows the percentage efficiency for various $\delta\mu$ under different values of b. We set $h = 5$. As $\delta\mu$ increases, the efficiency first increases until it reaches its highest value before it starts to drop. The efficiency peaks when $\delta\mu \in [0.4, 0.6]$. As $\delta\mu \to 0$, the efficiency decreases. This is because the cranes are more likely to be blocked if they have similar rates. On the other side of the peak of efficiency, the system's throughput decreases with $\delta\mu$. This is because during each reset in a normal mode, the cranes generally wait for the final crane, which is often the slowest crane (crane 1), to finish handling its current container before they can take over jobs from their predecessors. As $\delta\mu$ increases, crane 1 becomes slower, causing a longer average waiting time for other cranes in each reset.

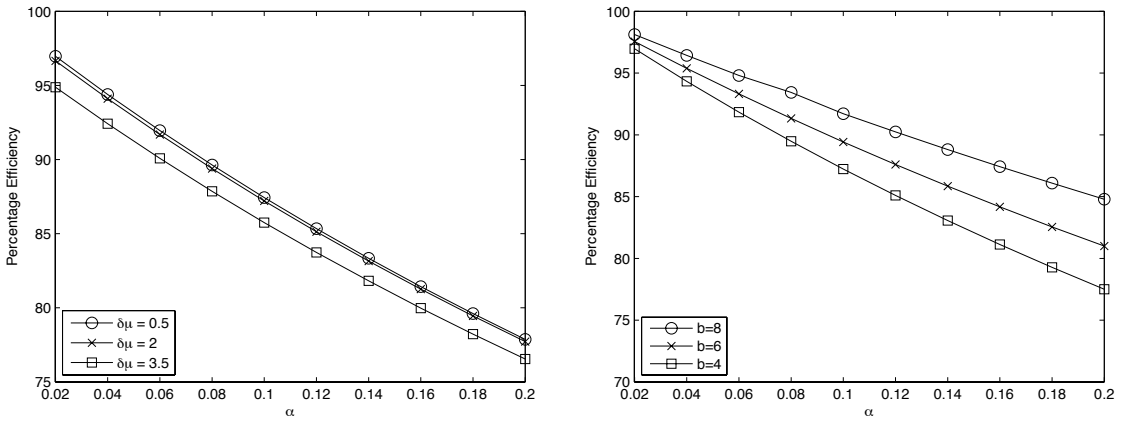


(a) (b)

Fig. 7 Impact of job size and of difference in processing rate.

5.3 Impact of crane travel time

Figure 8(a) shows how the efficiency drops as α increases under different combinations of crane processing rates with $b = 4$ and $h = 5$. For each combination of rates, the percentage efficiency decreases almost linearly with α. We find similar results in Figure 8(b), where we set $\delta\mu = 0.5$ and use different values of b. Figure 8 suggests that although the efficiency decreases with α, the system remains highly efficient for a broad range of settings. Even if $\alpha = 0.2$ (the crane travel time per bay is 20% of the truck turnover time), the system is still above 76% efficient.



(a) (b)

Fig. 8 Impact of crane travel time.

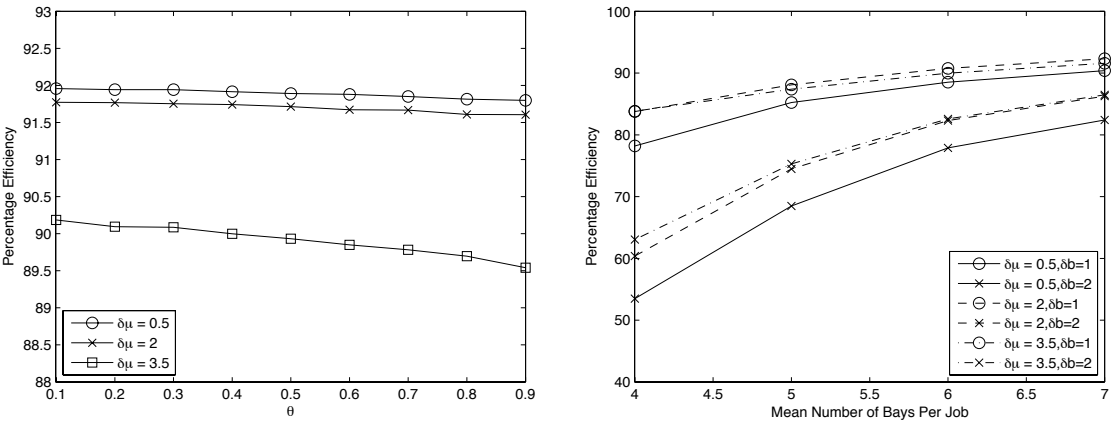
5.4 Impact of variability in truck turnover time

Recall that the turnover time is the time duration for a truck to travel from the berth to the yard and then back to the berth. It depends on the storage location in the yard of the container being carried and the stowage plan in the yard. In practice, the turnover time is hardly constant because different containers are stored at different locations in the yard and the truck may get stuck in the traffic along the way. Furthermore, depending on the stowage plan, the loading or unloading time in the yard is generally not constant and can be uncertain. To see the impact of variability in turnover time, we assume the turnover time is equal to $1 + \epsilon$, where ϵ is a random variable uniformly distributed in $[-\theta, \theta]$. The variability in turnover time will cause the affected cranes to block other cranes and will result in lower efficiency. However, as shown in Figure 9(a) with $b = 4$, $h = 6$, and $\alpha = 0.06$, the system's performance only drops slightly as the variability (θ) in turnover time increases. The system remains above 89.5% efficient for a wide range of θ. We have also tried other distributions of ϵ with the same support on $[-\theta, \theta]$, such as a truncated normal distribution and a beta distribution with two peaks at the boundaries ($Beta(1/3, 1/3)$). The performance of the system under these distributions is very similar to that shown in Figure 9(a).

5.5 Impact of variability in number of bays per job

In this section we assume each job occupies a constant number $\bar{b}+\delta b$ of bays, but among them only $\bar{b}+X_b$ bays are non-empty, where $\bar{b}$ represents the mean number of non-empty bays per job and X_b is a random variable uniformly distributed in $[-\delta b, \delta b]$. Figure 9(b) shows the percentage efficiency of the system under three different combinations of processing rates. We set $h = 6$ and $\alpha = 0.06$. For each combination of rates, we consider two different levels of variability with $\delta b = 1$ and 2. The results suggest that the system is more efficient if δb is low. The difference in performance between low and high variability decreases as the rates of different cranes become more different (the gap between the curves corresponding to the cases with $\delta b = 1$ and 2 is the smallest for $\delta\mu = 3.5$). The system is less sensitive to the variability in number of bays per job if the cranes have very different processing rates. Figure 9(b) also suggests that the system becomes more efficient as the mean number of bays per job $\bar{b}$ increases. This echoes the results of Section 5.1. Finally, we do not find any single combination of processing rates that dominates other combinations.

Although we cannot analytically obtain any performance guarantee for the nonuniform workload model, the numerical studies above demonstrate that by following the simple protocol the system can absorb the variability in workload, turnover time, and number of bays per job so that it remains highly efficient.



(a) (b)

Fig. 9 Impact of variability in truck turnover time and of variability in number of bays per job.

6 Conclusion

A key objective of container terminals is to serve arriving vessels swiftly. This requires effective coordination of quay cranes to maximize the throughput subject to operational constraints. The problem is further complicated by various types of uncertainty in the terminal operations.

Most of the existing methods try to achieve this objective by solving a quay crane scheduling problem. Typically, these methods construct a plan for a finite time horizon and they handle uncertainty by revising or reconstructing the plan when it is affected by shocks. Thus, they can be quite tedious to implement in practice. To counter this, we propose a new and potentially applicable operating protocol to coordinate the quay cranes. The protocol is easy to implement and does not require any complex computation to respond to shocks. Under our approach, the non-crossing and the minimum separation constraints can be fulfilled naturally and the system can absorb the impact of random shocks.

To understand the system's dynamics, we first analyze a model where workload is uniformly distributed on each bay of a vessel. If the processing rates of cranes 1 to n are from slowest to fastest, then independent of its initial state, the system always attains its efficient form after a transient period. This implies that, after a single shock, the system can recover to its efficient form. When the system operates on its efficient form its average throughput is very near its full capacity if the crane travel time per bay is relatively short compared to the turnover time of an internal truck (if $\alpha \leq 0.06$). When the system is subject to a sequence of random shocks, it repeatedly recovers from each shock and persistently restores its throughput. All this can be achieved without complex computation or onerous re-planning.

If workload is nonuniform, the system may not attain its efficient form because blocking may persist. However, numerical experiments suggest that our approach remains effective as long as the processing rates of cranes 1 to n are from slowest to fastest. Specifically, the system attains well above 90% efficiency for a broad range of system settings if $\alpha \leq 0.06$. The average throughput increases with the job size. We also find that given a fixed capacity, a manager should avoid adopting crane processing rates that are too similar ($\delta\mu$ is too small) or too different from each other ($\delta\mu$ is too large) in order to maximize the system's throughput. The system's performance is robust in more general settings where the turnover time of internal trucks is variable and the number of bays per job is nonuniform.

Our approach is appealing to port operators as it requires neither a system reconfiguration nor any investment in facility. No sophisticated software is required and thus, maintenance cost is insignificant. No precise information on vessel arrival and departure times is required because the system spontaneously reallocates the workload when circumstances change. All we need is to group containers into jobs and allocate trucks to the cranes such that the processing rates of cranes 1 to

n are from slowest to fastest. The result is that the system can constantly maintain its throughput near its full capacity even though it is subject to shocks in the operations.

In this paper, we assume the processing rate of a crane is determined by the number of internal trucks supporting the crane. This assumption holds mainly for congested ports, such as Singapore, where the time to unload or load a container at the berth is significantly shorter than the turnover time of an internal truck. This assumption may not hold in other situations. Furthermore, this paper assumes the objective is to maximize the long-run average throughput of the entire quay crane system. For other objectives such as minimizing the service time (makespan) of each individual vessel, the proposed protocol may not be suitable.

References

Adida, E., G. Perakis. 2006. A robust optimization approach to dynamic pricing and inventory control with no backorders. *Math. Progr.* **107**(1-2) 97–129.

Bartholdi, J.J. III, D.D. Eisenstein. 1996. A production line that balances itself. *Oper. Res.* **44**(1) 21–34.

Bartholdi, J.J. III, D.D. Eisenstein. 2005. Using bucket brigades to migrate from craft manufacturing to assembly lines. *Manufacturing Service Oper. Management* **7**(2) 121–129.

Bartholdi, J.J. III, D.D. Eisenstein. 2012. A self-coordinating bus route to resist bus bunching. *Transport. Res.* **46B** 481–491.

Bartholdi, J.J. III, D.D. Eisenstein, R.D. Foley. 2001. Performance of bucket brigades when work is stochastic. *Oper. Res.* **49**(5) 710–719.

Bartholdi, J.J. III, D.D. Eisenstein, Y.F. Lim. 2006. Bucket brigades on in-tree assembly networks. *Eur. J. Oper. Res.* **168**(3) 870–879.

Bartholdi, J.J. III, D.D. Eisenstein, Y.F. Lim. 2009. Deterministic chaos in a model of discrete manufacturing. *Naval Res. Logist.* **56**(4) 293–299.

Bartholdi, J.J. III, D.D. Eisenstein, Y.F. Lim. 2010. Self-organizing logistics systems. *Ann. Rev. in Control* **34**(1) 111–117.

Ben-Tal, A., B. Golany, A. Nemirovski, J. Vial. 2005. Supplier-retailer flexible commitments contracts: A robust optimization approach. *Manufacturing Service Oper. Management* **7**(3) 248–273.

Ben-Tal, A., A. Goryashko, E. Guslitzer, A. Nemirovski. 2004. Adjustable robust solutions of uncertain linear programs. *Math. Progr.* **99** 351–376.

Ben-Tal, A., A. Nemirovski. 1999. Robust solutions to uncertain programs. *Oper. Res. Let.* **25** 1–13.

Ben-Tal, A., A. Nemirovski. 2000. Robust solutions of linear programming problems contaminated with uncertain data. *Math. Progr.* **88** 411–424.

Bertsimas, D., D.A. Iancu, P.A. Parrilo. 2010. Optimality of affine policies in multistage robust optimization. *Math. Oper. Res.* **35**(2) 363–394.

Bertsimas, D., M. Sim. 2003. Robust discrete optimization and network flows. *Math. Progr.* **98** 49–71.

Bertsimas, D., M. Sim. 2004. The price of robustness. *Oper. Res.* **52**(1) 35–53.

Bertsimas, D., A. Thiele. 2006. A robust optimization approach to inventory theory. *Oper. Res.* **54**(1) 150–168.

Bierwirth, C., F. Meisel. 2010. A survey of berth allocation and quay crane scheduling problems in container terminals. *Eur. J. Oper. Res.* **202**(3) 615–627.

Blażewicz J., T.C.E. Cheng, M. Machowiak, C. Oğuz. 2011. Berth and quay crane allocation: a moldable task scheduling model. *J. Oper. Res. Soc.* **62**(7) 1189–1197.

Daganzo, C.F. 1989. The crane scheduling problem. *Transport. Res.* **23B**(3) 159–176.

Eisenstein, D.D. 2005. Recovering cyclic schedules using dynamic produce-up-to policies. *Oper. Res.* **53**(4) 675–688.

Giallombardo, G., L. Moccia, M. Salani, I. Vacca. 2008. The tactical berth allocation problem with quay crane assignment and transshipment-related quadratic yard costs. In: Proceedings of the European Transport Conference (ETC), pp. 1–27.

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

Guan, Y., K.-H. Yang. 2010. Analysis of berth allocation and inspection operations in a container terminal. *Maritime Econ. Logist.* **12**(4) 347–369.

Guan, Y., K.-H. Yang, Z. Zhou. 2010. The crane scheduling problem: models and solution approaches. *Ann. Oper. Res.* DOI 10.1007/s10479-010-0765-3.

Han, X., Z. Lu, L. Xi. 2010. A proactive approach for simultaneous berth and quay crane scheduling problem with stochastic arrival and handling time. *Eur. J. Oper. Res.* **207** 1327–1340.

Hendriks, M., M. Laumanns, E. Lefeber, J.T. Udding. 2010. Robust cyclic berth planning of container vessels. *OR Spectrum* **32** 501–517.

Imai, A., H.C. Chen, E. Nishimura, S. Papadimitriou. 2008. The simultaneous berth and quay crane allocation problem. *Transport. Res.* **44E**(5) 900–920.

Kim, K.H., Y.M. Park. 2004. A crane scheduling method for port container terminals. *Eur. J. Oper. Res.* **156** 752–768.

Lee, D.-H., L. Song, H. Wang, 2006. Bilevel programming model and solutions of berth allocation and quay crane scheduling. In: Proceedings of 85th Annual Meeting of Transportation Research Board (CD-ROM). Annual Meeting of Transportation Research Board. Washington DC.

Lee, D.-H., H.Q. Wang, L. Miao. 2008. Quay crane scheduling with non-interference constraints in port container terminals. *Trasport. Res.* **44E** 124–135.

Legato, P., R.M. Mazza, R. Trunfio. 2010. Simulation-based optimization for discharge/loading operations at a maritime container terminal. *OR Spectrum* **32** 543–567.

Legato, P., R. Trunfio, F. Meisel. 2012. Modeling and solving rich quay crane scheduling problems. *Comp. Oper. Res.* **39** 2063–2078.

Li, W., Y. Wu, M.E.H. Petering, M. Goh, R. de Souza. 2009. Discrete time model and algorithms for container yard crane scheduling. *Eur. J. Oper. Res.* **198** 165–172.

Lim, A., B. Rodrigues, F. Xiao, Y. Zhu. 2004. Crane scheduling with spatial constraints. *Naval Res. Logist.* **51**(3) 386–406.

Lim, A., B. Rodrigues, Z. Xu. 2007. A m-parallel crane scheduling problem with a non-crossing constraint. *Naval Res. Logist.* **54** 115–127.

Lim, Y.F. 2011. Cellular bucket brigades. *Oper. Res.* **59**(6) 1539–1545.

Lim, Y.F. 2012. Order-picking by cellular bucket brigades: A case study. R. Manzini, ed. *Warehousing in The Global Supply Chain*. Springer-Verlag, London, 71–85.

Lim, Y.F., Y. Wu. 2014. Cellular bucket brigades on U-lines with discrete work stations. *Production Oper. Management* **23**(7), 1113–1128.

Lim, Y.F., K.K. Yang. 2009. Maximizing throughput of bucket brigades on discrete work stations. *Production Oper. Management* **18**(1) 48–59.

Liu, J., Y. Wan, L. Wang. 2006. Quay crane scheduling at container terminals to minimize the maximum relative tardiness of vessel departures. *Naval Res. Logist.* **53** 60–74.

Meier, L., R. Schumann, 2007. Coordination of interdependent planning systems, a case study. In: R. Koschke, H. Otthein, K.-H. Rödiger, M. Ronthaler (Eds.), Lecture Notes in Informatics (LNI) P-109, Köllen Druck+Verlag GmbH, Bonn, pp. 389–396.

Meisel, F. 2009. *Seaside Operations Planning in Container Terminals*. Physica-Verlag, Berlin Heidelberg. ISBN 978-3-7908-2190-1.

Meisel, F. 2011. The quay crane scheduling problem with time windows. *Naval Res. Logist.* **58** 619–636.

Meisel, F., C. Bierwirth, 2006. Integration of berth allocation and crane assignment to improve the resource utilization at a seaport container terminal. In: H.-D. Haasis, H. Kopfer, J. Schönberger (Eds.), Operations Research Proceedings 2005, Springer, Berlin, pp. 105–110.

Meisel, F., C. Bierwirth, 2009. Heuristics for the integration of crane productivity in the berth allocation problem. *Transport. Res.* **45E**(1) 196–209.

Meisel, F., C. Bierwirth. 2011. A unified approach for the evaluation of quay crane scheduling models and algorithms. *Comp. Oper. Res.* **38** 683–693.

Meisel, F., C. Bierwirth. 2013. A framework for integrated berth allocation and crane operations planning in seaport container terminals. *Transportation Sci.* **47**(2) 131–147.

Oğuz, C., J. Blażewicz, T.C.E. Cheng, M. Machowiak, 2004. Berth allocation as a moldable task scheduling problem. In: Proceedings of the 9th International Workshop on Project Management and Scheduling (PMS 2004), Nancy, pp. 201–205.

Park, Y.M., K.H. Kim. 2003. A scheduling method for berth and quay cranes. *OR Spectrum* **25**(1) 1–23.

Peterkofsky, R.I., C.F. Daganzo. 1990. A branch and bound solution method for the crane scheduling problem. *Transport. Res.* **24B**(3) 139–172.

See, C., M. Sim. 2010. Robust approximation to multi-period inventory management. *Oper. Res.* **58**(3) 583–594.

Theofanis, S., M. Golias, M. Boile. 2007. Berth and quay crane scheduling: a formulation reflecting service deadlines and productivity agreements. In: Proceedings of the International Conference on Transport Science and Technology (TRANSTEC 2007), Prague, pp. 124–140.

UNCTAD. 2009. *Review of Maritime Transport*. United Nations Conference on Trade and Development.

Xu, Y., Q. Chen, X. Quan. 2012. Robust berth scheduling with uncertain vessel delay and handling time. *Ann. Oper. Res.* **192** 123–140.

Zeng, Q., Z. Yang, X. Hu. 2011. Disruption recovery model for berth and quay crane scheduling in container terminals. *Engineering Optimization* **43**(9) 967–983.

Zhen, L., L.H. Lee, E.P. Chew. 2011. A decision model for berth allocation under uncertainty. *Eur. J. Oper. Res.* **212** 54–68.

Zhu, Y., A. Lim. 2006. Crane scheduling with non-crossing constraint. *J. Oper. Res. Soc.* **57**(12) 1464–1471.