OR Spectrum (2005) 27: 491–506
DOI: 10.1007/s00291-004-0194-2

OR Spectrum

Dispatching vehicles in a mega container terminal*

Ebru K. Bish[1], Frank Y. Chen[2], Yin Thin Leong[3], Barry L. Nelson[4], Jonathan Wing Cheong Ng[5], and David Simchi-Levi[6]

[1] Virginia Polytechnic Institute and State University, Department of Industrial and Systems Engg., Blacksburg, VA 24061-0118, USA
[2] The Chinese University of Hong Kong, Department of System Engg. and Engg. Mgmt., NT, Hong Kong, China (e-mail: yhchen@se.cuhk.edu.hk)
[3] Port of Singapore Authority (PSA), Singapore, Singapore
[4] Northwestern University, Department of Industrial Engg. and Management Sciences, Evanston, IL, USA
[5] University of Hong Kong, Department of Industrial and Manufacturing Systems Engg., Hong Kong, China
[6] Massachusetts Institute of Technology, Department of Civil and Environmental Engg., Cambridge, MA, USA

Abstract. We consider a container terminal discharging and uploading containers to and from ships. The discharged containers are stored at prespecified storage locations in the terminal yard. Containers are moved between the ship area and the yard using a fleet of vehicles, each of which can carry one container at a time. The problem is to dispatch vehicles to the containers so as to minimize the total time it takes to serve a ship, which is the total time it takes to discharge all containers from the ship and upload new containers onto the ship. We develop easily implementable heuristic algorithms and identify both the absolute and asymptotic worst-case performance ratios of these heuristics. In simple settings, most of these algorithms are optimal, while in more general settings, we show, through numerical experiments, that these algorithms obtain near-optimal results for the dispatching problem.

Keywords: Port terminal operations – Vehicle dispatching – Heuristics

1 Introduction and motivation

In the last few years we have seen the breakdown of many trade barriers and the globalization of trade. These developments have increased the importance of logistics and transportation, and in particular, the importance of marine transportation

* Research was supported in part by the Port of Singapore Authority (PSA).
Correspondence to: F.Y. Chen

systems. These systems include a network of terminals around the globe that allow manufacturers and shippers to deliver goods quickly to their customers. These terminals serve as hubs for the transshipment of containers from ship to ship or to other modes of transportation, e.g., rail and trucks. In this paper we analyze a container terminal, where the majority of the terminal operations consists of ship-to-ship transshipments.

In today's competitive market place, a speedy transshipment of containers to and from ships is important to both the carrier, since it provides significant operational efficiencies, and to the terminal, which can handle a large number of ships per day. Unfortunately, in many regions around the globe, the terminals are now working at, or close to, capacity and there is significant pressure from the business sectors to increase terminal throughput and, in particular, to *decrease ship turnaround time at the terminal*. In most cases, this requires the development of methodology and tools which will allow the efficient coordination of activities within the terminal area. In this paper we consider one aspect of the terminal operation, which is to dispatch vehicles to containers in the terminal. This research was motivated by our industry partner, who operates a major container terminal. In what follows, we describe the operations of this container terminal, noting here that most container terminals operate in a similar way.

When a ship arrives at the terminal, containers are first discharged from the ship onto vehicles by *quay cranes*; the vehicles then transport the containers to various storage locations in the *yard area*. Typically, after most, or all, containers have been discharged from the ship, other containers are uploaded onto the ship. These containers are carried by vehicles from the yard to the *ship area*, and are loaded onto to ship by the quay cranes. Thus, two types of cranes exist in the terminal: *quay cranes*, which are used to load and unload containers to and from the ship, and *yard cranes*, which are used to load and unload containers at the terminal yard storage area.

Most containers handled by the terminal are standard size (Twenty-foot-equivalent unit (TEU)) containers. Due to the large container sizes, a crane would unload a container only onto a vehicle; unloading to the ground would require additional crane operation to lift the container from the ground and load onto a vehicle, and therefore, is not desirable. Thus, a vehicle needs to be available by the crane throughout the loading and unloading operations. This constraint will be further discussed in Section 3.

The terminal typically handles a small number of ships at a time, and each ship is served by a number of quay cranes. A few hours before the arrival of an incoming ship, the terminal receives detailed information about its contents; i.e., containers that are to be discharged into the yard, as well as a list of containers currently in the yard that should be uploaded onto the ship. This information allows the terminal dispatchers to generate the so-called *crane job sequence*: For each quay crane serving the ship, a detailed sequence specifying the order of the containers that are to be discharged/loaded onto the ship. This sequence is mainly determined by the current positions of the containers on the ship, their destinations and contents. Containers can be stacked on top of each other on the ship. Thus, the sequence in which containers will be discharged is based on the containers' current positions

on the ship. Similarly, the sequence in which containers will be uploaded onto the ship is based on the contents of the containers (i.e., if a container is carrying delicate items, then no container can be stored on top of it on the ship; thus, this container must be stored at the top of a stack). This information, together with the container's destination, is used to determine an uploading sequence. Finally, of course, the sequence always starts with discharged containers, which are followed by the containers loaded onto the ship.

Thus, at any point in time, the quay crane operator has information on the next container he/she is going to work on. If this is a container to be discharged from the ship, then the crane sequence will identify a number of potential storage locations, typically two to four, in the yard for this container. If this is a container to be loaded onto the ship, then the crane sequence also identifies the current location of the container in the yard area.

It is no surprise that managing, controlling and operating such a system is very complex. At the operational level the questions are clear: how should vehicles be dispatched to containers, what is an optimal location for a container discharged from the ship, how should vehicles be routed in this complex network, and what is an effective traffic control mechanism? Similarly, at the strategic level, the issues include optimizing the number of quay cranes, vehicles and yard cranes.

Evidently, these issues are interrelated. Unfortunately, solving a single integrated model that addresses, for instance, all the operational decisions, is well beyond today's computing capability. For that reason, in this research we decompose the problem into several related models: dispatching vehicles to containers, assigning discharged containers to specific locations, and routing vehicles. Our approach is to analyze each model separately in order to gain an insight into the system (see [4]). In this study, we focus on the problem of dispatching vehicles to the containers for a single ship, assuming that a fleet of vehicles are already assigned to this ship. In doing so, we treat other aspects of the system management as given inputs. This includes selecting an appropriate location for a discharged container, vehicle routing, traffic control, etc. Specifically, we focus on the impacts of vehicle deployment on the system throughput. Our objective is to find *easily implementable* vehicle dispatching policies that *minimize the ship makespan*, which is the time the last vehicle returns to the ship area after all containers are discharged from the ship and are taken to their storage locations in the yard, and all new containers are uploaded onto the ship. We refer to this problem as the *vehicle dispatching problem*.

This paper is organized as follows. In the next section, we give a brief review of the related literature. In Section 3, we consider the vehicle dispatching model for a single ship with a single quay crane, and analyze the performance of different vehicle dispatching policies on discharging job sequences, uploading job sequences, and combined job sequences. Based on the insights obtained for these simple models, in Section 4 we analyze a more general model of a single ship with multiple quay cranes, and test the performance of the proposed heuristics using computational analysis. Finally, in Section 5, we discuss future research directions and extensions to the vehicle dispatching problem.

2 Literature review

Problems associated with dispatching and routing vehicles arise frequently in *logistics systems*, see, for instance, Bramel and Simchi-Levi [5]. Thus, these problems have been extensively studied in the operations research/management science literature under different settings including, but not limited to, vehicle fleet management, truck routing, and warehouse management. Unfortunately, most of this research is not directly applicable to a container terminal operation due to its unique characteristics. This, in turn, requires the development of algorithms that take into account the special characteristics and constraints associated with container terminals.

This review is not meant to be exhaustive, but rather indicative of the recent developments that are most related to the problem analyzed here; see Bish [2] and Bish et al. [3] for more extensive reviews of the other related areas, such as material handling systems and resource-constrained scheduling, and Steenken, Voss and Stahlbock [16] and Vis and De Koster [19] for overviews of container terminal operations research.

Most of the literature on container terminals has used queuing theory to analyze terminal operations. These queuing models focus on strategic issues such as determining the equipment capacity, both on the water-side (such as berth capacity), and on the land-side (such as the number of quay cranes, vehicles, and yard cranes); see, for instance, Daganzo [7]. Several researchers focus on the operational level issues, such as scheduling the cranes and determining storage locations for the unloaded containers (see, for instance [2,3,6,10–15]).

Most recently, Kim and Bae [9] develop vehicle dispatching methods in container terminals by utilizing information on locations and times of future delivery tasks. They develop a mixed-integer programming model for assigning optimal delivery tasks to vehicles. Since the mathematical model requires an excessive amount of computational time, they also propose a heuristic algorithm; their numerical study indicates that the proposed heuristic is quite effective. Vis, De Koster and Savelsbergh [18] also consider the transport of containers between the ship and the yard, with the objective of minimizing the number of vehicles used. These two papers assume that each vehicle has a unit-load capacity. Grunow, Gunther and Lehmann [8] further analyze dispatching methods for multi-load vehicles in highly automated container terminals. This stream of research focuses on equipment allocation and dispatching problems, while Vis and Harika [17] and Yang, Choi and Ha [20] evaluate the relative performance of AGVs (Automated Guided Vehicle) and ALVs (Automated Lifting Vehicle) at container terminals.

In this paper, our objective is to develop algorithms that are easy to implement, especially for *large problem sizes*, and whose effectiveness can be characterized analytically. For this purpose, we focus on *simple* vehicle dispatching rules, and develop analytical bounds on the deviation of the heuristic solution from the optimal solution for any problem instance as well as for large problem instances, and complement our analysis with a numerical study.

3 The vehicle dispatching problem: A single crane model

In what follows, we first analyze the vehicle dispatching problem by focusing on a single ship single quay crane model, and obtain insights into the effectiveness of various algorithms for different instances of this problem. In Section 4, we use these insights to analyze a more general problem with multiple quay cranes.

Thus, we first consider a single ship served by a single quay crane with a fixed number, k, of vehicles assigned to it. We assume that all the vehicles are initially at the ship area and return to the ship area after completing the discharging and uploading of the ship. Throughout, we use the terms dispatching policy and algorithm interchangeably, and we refer to each container as a *job*. Throughout the paper, we assume that the yard cranes are always available (similar assumptions are used in other papers on container terminal operations; see, for instance, Kim and Bae [9]) and all operation times are deterministic (however, as will be shown in the sequel, some of our results still hold even when these operation times are random).

As mentioned above, our objective is to find an effective dispatching policy that assigns vehicles to jobs so as to *minimize the makespan*. In the vehicle dispatching problem, makespan is the time the last vehicle returns to the ship area after all containers are discharged and are taken to their locations in the yard, and after all containers are uploaded onto the ship.

Associated with the quay crane is a predetermined *crane job sequence*,

$$J_{-/+} : \{J_1, J_2, \cdots, J_n\},$$

with J_i, $i = 1, 2, \ldots, n$, being either a job to be discharged from the ship (denoted by a "–" job) or a job to be loaded onto the ship (denoted by a "+" job). The job sequence may consist of only "–" jobs, in which case it is denoted by J_-, only "+" jobs, in which case it is denoted by J_+, or a mix of "–" and "+"jobs, in which case it is denoted by $J_{-/+}$.

If the job sequence is a $J_{-/+}$ sequence, then it consists of two parts: the first part includes all the jobs to be discharged from the ship, that is, all the "–" jobs, while the second part includes all the jobs to be loaded onto the ship, that is, all the "+" jobs.

Obviously, this predetermined job sequence imposes precedence constraints among the jobs. That is, a "–" job cannot be discharged until all "–" jobs preceding it in the job sequence are discharged; in other words, the quay crane cannot start the task of discharging a specific "–" job until all its predecessor "–" jobs have been discharged from the ship. Similarly, the quay crane cannot load a "+" job until all "+" jobs preceding it in the job sequence are loaded onto the ship. Finally, a "+" job cannot be loaded until all "–" jobs in the sequence have been discharged.

Each "–" ("+") job requires a crane movement that will *lift* it up from the ship (or the vehicle), and *place* it onto a vehicle (or the ship). Clearly, a vehicle needs to be available by the crane only during the time the crane is *placing* the job onto the vehicle. Thus, the total crane processing time of a job consists of two components: one is the lifting time, the other is the placing time (during which a vehicle is needed by the crane). For notational convenience and simplicity, we

do not distinguish between these two components, and assume that each vehicle needs to be available by the crane throughout the discharging/uploading process. However, the subsequent analysis can be easily modified to handle the case where there are two separate components for crane processing, and a vehicle needs to be available only during the placing time. In our analysis, we assume that the time required to discharge/upload a container by the quay crane is deterministic, and is the same for all the jobs. We denote this time by s.

Containers are carried between the ship area and the yard using a fleet of vehicles, each of which can carry one container at a time. Without loss of generality, we assume that each vehicle travels at unit speed, i.e., each vehicle travels one unit of distance per unit time, all vehicle travel times between the ship area and a specific location in the yard are deterministic and are known in advance.

To simplify the analysis, we assume, throughout the paper, that there is always an available yard crane ready to respond to a service request of any vehicle. Thus, the time it takes a yard crane to load or unload a container is assumed to be incorporated into the container travel times between the ship area and the yard area. Therefore, throughout, the term *crane* will always refer to a quay crane.

Associated with each "–" ("+") job is a predetermined drop-off (pick-up) point in the yard, called the *location* of the job. Let d_i be the travel time from the ship to job J_i's location; i.e., the drop off location of J_i if it is a "–" job, or the pick up point of job J_i if it is a "+" job. We refer to d_i as *travel time* or *distance* interchangeably.

We first describe the *greedy algorithm*: The first k (=number of vehicles) jobs are assigned, each to a single vehicle. We then assign the next job to the first available vehicle. Specifically, when assigning a "–" job, the first available vehicle that arrives at the quay crane will be dispatched to this job. Similarly, when assigning a "+" job, the first available vehicle that can arrive at the job's location at the earliest time will be dispatched to this job. That is, if a vehicle is currently busy with another job assignment, then the time it can be available at the next "+" job's location will be the time it completes its current assignment plus the traveling time from the destination of its current assignment to the next job's location; if a vehicle is currently free, then this time will simply be the traveling time from its current location to the next job's location. Based on these times, we then select the vehicle that can arrive at the next job's location at the earliest time.

In the following sections, we present our results for different cases of the vehicle dispatching problem.

3.1 Analysis of various dispatching policies

3.1.1 J_- Job sequences

Consider a J_- job sequence. Note that for such a job sequence, once a vehicle takes a "–" job, it has to drop the job to its location in the yard, and then it has to make an empty trip back to the ship area to take its next job. We apply the greedy algorithm defined above to dispatch vehicles to jobs. We have the following result, whose proof is straightforward and is thus omitted.

Theorem 1 *For any J_- job sequence, the greedy algorithm is optimal, that is, the greedy algorithm minimizes the makespan.*

Remark The greedy algorithm is still optimal even if crane processing times are job-specific, that is, the quay crane time associated with job J_i is s_i, and not a constant s. Similarly, it is optimal even when the vehicle travel time and quay crane processing times for each job are random variables.

To illustrate the greedy algorithm, consider an example with four "–" jobs:

$J_- : \{J_1, J_2, J_3, J_4\}$,

with $d_1 = 1$, $d_2 = 5$, $d_3 = 1$, $d_4 = 5$ and $s = 2$ (all in minutes). Let $k = 2$ and in what follows we use V_1 and V_2 to denote the two vehicles. The greedy algorithm works as follows. First, assign V_1 to J_1 and V_2 to J_2. After $s = 2$ minutes, V_1 leaves the crane with J_1, and the crane starts discharging J_2. After another 2 minutes, V_2 leaves with J_2. Now the first available vehicle for jobs J_3 and J_4 is clearly V_1 by times 4 and 8, respectively. The dispatching solution can be represented as $V_1 : \{J_1, J_3, J_4\}$, and $V_2 : \{J_2\}$. The completion time for this J_- sequence is $20 (= 8 + 2 + 10)$ minutes.

3.1.2 J_+ Job sequences

Consider now a J_+ job sequence. For such a job sequence, once a vehicle is assigned to a "+" job, it makes an empty trip to the job's location starting from the ship area, takes the job, and returns back to the ship area with the job. It is easy to see that the greedy algorithm does not necessarily generate an optimal strategy for a J_+ job sequence.

Given a job sequence $J_+ : \{J_1, J_2, \ldots, J_n\}$, consider the following polynomial time algorithm, called the *reversed greedy algorithm*. The reversed greedy algorithm works as follows: Replace each "+" job by a "–" job with the same location, that is, if location p is the pick-up point for a specific "+" job, the associated "–" job has location p as its drop-off point. Now, reverse the order to get the *reversed job sequence* $J_-^R : \{J_n, J_{n-1}, \ldots, J_2, J_1\}$. Apply the greedy algorithm to this reversed list (of "–" jobs), to obtain a set of jobs assigned to each vehicle. For instance, the jobs assigned to vehicle l, $l = 1, 2, \ldots, k$, are given by $V_l : \{J_{l_1}, J_{l_2}, \ldots, J_{l_{f_l-1}}, J_{l_{f_l}}\}$ and they are served by the lth vehicle following that order. The final step of the algorithm is to reverse again the sequence of jobs assigned to each vehicle. That is, vehicle l will serve this set of jobs assigned to it following the order: $\{J_{l_{f_l}}, J_{l_{f_l-1}}, \ldots, J_{l_2}, J_{l_1}\}$.

We have the following result (please see Appendix for the proof):

Theorem 2 *The reversed greedy algorithm is optimal for any J_+ instance.*

Remark Theorem 2 still holds when the crane processing times are job-specific.

Although we have identified the optimal vehicle dispatching rule for uploading job sequences, it is interesting to study how well the simple greedy algorithm, introduced in the previous section, would perform for such job sequences. This is

Table 1. Average percent deviation of the greedy algorithm's makespan from optimality for uploading job sequences

500 Uploading jobs					
Spread a		2	6	10	16
k	4	1.8 %	4.6 %	5.8 %	8.6 %
	5	2.3 %	5.5 %	9.6 %	10.1 %
	6	2.6 %	6.3 %	9.8 %	9.9 %
	7	2.4 %	6.6 %	10.5 %	11.2 %
	8	2.6 %	6.4 %	11.0 %	12.1 %

because the greedy algorithm is an appealing dispatching policy due to its ease of implementation, flexibility, and robustness (i.e., minor disruptions to the schedule can be easily handled by the greedy algorithm, which schedules jobs one at a time, following the order of the job sequence. This, however, is not true for the reversed greedy algorithm, in which small changes to the schedule would require the entire schedule to be regenerated by the reversed greedy.) For this purpose, we have conducted computational experiments, the results of which are presented in the next section.

3.1.3 Computational analysis of the greedy algorithm for J_+ sequences

In this section, we analyze the effectiveness of the greedy algorithm for uploading job sequences. We consider sequences each consisting of 500 "+" jobs. We set the crane discharging/uploading time to be 3 minutes ($s = 3$). The traveling time of each job (between the ship area and its location in the yard area) is generated from a uniform distribution. To determine the impact of job distribution in the yard on the performance of the greedy algorithm, we use four different sets of range (*spread*) in our uniform distribution: in the first set, traveling times are uniformly distributed between 2 and 4 minutes (with a *spread*, a, of 2 minutes), in the second set, between 2 and 8 minutes ($a = 6$), in the third, between 2 and 12 minutes ($a = 10$), and in the fourth, between 2 and 18 minutes ($a = 16$). Thus, as a increases, job locations become more spread apart from each other in the yard area. We replicate each scenario 500 times and determine the percent deviation of the makespan obtained by the greedy algorithm from the optimal makespan (obtained by the reversed greedy algorithm) over the 500 problems. The results are reported in Table 1.

As can be seen from the table, the greedy algorithm generates schedules with a makespan of at most 12% over the optimal makespan. For a fixed number of vehicles, k, the ratio increases with the spread, a. Thus, the gap between the optimal makespan and the makespan of the greedy algorithm increases as jobs get more spread out in the yard area. Similarly, for a fixed value of a, the performance of the greedy algorithm generally deteriorates as the number of vehicles increases.

In practice, terminal dispatchers shelf most "+" jobs that will be loaded onto a particular ship in adjacent clusters in the yard area. Thus, the "spread" for these jobs is usually small. Consequently, we believe that the greedy algorithm would

provide a rather efficient solution to "+" job sequences, and therefore, is a desirable approach due to its simplicity and flexibility.

3.1.4 $J_{-/+}$ job sequences

Consider now a general $J_{-/+}$ job sequence. As mentioned above, if the schedule of a vehicle ends with a "–" job, then the vehicle has to make an empty trip back to the ship area after dropping its last "–" job in the yard area. Similarly, if the schedule of a vehicle begins with a "+" job, then the vehicle has to make an empty trip from the ship area to the yard area to take the "+" job. However, if the schedule of a vehicle is such that the vehicle takes its first "+" job after dropping its last "–" job, then the vehicle saves these two empty trips, and instead, it travels from the location of its last "–" job to the location of its first "+" job. These travel times are sequence dependent, since they depend on the order of the jobs taken by the vehicle.

The optimality of the greedy algorithm for J_- job sequences and the optimality of the reversed greedy algorithm for J_+ job sequences suggest the following algorithm for a $J_{-/+}$ job sequence.

We start with the greedy algorithm applied to the first part of the job sequence, which consists of all the "–" jobs. We then apply the reversed greedy algorithm to the second part of the job sequence which consists of all the "+" jobs. Finally, we combine the two schedules. We refer to this algorithm as the *combined algorithm* (please see Bish et. al [1] for details).

We let Z^C and Z^* respectively denote the makespan obtained by the combined algorithm and the optimal makespan and n denote the number of jobs in the sequence. The next theorem characterizes the effectiveness of the combined algorithm (see Bish et. al [1] for its proof).

Theorem 3 *For every finite instance of a $J_{-/+}$ job sequence, we have*

$$\frac{Z^C}{Z^*} \leq 3.$$

In addition,

$$\lim_{n\to\infty} \frac{Z^C}{Z^*} = 1.$$

The asymptotical performance of the algorithm is especially important, since in practice the number of jobs is in thousands. In addition, in Bish et. al [1] we provide a pseudo-polynomial time algorithm that is optimal for any instance of $J_{-/+}$ job sequences.

In the next section, we use the insights obtained for the single crane model to analyze the vehicle dispatching problem with multiple quay cranes.

4 The vehicle dispatching problem: A multiple crane model

In the previous sections we focused on a single crane model, and showed that the greedy algorithm is optimal (i.e., it minimizes the ship makespan) for a discharging

(J_-) job sequence, and the reversed greedy algorithm is optimal for an uploading (J_+) job sequence. Our next objective is to extend this analysis to the more general case, where multiple quay cranes are assigned to serve a single ship. Associated with each quay crane is a job sequence and the objective is to assign vehicles to containers so as to minimize the time all jobs are done. That is, the objective is to minimize the makespan over all quay cranes. This objective is consistent with a terminal's objective of releasing ships at the earliest possible time.

Thus, the next question is whether the greedy and reversed greedy algorithms continue to be optimal for discharging and uploading job sequences, respectively, when there are multiple cranes. We first focus on situations in which each quay crane has a J_- job sequence. In the multi-crane environment, the greedy algorithm should be interpreted as assigning an available vehicle to the first available ship crane. In this case, however, it is easy to construct examples that demonstrate the greedy algorithm not necessarily to be optimal.

In practice, however, the greedy algorithm is an appealing solution procedure due to its simplicity and flexibility. Therefore, we now use a simulation study to investigate the performance of the greedy algorithm for a multiple crane model with discharging job sequences. Based on this analysis, we, then, refine the greedy algorithm so as to improve its performance for the multiple crane model. Due to the symmetricity between the greedy algorithm and a discharging job sequence, and the reversed greedy algorithm and an uploading job sequence, as observed in the previous section, the performance of the reversed greedy algorithm for a multiple crane model with uploading job sequences will be similar.

In what follows, we first describe the design of our computational experiments and then discuss our findings.

4.1 Design of the computational experiments

Our objective in this section is to evaluate the performance of the greedy algorithm for the multiple crane vehicle dispatching problem with discharging job sequences. For this purpose, we compare the makespan obtained by the greedy algorithm with that of the optimal makespan, obtained by solving a Mixed Integer Program (MIP); the MIP formulation is given in Bish et. al [1]. However, it takes the MIP on the order of a couple of hours on a Sun Sparc 10 workstation to find the optimal solution, even for small sized problems consisting of only 4 vehicles, 2 cranes, and 20 jobs on each crane. Thus, it is not a practical approach for actual dispatching purposes, especially when problems with 500-2500 containers are common in practice.

For this reason, we limit our computational analysis to cases with only 4 vehicles, 2 cranes, each with a job list of $8 - 12$ jobs, and solve 200 such problems. For each job, we generate a traveling time between the ship area and the job's location based on a uniform distribution in the range of 1 to 17 min. We assume that it takes a crane 2 minutes to lift a container from the ship (or the vehicle), and it takes 1 minute to place (pick) the container on (from) the vehicle (observe that letting the first time component to zero reduces the formulation to the model addressed in the previous sections). Thus, a vehicle needs to be available by a crane only during the last minute of job discharging/uploading.

Table 2 summarizes the percent deviation of the makespan obtained by the greedy algorithm from the optimal makespan over the 200 problems: for each range of deviations, we report the number of problem instances with deviation in that range. Table 2 shows that the greedy algorithm performs reasonably well in most cases (in almost 80% of the instances, the deviation from the optimal solution is less than 10%), with an average deviation of 7% from the optimal solution. The next question is whether the performance of the greedy algorithm could further be enhanced by small refinements that are not computationally expensive to implement. This is discussed in the next section.

Table 2. Percent deviation of the heuristic makespan from optimality

% deviation from optimality	$< 1\%$	1-3 %	3-5 %	5-10 %	$> 10\%$
# of instances in this range (out of 200)	3	26	38	88	45
Average deviation $= 7\%$					

4.2 A refined greedy algorithm

Clearly, the main reason for the poor performance of the greedy algorithm is its "myopic" nature. To overcome its "myopic" nature, we propose an enhancement to the greedy algorithm, and include a simple look-ahead rule, described below.

Let $J_{i,j}$, $i = 1, 2$, and $j = 1, 2, ...$, denote the j^{th} job in the sequence of crane i. In what follows, we represent each job in terms of its traveling time (between the ship area and its location). Let l_i be the number of jobs in the job list of crane i. Given a fixed $p \leq l_i$, we assign a weight $w_{i,j} = \sum_{k=j}^{\min\{j+p,l_i\}} J_{i,k}$ to each job $J_{i,j}$, for $j = 1, \ldots, l_i$. Thus, the weight of each job represents the minimum time required to complete the remaining jobs on crane i's list, which excludes crane and queuing times. When a vehicle arrives at the ship area, it determines the job(s) that are available at the earliest time for pick-up (which is determined by the earliest available time of the corresponding crane). If there is only one such job, then it selects that job for pick-up (as in the greedy algorithm). If, on the other hand, there are multiple jobs available at the same time, then the vehicle selects the job with the maximum weight. Thus, in the latter case, the vehicle will give higher priority to the job with a longer traveling time, or to the crane job sequence with a longer time for the remaining jobs.

Finally, we further modify the greedy algorithm by the following *enhancement*: When there are a certain number, x, of jobs left in the system, we perform an explicit enumeration to determine the best schedule for these remaining jobs. Clearly, x should be a very small number. Presumably, this last enhancement is not as effective for reasonably long job sequences. To confirm this, we tested a few examples with 20 jobs. It was found that this refinement "enhanced" efficiency by at most 0.3%

in those examples. (The main reason for us to use such an additional enhancement is to remove the "ending" effect which may arise in small-sized problems.)

Next, we tested the performance of the refined greedy algorithm on the same set of 200 problem instances (with $p = 8, x = 4$). The results show that the refined greedy algorithm generated near-optimal solutions for most instances, with an average deviation from optimality of 1.55%, and a standard deviation of 2.61%. The result is summarized in Table 3, shows the distribution of this deviation for the refined greedy algorithm. As can be seen from the table, the refined greedy algorithm performs much better than the greedy algorithm.

Table 3. Percent deviation of the heuristic makespan from optimality

% deviation from optimality	$< 1\%$	1-3 %	3-5 %	5-10 %	$> 10\%$
# of instances in this range (out of 200)	22	95	40	43	0
Average deviation $= 1.55\%$, standard deviation $= 2.61\%$.					

5 Conclusions and future research directions

Our goal in this research is to come up with simple, easily implementable vehicle dispatching policies that generate *good* makespan values for the vehicle dispatching problem.

The greedy algorithm is an appealing solution due to its simplicity and flexibility. Therefore, in this analysis, we considered the greedy algorithm, together with the reversed greedy algorithm, the combined algorithm and the combined greedy algorithm, all of which are based on the greedy algorithm. By considering a single-ship/single-crane model, we were able to prove the optimality of the greedy algorithm for a discharging job sequence, the optimality of the reversed greedy algorithm for an uploading job sequence, the asymptotic optimality of the combined algorithm together with the optimality of the combined greedy algorithm for a combined job sequence. Based on these results, we, then, analyzed a more general problem of a single ship with multiple cranes, and tested the performance of the greedy algorithm for this problem through computational analysis. The results show that, although not optimal, the greedy algorithm performs reasonably well for a multiple crane vehicle dispatching problem with discharging job sequences. We further enhanced the performance of the greedy algorithm by including a look-ahead type of rule, which we refer to as the refined greedy algorithm. Computational analysis reveals that the performance of the refined greedy algorithm is very satisfactory: an average deviation of 1.55% deviation from the optimal solution over all problems tested.

We must note, however, that this research is only a start to analyze the operational issues in container terminals, and there are still many open issues that need to be analyzed.

In practice, other issues need to be incorporated into the analysis and addressed by the algorithms. One important issue is how to determine a storage location for each discharged container. In the model considered here, the storage location of each discharged container is assumed to be given. This problem, where the location of each discharged container is also a decision variable, has been analyzed in [3] and [4]. Another issue would be identifying routes for each vehicle so as to avoid congestion. There is also the issue of coordinating yard crane work load, etc. Yet another important research direction would be to extend this analysis to a multiple ship model. This direction has been studied in several recent papers; see, for instance, Bish [2] and Kim and Bae [9]. Bish [2] considers the vehicle dispatching and container location problem for a multi-ship multi-crane model, develops a heuristic algorithm, which assigns locations to containers based on a transshipment problem and dispatches vehicles to jobs based on a modified version of the greedy algorithm, and analyzes the effectiveness of the heuristic from both worst-case and computational points of view. Her results suggest that a modified version of the greedy algorithm works very well in a multi-ship setting as well. However, analytical results are presented only for a two-ship model and need to be extended to consider any number of ships. On the other hand, Kim and Bae [9] develop a mathematical programming formulation for a multi-ship multi-crane model, suggest a heuristic algorithm, and analyze its performance through a numerical study. We believe that this line of work needs to be extended to analytically characterize the effectiveness of simple heuristics, such as modified versions of the greedy algorithm discussed in this paper, in the context of a multi-ship model.

Although we considered a simplified model in this research, the insights gained in this paper proved to be helpful in analyzing more complex situations at terminal ports.

Appendix: Proof of Theorem 2

Consider any job sequence consisting of jobs $\{J_1, J_2, \cdots, J_n\}$. For dispatching policy π, we refer to the time a vehicle is assigned to J_i, $i = 1, 2, \cdots, n$, as the *start time* of J_i, and denote it as $ST_i(\pi)$. Similarly, we refer to the time J_i, $i = 1, 2, \cdots, n$, is completed under that policy as the *completion time* of J_i, and denote it as $CT_i(\pi)$.

Thus, in a J_- job sequence, the start time of a "–" job is the time the crane starts the task of discharging the job to a vehicle, and the completion time of a "–" job is the time the vehicle returns to the ship area after carrying the discharged job to its location in the yard. In a J_+ job sequence, the start time of a "+" job is the time a vehicle is dispatched to the job's location to bring the job to the ship, and the completion time of a "+" job is the time the quay crane finishes loading the job onto the ship. We will omit the policy parameter and use ST_i and CT_i, when the policy is obvious from the context or when a specific property must hold for all policies.

As stated before, job precedence constraints for a $J_- : \{J_1, J_2, \cdots, J_n\}$ job sequence imply that

$$ST_i \geq ST_{i-1} + s \quad i = 2, \cdots, n,$$

whereas, for a $J_+ : \{J_1, J_2, \cdots, J_n\}$ job sequence we must have

$$CT_i \geq CT_{i-1} + s \quad i = 2, \cdots, n.$$

To prove the Theorem, we need the following lemma.

Lemma 4 *Consider a dispatching policy π_+ applied to a "+" job sequence J_+, with a makespan of $Z(\pi_+)$. There exists a dispatching policy π_- applied to the reversed job sequence J_-^R associated with J_+ that achieves the same makespan, i.e., $Z(\pi_-) = Z(\pi_+)$.*

Proof. Consider a "+" job sequence $J_+ : \{J_1, J_2, \cdots, J_n\}$, and a dispatching policy π_+. We let $V_l : \{J_{l_1}, J_{l_2}, \cdots, J_{l_{f_l}}\}$ denote the job sequence assigned to vehicle $l, l = 1, 2, \cdots, k$, under this dispatching policy. The precedence constraints for this J_+ job sequence imply that job i, $i = 2, \cdots, n$, cannot be completed until all its predecessors in J_+, i.e., jobs $J_1, J_2, \cdots, J_{i-1}$, are completed. Hence, we have

$$CT_i(\pi_+) \geq CT_{i-1}(\pi_+) + s \quad i = 2, \cdots, n. \tag{1}$$

Clearly, the makespan for this dispatching policy is $Z(\pi_+) = CT_n(\pi_+)$.

Now consider the corresponding reversed "–" job sequence, $J_-^R : \{J_n, J_{n-1}, \cdots, J_2, J_1\}$. Our objective is to find a dispatching policy π_- for the reversed job sequence with a makespan of $Z(\pi_-)$ such that $Z(\pi_-) = Z(\pi_+)$.

For this purpose, consider the dispatching policy π_- obtained as follows. Reverse the job sequence $V_l, l = 1, 2, \cdots, k$, defined as above, and denote the resulting sequence as $V_l^R : \{J_{l_{f_l}}, J_{l_{f_l-1}}, \cdots, J_{l_1}\}$. Under this policy, vehicle l starts with job $J_{l_{f_l}}$, continues with job $J_{l_{f_l}-1}$, and so on. Start job J_n, a "–" job now, at time $Z(\pi_+) - CT_n(\pi_+) = 0$, job J_{n-1} at $Z(\pi_+) - CT_{n-1}(\pi_+)$, $\cdots$, and job J_1 at $Z(\pi_+) - CT_1(\pi_+)$.

Now, if we can show that the schedule obtained by dispatching policy π_- satisfies (i) the precedence constraints for the J_-^R job sequence, and (ii) vehicle capacity constraints, then we have a dispatching policy for which

$$Z(\pi_-) = Z(\pi_+) - CT_1(\pi_+) + 2d_1 + s = Z(\pi_+)$$

and we are done.

Consider jobs J_{i-1} and J_i, $i = 2, \cdots, n$. Under dispatching policy π_-, we have:

$$\begin{aligned} ST_{i-1}(\pi_-) - ST_i(\pi_-) &= Z(\pi_+) - CT_{i-1}(\pi_+) - Z(\pi_+) + CT_i(\pi_+) \\ &= CT_i(\pi_+) - CT_{i-1}(\pi_+) \geq s, \quad \text{from Equation (1)} \end{aligned}$$

and hence the precedence constraints for the J_-^R job sequence are satisfied.

Next, we have to show that the schedule obtained by dispatching policy π_- also satisfies the vehicle capacity constraints. Vehicle $l, l = 1, 2, \cdots, k$, can serve jobs

$\{J_{l_{f_l}}, J_{l_{f_l}-1}, \cdots, J_{l_1}\}$ under dispatching policy π_-, since for jobs J_{l_h-1} and J_{l_h}, $h = 2, \cdots, f_l$, we have the following:

$$\begin{aligned} ST_{l_h-1}(\pi_-) - ST_{l_h}(\pi_-) &= Z(\pi_+) - CT_{l_h-1}(\pi_+) - Z(\pi_+) + CT_{l_h}(\pi_+) \\ &= CT_{l_h}(\pi_+) - CT_{l_h-1}(\pi_+) \geq 2d_{l_h} + s, \end{aligned}$$

indicating that vehicle capacity constraints are satisfied. Hence, dispatching policy π_- generates a feasible schedule for the J_-^R job sequence with a makespan of $Z(\pi_+)$. This completes the proof. □

Lemma 4 leads to the following corollary:

Corollary 5 *Consider a dispatching policy π_- applied to a job sequence J_-, with a makespan of $Z(\pi_-)$. There exists a dispatching policy π_+ for the reversed "+" job sequence associated with J_- such that it achieves exactly the same makespan, i.e., $Z(\pi_+) = Z(\pi_-)$.*

Now we are ready to prove Theorem 2.

Proof of Theorem 2. Let π_+^* be the optimal dispatching policy for a J_+ job sequence, with a makespan of $Z(\pi_+^*)$. By Lemma 4, we can find a dispatching policy π_- for the corresponding reversed "–" job sequence J_-^R such that $Z(\pi_-) = Z(\pi_+^*)$.

Now consider the optimal dispatching policy π_-^* for this J_-^R job sequence with a makespan of $Z(\pi_-^*)$. By Corollary 5, we can find a dispatching policy π_+ for the corresponding reversed "+" job sequence J_+, which is the original J_+ job sequence, such that $Z(\pi_-^*) = Z(\pi_+)$.

By the optimality of $Z(\pi_-^*)$ for the J_-^R job sequence, we have:

$$Z(\pi_+^*) = Z(\pi_-) \geq Z(\pi_-^*) = Z(\pi_+) \tag{2}$$

On the other hand, the optimality of $Z(\pi_+^*)$ for the J_+ job sequence implies that

$$Z(\pi_+^*) \leq Z(\pi_+),$$

and hence, Equation (2) holds as equality. That is,

$$Z(\pi_+^*) = Z(\pi_-^*) \tag{3}$$

Finally, Theorem 1 tells us that the greedy algorithm is an optimal dispatching policy for the J_-^R job sequence. Thus, given a J_+ job sequence, we can obtain the reversed job sequence J_-^R associated with J_+ and find the optimal schedule for this J_-^R job sequence by applying the greedy algorithm. Given the optimal schedule for the J_-^R job sequence, we can construct a schedule for the original J_+ job sequence as done in the proof of Lemma 4. That is, we can do that by reversing the sequence of jobs assigned to each vehicle. Furthermore, this must be (one of) the optimal schedule(s) for the J_+ job sequence, since $Z(\pi_-^*) = Z(\pi_+^*)$ by Equation(3). Observing that this is the reversed greedy algorithm completes the proof. □

References

1. Bish EK, Chen FY, Leong YT, Nelson BL, Ng JW, Simchi-Levi D (2000) Dispatching vehicles in a mega container terminal. Unabridged Technical Report, Virginia Polytechnic Institute and State University, Dept of Industrial and Systems Engg
2. Bish EK (2003) A multiple-crane-constrained scheduling problem in a container terminal. European Journal of Operational Research 144: 83–107
3. Bish EK, Leong T, Li C, Ng JWC, Simchi-Levi D (2001) Analysis of a new scheduling and location problem. Naval Research Logistics 48: 363–385
4. Bish EK (1999) Theoretical analysis and practical algorithms for problems in a mega container terminal. Ph.D. Dissertation, Northwestern University
5. Bramel J, Simchi-Levi D (1997) The logic of logistics. Theory, algorithms, and applications for logistics management. Springer, New York, NY
6. Castilho BD, Daganzo CF (1993) Handling strategies for import containers at marine terminals. Transportation Research 27B(2): 151–166
7. Daganzo CF (1990) The productivity of multipurpose seaport terminals. Transportation Science 24: 205–216
8. Grunow M, Günther HO, Lehmann M (2004) Dispatching multi-load AGVs in highly automated seaport container terminals. OR Spectrum 26: 211–235
9. Kim KH, Bae JW (2004) A look-ahead dispatching method for automated guided vehicles in automated port container terminals. Transportation Science 38: 224–234
10. Kim KH, Kang JS, Ryu K-R (2004) A beam search algorithm for the load sequencing of outbound containers in port container terminals. OR Spectrum 26: 93–116
11. Kim KH, Park YM, Ryu K-R (2000) Deriving decision rules to locate export containers in container yards. European Journal of Operational Research 124: 89–101
12. Kim KH, Kim HB (1999) Segregating space allocation models for container inventories in port container terminals. International Journal of Production Economics 59: 415–423
13. Kim KH, Kim KY (1999) An optimal routing algorithm for a transfer crane in port container terminals. Transportation Science 33(1): 173–176
14. Kim KH (1997) Evaluation of the number of rehandles in container yards. Computers and Industrial Engineering 32(4): 701–711
15. Kim KY, Kim KH (1999) A routing algorithm for a single straddle carrier to load export containers onto a containership. International Journal of Production Economics 59: 425–433
16. Steenken D, Voss S, Stahlbock R (2004) Container terminal operation and operations research – A classification and literature review. OR Spectrum 26: 3–49
17. Vis IFA, Harika I (2004) Comparison of vehicle types at an automated container terminal. OR Spectrum 26: 117–143
18. Vis IFA, De Koster R, Savelsbergh MWP (2004) Minimum vehicle fleet size under time window constraints at a container terminal. Transportation Science (forthcoming)
19. Vis IFA, De Koster R (2003) Transshipment of containers at a container terminal: an overview. European Journal of Operational Research 147: 1–16
20. Yang CH, Choi YS, Ha TY (2004) Simulation-based performance evaluation of transport vehicles at automated container terminals. OR Spectrum 26: 149–170