Loading and Unloading Operations in Container Terminals

Chung-Lun Li* George L. Vairaktarakis†

July 2001

Revised May 2002

Revised January 2003

Revised April 2003

Abstract

We consider the problem of optimizing the time for loading and unloading containers to and from a ship at a container terminal, where containers are required to be transported by trucks between the ship and their designated locations in the container yard. An optimal algorithm and some efficient heuristics are developed to solve the problem with a single quay crane. The effectiveness of the heuristics is studied both analytically and computationally.

*Department of Shipping and Transport Logistics, The Hong Kong Polytechnic University, Hung Hom, Kowloon, Hong Kong. Email: *msclli@polyu.edu.hk*

†Weatherhead School of Management, Department of Operations, Case Western Reserve University, 10900 Euclid Avenue, Cleveland, OH 44106-7235. Email: *gxv5@weatherhead.cwru.edu*

1 Introduction

We consider the problem of optimizing the time for loading and unloading containers to and from a ship at a container terminal, where containers are required to be transported by trucks between the ship and their designated locations in the container yard. Generally speaking, a container terminal consists of the quayside, where containers are unloaded from and loaded onto ships, and the container storage yard, where containers are stored. External trucks are responsible for taking containers in and out of a container terminal. Internal trucks are responsible for transporting containers within the terminal, mainly between the container yard and the quayside. When a ship arrives at the terminal, containers are first unloaded from the ship onto the internal trucks by quay cranes. Next, the internal trucks transport the containers to their prespecified locations in the container yard. Upon arrival at the container block, a container is unloaded from the truck by a yard crane. After most of the containers have been unloaded from the ship, the outbound containers are transported from the storage yard to the quayside by those internal trucks and are loaded onto the ship by the quay cranes. Typically, several quay cranes can work in parallel to serve a ship, where each crane is designated to handle one section of the ship. To maximize the efficiency of this operation, it is important to have the turnaround time of the ship minimized. The speed of this loading/unloading operation is limited by the availability of the quay cranes and trucks. A quay crane can handle only one container at a time, and a truck can normally transport one container at a time as well. The yard cranes, on the other hand, have more capacity, and they usually spend their spare time shuffling the containers within a container block so as to reduce the loading/unloading time of the future tasks.

Research related to the efficiency of container terminal operations has appeared in the literature. This includes the use of queuing models to study the utilization of container terminal capacity, see for example, Daganzo [9], Easa [11], and Gransberg and Basilotto [12]. It also includes the studies of operational issues such as berth allocation (Brown *et al.* [6], Li *et al.* [16], and Lim [17]), crane movements (Daganzo [8], Kim and Kim [13], and Peterkofsky and Daganzo [18]), storage allocation (Bish [2], Bish *et al.* [4], de Castilho and Daganzo [10], and Kim *et al.* [14]), and vehicle dispatching

(Bostel and Dejax [5] and Powell and Carvalho [19]).

Recently, Bish *et al.* [3] considered a vehicle dispatching problem in a container terminal. In their model, containers are first unloaded from a ship by quay cranes following a prespecified sequence (known as "crane job sequence"). The unloaded containers are transported to their destinations in the container yard by a fleet of identical vehicles, each of which can carry one container at a time. Another set of containers are then transported from the yard by the same set of vehicles and are loaded onto the ship, also following a prespecified sequence. It is assumed that the travel times of vehicles satisfy the triangle inequality. The time that a quay crane needs to load/unload a container is assumed to be the same for all the jobs. The objective is to determine a vehicle schedule so as to minimize the makespan, i.e., the time it takes to complete the entire loading/unloading operation for the ship. They developed optimal and heuristic solution procedures for solving this problem. They performed worst case analysis as well as computational analysis on their heuristic method for the single quay crane case of the problem. Their computational analysis is further extended to the multiple quay crane case.

In this paper, we build on the work of Bish *et al.* [3]. We relax the assumption that the loading and unloading time of container is the same for all the jobs. In the presence of a single crane, we present an optimal algorithm which is efficient for small problems. For large problems, we provide a lower bound procedure as well as heuristics with favorable worst-case error bound and fast asymptotic convergence. Our analyses for the single crane case are supported by extensive computational experiments on randomly generated problems.

In the next section, some basic results of the single crane problem are presented. An optimal algorithm is provided in Section 3, and a procedure for determining a lower bound of the optimal solution value is developed in Section 4. Several heuristics are developed and analyzed in Section 5, and computational results are reported in Section 6 followed by some concluding remarks in Section 7.

2 Notation and Basic Results

Throughout the analysis, each container is referred to as a "job". The vehicle dispatching problem with a single quay crane can be formally described as follows: We would like to minimize the time required to execute a given set of unloading jobs in a predetermined sequence, followed by a given set of loading jobs in a predetermined sequence. The execution time of every job is deterministically known and consists of two components — the time required for the quay crane to pick/drop a container from/onto a truck (during this time both the crane and the truck are occupied), plus the time required to transport the container between the crane and the appropriate yard location. Upon completing the last unloading job assigned to a truck, the truck transfers directly to the first loading job assigned to it (if any) without passing through the crane. All trucks are assumed to be located next to the quay crane at time 0 and are required to return to the quay crane at the end of all operations. Since the loading and unloading sequences of the jobs are given, our only decision is the assignment of the loading and unloading jobs to trucks so as to minimize the makespan of the schedule.

The following notation will be used:

m = number of trucks;

n_1 = number of unloading jobs;

n_2 = number of loading jobs;

$U = \{J_1^U, J_2^U, \ldots, J_{n_1}^U\}$ = set of unloading jobs, ordered in the required processing sequence;

$L = \{J_1^L, J_2^L, \ldots, J_{n_2}^L\}$ = set of loading jobs, ordered in the required processing sequence;

$s(J)$ = processing requirement of the quay crane for job $J \in U \cup L$;

$t_0(J)$ = one-way travel time required by any truck to transport a job $J \in U \cup L$ between the crane and the appropriate yard location;

$t(J, J')$ = time for any truck to travel between the yard locations associated with jobs $J \in U$ and $J' \in L$ (note that the triangle inequality implies that $t(J, J') \leq t_0(J) + t_0(J')$).

$Z(\sigma)$ = duration (i.e., makespan) of a feasible unloading/loading schedule σ;

We assume that all problem parameters (including $s(J)$, $t_0(J)$, and $t(J, J')$) are nonnegative integers. We first describe two job assignment rules. One rule is for the unloading jobs, and the other is for the loading jobs.

Definition 1 *First Available Truck (FAT) rule: Assign the first unscheduled job in U to the available truck which first completes the jobs previously assigned to it.*

Definition 2 *Last Busy Truck (LBT) rule: Select any value τ. Schedule jobs to trucks so that the last job completes at time τ, and proceed backwards. Assign the last unscheduled job in L to the last available truck that became busy with jobs previously assigned to it.*

For example, if there are 2 trucks and 4 unloading jobs with $s(J_1^U) = s(J_2^U) = s(J_3^U) = s(J_4^U) = 3$, $t_0(J_1^U) = 5$, $t_0(J_2^U) = 4$, $t_0(J_3^U) = 2$, and $t_0(J_4^U) = 3$, then the schedule generated by the FAT rule is depicted in Figure 1(a). Note that the round-trip travel time for each unloading job J is $2t_0(J)$, and the makespan of this schedule is 25. Now, consider another example with 2 trucks, 4 loading jobs, $s(J_1^L) = s(J_2^L) = s(J_3^L) = s(J_4^L) = 3$, $t_0(J_1^L) = 5$, $t_0(J_2^L) = 4$, $t_0(J_3^L) = 2$, and $t_0(J_4^L) = 3$, then the schedule generated by the LBT rule is shown in Figure 1(b). In this schedule, if we set $\tau = 25$, then the quay crane will start working at time 0 and the makespan of the schedule is 25.

Note that the FAT rule is the same as the *list scheduling* rule in the machine scheduling literature (see, for example, Lawler *et al.* [15]). Note also that the FAT and LBT rules have appeared in Bish *et al.* [3] as the "greedy algorithm" and "reversed greedy algorithm", respectively. Bish *et al.* have shown that if $L = \emptyset$ (i.e., there are only unloading jobs), then the FAT rule will generate an optimal schedule. We denote this optimal schedule as σ^{FAT}. By symmetry, if $U = \emptyset$ (i.e., there are only loading jobs), then the LBT rule will generate an optimal schedule (see [3]) which we denote as σ^{LBT}.

Consider a subproblem in which we would like to schedule only the unloading jobs U, where subset $U_0 = \{J_{\lambda_1}^U, J_{\lambda_2}^U, \ldots, J_{\lambda_m}^U\}$ includes all unloading jobs that are served last by a truck. We let

$\sigma^U(U_0)$ denote the optimal schedule for this subproblem. Another subproblem is to schedule only the loading jobs L, where subset $L_0 = \{J^L_{\mu_1}, J^L_{\mu_2}, \ldots, J^L_{\mu_m}\}$ includes all loading jobs served first by a truck. We let $\sigma^L(L_0)$ denote the optimal schedule for this subproblem. The schedules $\sigma^U(U_0)$ and $\sigma^L(L_0)$ become "partial schedules" of the original problem.

Lemma 1 *(Bish et al. [3]) Suppose that $L = \emptyset$. Then the FAT rule will generate a schedule that minimizes the time required to execute all unloading jobs. Moreover, if $J^U_{\lambda_i}$ is required to be served last on a truck and $\lambda_1 < \lambda_2 < \cdots < \lambda_m$, then for every $i = 1, 2, \ldots, m$, the FAT rule will generate a schedule that minimizes the time required to unload jobs $J^U_{\lambda_1}, J^U_{\lambda_2}, \ldots, J^U_{\lambda_i}$ given that each of these jobs is served last on a truck.*

Lemma 2 *(Bish et al. [3]) Suppose that $U = \emptyset$. Then the LBT rule will generate a schedule that minimizes the time required to execute all loading jobs. Moreover, if $J^L_{\mu_i}$ is required to be the first job served by a truck and $\mu_1 < \mu_2 < \cdots < \mu_m$, then for every $i = 1, 2, \ldots, m$, the LBT rule will generate a schedule that minimizes the time that elapses between the start and finish times of trucks that serve first one of the jobs $J^L_{\mu_1}, J^L_{\mu_2}, \ldots, J^L_{\mu_i}$.*

Bish *et al.*'s lemmas assume that the processing requirement of the quay crane, $s(J)$, is constant. However, the proofs remain valid even when the quay crane processing times are job dependent.

3 An Optimal Algorithm

Bish *et al.* [3] have developed an optimal algorithm for solving the single crane vehicle dispatching problem. Their algorithm has a running time of $O(n^{2m-1} \cdot m!)$, where $n = n_1 + n_2 + 2m - 2$. In this section, we develop an optimal algorithm with a lower computational complexity.

Given a set of final unloading jobs U_0 and a set of leading loading jobs L_0 for the m trucks, Lemmas 1 and 2 state that the FAT and LBT rules provide optimal solutions to the unloading and loading subproblems, respectively. Therefore, one can enumerate exhaustively all possible pairs U_0, L_0 of final unloading and leading loading job subsets (see [3]). Two issues remain to

be considered: calculating the number of possible U_0, L_0 pairs, and finding an optimal way to concatenate the partial schedules $\sigma^U(U_0)$ and $\sigma^L(L_0)$.

Note that it may be optimal to assign to a truck only loading (unloading) jobs, for example, when a container is located very far from the quay crane. In case a truck is assigned to process loading (unloading) jobs only, then one could introduce in U (L) a dummy unloading (loading) job with zero crane processing requirement and zero distance from the quay crane. Also, we observe that $J^U_{n_1} \in U_0$ and $J^L_1 \in L_0$ due to the precedence requirements for the jobs. Hence, if we introduce $m-1$ dummy jobs before J^U_1 in U, and $m-1$ dummy jobs following $J^L_{n_2}$ in L, then we only need to consider solutions where at least one loading job and at least one unloading job are served by each truck. Therefore, we may assume that U consists of $n_1 + m - 1$ jobs and that L consists of $n_2 + m - 1$ jobs. With this assumption, the number of possible final unloading job subsets, U_0, is $\binom{n_1+m-2}{m-1} \leq n_1^{m-1} \leq O(n^{m-1})$, where $n = n_1 + n_2 + 2m - 2$. Similarly, the number of possible leading loading job subsets, L_0, is $\binom{n_2+m-2}{m-1} \leq n_2^{m-1} \leq O(n^{m-1})$. Thus, the number of possible U_0, L_0 pairs is $O(n^{2m-2})$.

For a given set U_0 of final unloading jobs, Lemma 1 states that the FAT rule produces an optimal partition of jobs in U into m parts. Similarly, for a given set L_0 of leading loading jobs, Lemma 2 states that the LBT rule produces an optimal partition of jobs in L into m parts. Then, one has to match the m parts of U with the m parts of L, and then assign each pair of parts to a truck so as to minimize the time required to complete all jobs. This can be done using the following "bottleneck assignment model" (see Ahuja *et al.* [1], p. 505) for the final unloading jobs $U_0 = \{J^U_{\lambda_1}, J^U_{\lambda_2}, \ldots, J^U_{\lambda_m}\}$ and leading loading jobs $L_0 = \{J^L_{\mu_1}, J^L_{\mu_2}, \ldots, J^L_{\mu_m}\}$. Let v_i $(i = 1, \ldots, m)$ denote the truck that gets assigned $J^U_{\lambda_i}$ as the final unloading job when the FAT rule is applied to set U_0 and v'_j $(j = 1, \ldots, m)$ denote the truck that gets assigned $J^L_{\mu_j}$ as the leading loading job when the LBT rule is applied to set L_0. Let T_i $(i = 1, \ldots, m)$ denote the total time it takes truck v_i to finish serving all its unloading jobs, including the time to transport the last job $J^U_{\lambda_i}$ to its destination, but excluding the time to travel back to the quay crane afterward. Let T'_j $(j = 1, \ldots, m)$ denote the total time it takes truck v'_j to finish serving all its loading jobs, including the time to transport

the first job $J^L_{\mu_j}$ from its location, but excluding the time to travel to that location beforehand. Let c^U denote the time that the crane finishes unloading job $J^U_{n_1}$ in schedule $\sigma^U(U_0)$. Let c^L be the elapsed time between the moment when the crane starts loading job J^L_1 and the completion time of all jobs in schedule $\sigma^L(L_0)$. Let $c_{\min} = c^U + c^L$. Since we have to finish unloading all jobs before we can perform the loading of jobs, $c_{\min}$ represents a minimum possible makespan of the schedule. Let

$$c_{ij} = \max\{T_i + t(J^U_{\lambda_i}, J^L_{\mu_j}) + T'_j,\ c_{\min}\}.$$

Here, $T_i + t(J^U_{\lambda_i}, J^L_{\mu_j}) + T'_j$ represents the time it takes a truck to serve all the unloading jobs assigned to truck v_i, travel to pick up $J^L_{\mu_j}$ directly, and serve all the loading jobs assigned to truck v'_j. The bottleneck assignment problem is to match the m parts of U with the m parts of L so as to minimize $\max_{i,j=1,\ldots,m}\{c_{ij}\}$. This bottleneck value minimizes the maximum time that any truck or the quay crane remains busy, and hence it equals the minimal makespan value for the pair U_0, L_0.

For each possible pair of U_0, L_0, solving the bottleneck assignment problem takes $O(m^{2.5}\log m)$ time (see [1]), and determining the schedule corresponding to a bottleneck matching using the FAT and LBT rules requires $O(n)$ time. Hence, the computational time required to evaluate each pair of U_0, L_0 is $O(n + m^{2.5}\log m)$. This implies that the overall complexity of this solution method is $O(n^{2m-2}(n+m^{2.5}\log m))$. This complexity is an improvement of the one provided by Bish *et al.* [3], which is $O(n^{2m-1}\cdot m!)$.

The above analysis suggests that when m is small, our problem can be solved efficiently. However, it remains an open question of whether the problem is polynomial time solvable. Evidently, for problems with many trucks and many jobs, efficient heuristics for the problem are desirable. We develop such heuristics in Section 5. To evaluate these heuristics, we need a tight lower bound. This is the focus of the next section.

4 Lower Bounds

In the following analysis, we assume that U consists of n_1 original unloading jobs plus $m-1$ dummy unloading jobs. Thus, we redefine

$$U = \{J_1^U, J_2^U, \ldots, J_{n_1+m-1}^U\}$$

as the set of unloading jobs, where the jobs are ordered in the required processing sequence and $J_1^U, J_2^U, \ldots, J_{m-1}^U$ are dummy jobs. Similarly, L consists of n_2 original loading jobs plus $m-1$ dummy loading jobs. Thus, we redefine

$$L = \{J_1^L, J_2^L, \ldots, J_{n_2+m-1}^L\}$$

as the set of loading jobs, where the jobs are ordered in the required processing sequence and $J_{n_2+1}^L, J_{n_2+2}^L, \ldots, J_{n_2+m-1}^L$ are dummy jobs. If J is a dummy unloading job and J' is a dummy loading job, then let us define $s(J) = s(J') = t_0(J) = t_0(J') = t(J, J') = 0$. If J is a dummy unloading job and J' is a non-dummy loading job, then we define $t(J, J') = t_0(J')$. Similarly, if J is a non-dummy unloading job and J' is a dummy loading job, then $t(J, J') = t_0(J)$.

Given a set of final unloading jobs $U_0 = \{J_{\lambda_1}^U, J_{\lambda_2}^U, \ldots, J_{\lambda_m}^U\}$ and a set of leading loading jobs $L_0 = \{J_{\mu_1}^L, J_{\mu_2}^L, \ldots, J_{\mu_m}^L\}$ for the m trucks, the bottleneck assignment algorithm proposed in Section 3 produces a schedule with minimal makespan. Consider any feasible solution to this bottleneck assignment problem, and let $J_{\alpha_i}^U$ and $J_{\beta_i}^L$ be the final unloading job and leading loading job, respectively, assigned to truck i, for $i = 1, \ldots, m$. Then, the job pairs $(J_{\alpha_1}^U, J_{\beta_1}^L), (J_{\alpha_2}^U, J_{\beta_2}^L), \ldots, (J_{\alpha_m}^U, J_{\beta_m}^L)$ yield a bipartite matching, say M, of cardinality m among the jobs in U_0 and L_0. Let U_0^*, L_0^* denote the pair of final unloading and leading loading subsets of jobs in an optimal schedule σ^*, and let M^* denote the corresponding optimal bipartite matching. In what follows, we will use M^* to derive a lower bound of $Z(\sigma^*)$.

Lemma 3 *Let*

$$LB_1(M) = \frac{1}{m}\Big\{ \sum_{i=1}^{n_1+m-1} \Big[s(J_i^U) + 2t_0(J_i^U)\Big] + \sum_{i=1}^{n_2+m-1} \Big[s(J_i^L) + 2t_0(J_i^L)\Big] - \sum_{(J_i^U, J_j^L)\in M} \pi_{ij}\Big\},$$

where $\pi_{ij} = t_0(J_i^U) + t_0(J_j^L) - t(J_i^U, J_j^L)$. *Then* $Z(\sigma^*) \geq LB_1(M^*)$.

Proof: Observe that the quantity $\sum_{i=1}^{n_1+m-1}\left[s(J_i^U)+2t_0(J_i^U)\right]+\sum_{i=1}^{n_2+m-1}\left[s(J_i^L)+2t_0(J_i^L)\right]-\sum_{(J_i^U,J_j^L)\in M^*}\pi_{ij}$ is the total time spent by the m trucks in the optimal schedule σ^* to unload the jobs in U_0^*, move from the jobs in U_0^* to the jobs in L_0^*, and then load the jobs in L_0^*, excluding any idle time between jobs. If this total time was equally divided amongst m trucks, then each truck would require $\frac{1}{m}$ of the total. This implies that the makespan of any schedule must be at least $LB_1(M^*)$. ∎

Lemma 4 *Let $\hat{T}_i$ be the completion time of J_i^U in schedule σ^{FAT}, including the time to travel back to the quay crane afterward. Let $\hat{T}'_j$ be the elapsed time to complete all loading jobs in σ^{LBT} starting from J_j^L, including the time to travel from the quay crane to the yard location of J_j^L. Let*

$$LB_2(M)=\max_{(J_i^U,J_j^L)\in M}\{\hat{T}_i+\hat{T}'_j-\pi_{ij}\},$$

where $\pi_{ij}=t_0(J_i^U)+t_0(J_j^L)-t(J_i^U,J_j^L)$. Then $Z(\sigma^*)\geq LB_2(M^*)$.

Proof: We consider any pair (J_i^U,J_j^L) of final unloading and leading loading jobs in an optimal schedule σ^*. We know that $\hat{T}_i$ is the shortest possible completion time for J_i^U among all feasible schedules (by Lemma 1). Similarly, $\hat{T}_j$ is the shortest elapsed time to complete all loading jobs when starting with J_j^L (by Lemma 2). Recall that in σ^{FAT} and σ^{LBT}, a two-way travel is included for every job. Hence, $\hat{T}_i$ and $\hat{T}'_j$ account for round trips for all jobs. The value $-\pi_{ij}$ adjusts the travel time consumed by a truck to move between the locations of J_i^U and J_j^L without visiting the quay crane in-between. Therefore, the quantity $\hat{T}_i+\hat{T}'_j-\pi_{ij}$ is a lower bound between the start time of J_1^U and the completion of $J_{n_2}^L$, and this is true for every pair $(J_i^U,J_j^L)\in M^*$. Hence, the maximum of these quantities, $\max_{(J_i^U,J_j^L)\in M^*}\{\hat{T}_i+\hat{T}'_j-\pi_{ij}\}$, is a lower bound of $Z(\sigma^*)$. ∎

Let

$$\mathcal{M}=\Big\{M\;\Big|\;M\text{ is a bipartite matching of cardinality }m\text{ among the jobs in }U_0\text{ and }L_0,\\ \text{where }J_{n_1+m-1}^U\in U_0,\ J_1^L\in L_0,\ U_0\subseteq U,\ L_0\subseteq L,\text{ and }|U_0|=|L_0|=m\Big\}.$$

The set $\mathcal{M}$ represents a collection of all possible bipartite matchings of all feasible combinations of final unloading job subset U_0 and leading loading job subset L_0.

Theorem 5 *Let* $Z_{LB} = \min_{M \in \mathcal{M}} \Big\{ \max\{LB_1(M), LB_2(M)\} \Big\}$. *Then* $Z(\sigma^*) \geq Z_{LB}$.

Proof: By Lemmas 3 and 4,

$$Z(\sigma^*) \geq \max\{LB_1(M^*), LB_2(M^*)\}. \tag{1}$$

Since $M^* \in \mathcal{M}$, we have

$$\min_{M \in \mathcal{M}} \Big\{ \max\{LB_1(M), LB_2(M)\} \Big\} \leq \max\{LB_1(M^*), LB_2(M^*)\}. \tag{2}$$

Combining (1) and (2) yields the desired result. ∎

Theorem 5 provides us with a lower bound Z_{LB} of $Z(\sigma^*)$. However, to calculate this lower bound, we need to evaluate $LB_1(M)$ and $LB_2(M)$ for all possible $M \in \mathcal{M}$, where the size of set $\mathcal{M}$ is quite large. In what follows, we develop an efficient method to compute this lower bound.

Note that Z_{LB} is equal to the optimal solution value of the following mathematical program:

$$\begin{array}{ll} \text{Minimize} & Z \\ \text{subject to} & Z \geq \dfrac{1}{m}\Big(D - \displaystyle\sum_{(J_i^U, J_j^L) \in M} \pi_{ij}\Big) \\ & Z \geq \hat{T}_i + \hat{T}'_j - \pi_{ij}, \quad \forall\ (J_i^U, J_j^L) \in M \\ & M \in \mathcal{M}, \end{array}$$

where $D = \sum_{i=1}^{n_1+m-1} \Big[s(J_i^U) + 2t_0(J_i^U)\Big] + \sum_{i=1}^{n_2+m-1} \Big[s(J_i^L) + 2t_0(J_i^L)\Big]$. To solve this mathematical program, we may use bisection search on all possible values of Z. For a given value of Z, we need to determine whether the above mathematical program is feasible. In other words, for a given value of Z, we would like to determine whether there exists bipartite matching $M \in \mathcal{M}$ such that

$$\pi_{ij} \geq \hat{T}_i + \hat{T}'_j - Z$$

for every $(J_i^U, J_j^L) \in M$ and

$$\sum_{(J_i^U, J_j^L) \in M} (-\pi_{ij}) \leq mZ - D.$$

This can be solved as a minimum cost network flow problem described as follows. We construct a network with a source s and a sink t. The underlying directed graph is $G = (V, A)$ with vertex set

$$V = \{s, t\} \cup \{J_1^U, \ldots, J_{n_1+m-1}^U\} \cup \{J_1^L, \ldots, J_{n_2+m-1}^L\}$$

and arc set

$$\begin{aligned} A &= \{s \to J_i^U \mid i = 1, \ldots, n_1 + m - 2\} \\ &\quad \cup \{J_i^U \to J_j^L \mid i = 1, \ldots, n_1 + m - 1;\ j = 1, \ldots, n_2 + m - 1\} \\ &\quad \cup \{J_j^L \to t \mid j = 2, \ldots, n_2 + m - 1\}. \end{aligned}$$

The outgoing flow requirement at s and the incoming flow requirement at t are both equal to $m-1$. Also, the incoming flow requirement at $J^U_{n_1+m-1}$ and the outgoing flow requirement at J_1^L are both equal to 1, thus forcing $J^U_{n_1+m-1}$ and J_1^L to be included in U_0 and L_0, respectively. Arcs $s \to J_i^U$ and $J_j^L \to t$ have cost 0, while arc $J_i^U \to J_j^L$ has unit cost $-\pi_{ij}$ ($i = 1, \ldots, n_1 + m - 1;\ j = 1, \ldots, n_2 + m - 1$). Arc capacities are:

$$\begin{aligned} &u(s \to J_i^U) = 1; \\ &u(J_i^U \to J_j^L) = \begin{cases} 1, & \text{if } \pi_{ij} \geq \hat{T}_i + \hat{T}'_j - Z; \\ 0, & \text{otherwise;} \end{cases} \\ &u(J_j^L \to t) = 1. \end{aligned}$$

Let $\mathcal{N}(Z)$ denote this minimum cost flow problem. For any given value of Z, a desired bipartite matching M exists if and only if the optimal total cost of $\mathcal{N}(Z)$ is at most $mZ - D$. We let $Z_1 < Z_2 < \cdots < Z_r$ be the distinct values of $\hat{T}_i + \hat{T}'_j - \pi_{ij}$ for $J_i^U \in U$ and $J_j^L \in L$ and define $Z_{r+1} \equiv +\infty$. Note that the network is the same for every $Z \in [Z_k, Z_{k+1})$. Hence, Z_{LB} can be determined by using bisection search on $Z_1, Z_2, \ldots, Z_{r+1}$ as described in the following procedure.

Algorithm LB**:**

Step 1. Set $\ell \leftarrow 1$ and $u \leftarrow r + 1$. Set $k \leftarrow \lfloor (u + \ell)/2 \rfloor$.

Step 2. Solve the minimum cost network flow problem $\mathcal{N}(Z_k)$. If the optimal total cost of the solution is less than $mZ_{k+1} - D$, then set $u \leftarrow k$, otherwise set $\ell \leftarrow k + 1$.

Step 3. If $u = \ell$, then set Z_{LB} equal to the optimal total cost of $\mathcal{N}(Z_k)$ and stop. Otherwise, set $k \leftarrow \lfloor (u + \ell)/2 \rfloor$ and go to Step 2.

Note that in Step 2, if the optimal total cost of $\mathcal{N}(Z_k)$ is less than $mZ_{k+1} - D$, then $Z_{LB} < Z_{k+1}$ and we set $u \leftarrow k$. Otherwise, the optimal total cost of $\mathcal{N}(Z)$ is at least $mZ_{k+1} - D$ for all $Z < Z_{k+1}$, which implies $Z_{LB} \geq Z_{k+1}$, and therefore, we set $\ell \leftarrow k+1$. The number of distinct values of $\hat{T}_i + \hat{T}'_j - \pi_{ij}$ is no greater than n^2. Hence, the bisection search requires $O(\log r) \leq O(\log n^2) = O(\log n)$ iterations. Each iteration requires to solve a minimum cost network flow problem, which is solvable in $O((|A| \log W) \cdot (|A| + |V| \log |V|))$ time using a capacity scaling algorithm, where W is the largest supply/demand parameter or arc capacity (see Ahuja *et al.* [1], p. 395). In our application, $|V| = O(n)$, $|A| \leq O(n^2)$, and $W = m - 1$. Thus, the complexity of each iteration of Algorithm LB is $O(n^4 \log m)$. Therefore, the complexity of Algorithm LB is $O(n^4 \log n \log m)$.

This completes the description of our lower bound, and it is used in Section 6 to evaluate the heuristics presented in the next section.

5 Heuristic Algorithms and Analysis

We now develop a few efficient heuristics for solving our vehicle dispatching problem.

Heuristic $H1$:

Step 1. Apply the FAT rule to the set U of unloading jobs and let the resulting schedule be σ^{FAT}.

Step 2. Apply the LBT rule to the set L of loading jobs and let the resulting schedule be σ^{LBT}.

Step 3. Concatenate the partial schedules $\sigma^{FAT}, \sigma^{LBT}$ by arbitrarily matching the final unloading jobs with the leading loading jobs.

This heuristic has been presented in Bish *et al.* [3] who showed that it has a worst case error bound of 200% (i.e., $Z(\sigma^{H1})/Z(\sigma^*) \leq 3$) and a running time of $O(n)$. The next theorem provides an improved worst case performance guarantee. We let σ^{H1} be the schedule obtained by Heuristic $H1$ and σ^* be an optimal schedule.

Theorem 6 *$Z(\sigma^{H1})/Z(\sigma^*) \leq 2$ and this bound is tight.*

Proof: See Appendix.

We now analyze the expected performance of Heuristic $H1$. Assuming that the travel times between the quay crane and the yard locations of the containers are independent and identically distributed with a uniform distribution, the following theorem provides some interesting properties of Heuristic $H1$.

Theorem 7 *Suppose that job travel times $t_0(J)$ $(J \in U \cup L)$ are independent and uniformly distributed in the interval $[0, b]$, where $b > 0$. Then for $n > 2$, the following hold:*

(i) $E\left[\frac{Z(\sigma^{H1})}{Z(\sigma^*)}\right] \leq 1 + \frac{8m}{n-2}$;

(ii) $Pr\left(\frac{Z(\sigma^{H1})-Z(\sigma^*)}{Z(\sigma^*)} > \frac{8m}{(n-1)\eta}\right) \leq (\eta e^{1-\eta})^{n-1}$ *for all* $0 < \eta < 1$.

Proof: See Appendix.

Inequality (i) provides us with an upper bound on the expected performance ratio of Heuristic $H1$. It also shows that, as n approaches infinity, the expected performance of the heuristic is asymptotically optimal if the number of trucks, m, is held constant. This is consistent with the asymptotic analysis result of Bish *et al.* [3]. Inequality (ii), on the other hand, implies that the probability of the relative error being more than any constant $\epsilon > 0$ approaches 0 exponentially fast as n approaches infinity. Note that the validity of these inequalities is based on the assumption that the job travel times $t_0(J)$ are independent and uniformly distributed in the interval $[0, b]$. In fact, the proof of Theorem 7 can be generalized to the case in which the job travel times are uniformly distributed in an interval $[a, b]$, where $0 \leq a < b$.

Note that in Step 3 of Heuristic $H1$, we concatenate the partial schedules without considering the matching of final unloading jobs with the leading loading jobs as in the optimal algorithm described in Section 3. Hence, we can improve the heuristic by replacing the straightforward concatenation with an optimal matching. This results in the following heuristic.

Heuristic $H2$:

Step 1. Apply the FAT rule to the set U of unloading jobs and let the resulting schedule be σ^{FAT}.

Step 2. Apply the LBT rule to the set L of loading jobs and let the resulting schedule be σ^{LBT}.

Step 3. Concatenate the partial schedules $\sigma^{FAT}, \sigma^{LBT}$ optimally by matching the final unloading jobs with the leading loading jobs through solving a bottleneck assignment problem as described in Section 3.

Solving the bottleneck assignment problem requires $O(m^{2.5} \log m)$ time (see Section 3). Hence, the running time of Heuristic $H2$ is $O(\max\{n, m^{2.5} \log m\})$. Note that Theorems 6 and 7 also hold for Heuristic $H2$. By Theorem 6, the relative error of the solution generated by Heuristic $H2$ is no more than 100%. Using the same worst case example as in the proof of Theorem 6, we can show that this error bound remains tight for Heuristic $H2$. In other words, improving upon Heuristic $H1$ using step 3 does not change the worst case performance of the heuristic.

Note that Heuristics $H1$ and $H2$ use the same sets of final unloading and leading loading jobs and that these jobs are induced by σ^{FAT} and σ^{LBT}. The next heuristic departs from those and uses the sets identified in our lower bound procedure LB.

Heuristic $H3$:

For every iteration of Algorithm LB, do the following:

Step (i). For $i = 1, \ldots, n_1 + m - 2$, put J_i^U in U_0 if and only if there is a flow in arc $s \rightarrow J_i^U$ in the minimum cost flow solution of the current iteration of Algorithm LB. For $i = 2, \ldots, n_2+m-1$, put J_i^L in L_0 if and only if there is a flow in arc $J_i^U \rightarrow t$ in the minimum cost flow solution of the current iteration of Algorithm LB. Also, let $J_{n_1+m-1}^U \in U_0$ and $J_1^L \in L_0$.

Step (ii). Given the set U_0 of final unloading jobs, obtain a partial schedule using the FAT rule and let the resulting schedule be σ^{FAT}. Given the set L_0 of leading loading jobs, obtain a partial schedule using the LBT rule and let the resulting schedule be σ^{LBT}.

Step (iii). Concatenate the partial schedules $\sigma^{FAT}, \sigma^{LBT}$ optimally by matching the final unloading jobs with the leading loading jobs through solving a bottleneck assignment problem as described in Section 3.

Select the best solution among those generated by the above iterations.

The computational complexity of each iteration of Heuristic $H3$ is the same as that of Algorithm LB, which is $O(n^4 \log m)$. The number of iterations is $O(\log n)$, and hence, the complexity of $H3$ is $O(n^4 \log n \log m)$.

6 Computational Experiments

A computational study is performed to test the effectiveness of the proposed heuristics. The test data are generated randomly, while the parameter settings are selected in such a way that the actual operating conditions are reflected.

The average container throughput per vessel in Hong Kong, one of the world's busiest ports, during the period of January 1999 – March 2000 was 944.4 twenty-foot equivalent units (TEUs) (see Shabayek and Yeung [20]), where the majority of the containers had capacity of 2 TEUs, while the others were mostly 1 TEU in capacity. Typically, each vessel is served by 3 to 4 quay cranes. Therefore, we estimate that the maximum number of jobs (i.e., containers) per vessel handled by each quay crane is around 160. In our computational experiments, the number of jobs, $n_1 + n_2$, is set to 20, 40, 80, and 160, where half of the jobs are unloading jobs and half of them are loading jobs, i.e., $n_1 = n_2$. The average number of internal trucks per quay crane operating in such a terminal is between 4 and 6. Hence, in our computational experiments, m is set to 2, 4, and 8. The travel times of the internal trucks depend on the size and shape of the terminal. To ensure that our experiments cover a wide range of data settings, these travel times are randomly generated. For each problem instance, the one-way travel times, $t_0(J)$, are integers uniformly generated from the range $[1, 50]$ or $[1, 100]$. Thus, there are 24 combinations of m, $n_1 + n_2$, and travel time ranges. For each of these combinations, we generate 10 random problems. After drawing a value $t_0(J)$, we

randomly generate an integer $x(J) \in [1, t_0(J) - 1]$ and assume that the location associated with job J has coordinates $(x(J), t_0(J) - x(J))$. The crane is located at position $(0, 0)$, and hence $t_0(J)$ is the rectilinear distance between the crane and job $J \in U \cup L$. Accordingly,

$$t(J, J') = \Big|x(J) - x(J')\Big| + \Big|(t_0(J) - x(J)) - (t_0(J') - x(J'))\Big|.$$

Our algorithms are applicable to any distance metric. Rectilinear distances are used to more closely capture the motivating application. The crane processing times, $s(J)$, are integers uniformly generated from the range $[1, 5]$ to reflect the fact that they are generally shorter than the travel times of jobs.

We programmed the optimal algorithm presented in Section 3, the Heuristics $H1$, $H2$, $H3$, as well as the lower bound procedure presented in Section 4. We conducted our experiemnts on a Pentium IV processor running at 2.0 GHz. For problem instances with $n_1 + n_2 \leq 40$, we use $R = [Z(\sigma^H) - Z(\sigma^*)]/Z(\sigma^*)$ to evaluate the effectiveness of Heuristic H ($H = H1, H2, H3$), where $Z(\sigma^H)$ represents the makespan of the schedule generated by the heuristic. The $Z(\sigma^*)$ values are obtained by running the optimal algorithm. This takes less than 1 hour for a 40-job instance and less than 10 minutes for a 20-job instance. For instances with $n_1 + n_2 \geq 80$, we use $R = [Z(\sigma^H) - Z_{LB}]/Z_{LB}$ to evaluate our heuristics. We let $avg(R)$ denote the average value of R over the 10 randomly generated problems. In Table 1, we report the $avg(R)$ value and the average CPU time for each combination of m, $n_1 + n_2$, $t_{\max}$ and for each heuristic.

Our results indicate that Heuristic $H2$ provides significant improvement over the solution provided by $H1$. Both heuristics' solutions, however, are significantly inferior to those of $H3$, which are near optimal for $n_1 + n_2 \geq 80$. Moreover, the performance of our heuristics is underestimated when $n_1 + n_2 \geq 80$ because the lower bound Z_{LB} rather than $Z(\sigma^*)$ is used to compute R. By construction, Heuristic $H2$ dominates $H1$. It is rare but not impossible for $H2$ to provide a better solution than $H3$. The figures in Table 1 demonstrate that the performance of all heuristics gets better as the number of jobs gets larger. This is consistent with Theorem 7. From Table 1, it is clear that the better performance of $H3$ is achieved at the expense of increased computational time. However, the CPU times of $H3$ are no more than a few seconds among all the test problems.

We conclude that $H3$ is a great tool to obtain near optimal solutions for large problems.

7 Conclusions

We have developed optimal and heuristic algorithms for the single quay crane vehicle dispatching problem. Our optimal algorithm is efficient when the number of trucks is small (e.g., 2 or 3 trucks). We have also suggested Heuristics $H2$ and $H3$ for solving problems with more trucks. Heuristics $H1$ and $H2$ have a worst case error bound of 100%, and their expected relative errors approach zero exponentially fast as then number of jobs increases. Computational experiments indicate that all three heuristics are effective, with the performance of $H3$ dominating those of $H1$ and $H2$.

Note that in our model, the number of trucks is assumed to be a given parameter. In practice, the number of trucks is quite flexible (it is possible to introduce additional trucks to the existing operation). Furthermore, in our model we consider only the container loading/unloading operation of a single quay crane and a single ship. In reality, trucks can be shared among different quay cranes and different ships. Therefore, an interesting future research direction is to consider the more general setting of scheduling trucks for multiple cranes and multiple ships, as well as the issue of determining the optimal number of trucks for the entire container terminal so as to minimize the average turnaround time of the ships.

References

[1] Ahuja, R. K., T. L. Magnanti and J. B. Orlin, *Network Flows: Theory, Algorithms, and Applications,* Prentice Hall, Englewood Cliffs, NJ, 1993.

[2] Bish, E. K., "A multiple-crane-constrained scheduling problem in a container terminal", *European Journal of Operational Research,* **144,** 83–107, 2003.

[3] Bish, E. K., F. Y. Chen, Y. T. Leong, Q. Liu, B. L. Nelson, J. W. C. Ng and D. Simchi-Levi, "Dispatching vehicles in a mega container terminal", working paper.

[4] Bish, E. K., Y. T. Leong, C.-L. Li, J. W. C. Ng and D. Simchi-Levi, "Analysis of a new scheduling and location problem", *Naval Research Logistics,* **48,** 363–385, 2001.

[5] Bostel, N. and P. Dejax, "Models and algorithms for container allocation problems on trains in a rapid transshipment shunting yard", *Transportation Science,* **32,** 370–379, 1998.

[6] Brown, G. G., K. J. Cormican, S. Lawphongpanich and D. B. Widdis, "Optimizing submarine berthing with a persistence incentive", *Naval Research Logistics,* **44,** 301–318, 1997.

[7] Coffman, E. G., Jr. and E. N. Gilbert, "On the expected relative performance of list scheduling", *Operations Research,* **33,** 548–561, 1985.

[8] Daganzo, C. F., "The crane scheduling problem", *Transportation Research B,* **23B,** 159–175, 1989.

[9] Daganzo, C. F., "The productivity of multipurpose seaport terminals", *Transportation Science,* **24,** 205–216, 1990.

[10] de Castilho, B. and C. F. Daganzo, "Handling strategies for import containers at marine terminals", *Transportation Research B,* **27B,** 151–166, 1993.

[11] Easa, S. M., "Approximate queueing models for analyzing harbor terminal operations", *Transportation Research B,* **21B,** 269–286, 1987.

[12] Gransberg, D. D. and J. P. Basilotto, "Cost engineering optimum seaport capacity", *Cost Engineering,* **40,** *9,* 28–32, 1998.

[13] Kim, K. H. and K. Y. Kim, "An optimal routing algorithm for a transfer crane in port container terminals", *Transportation Science,* **33,** 17–33, 1999.

[14] Kim, K. H. and Y. M. Park and K. R. Ryu, "Deriving decision rules to locate export containers in container yards", *European Journal of Operational Research,* **124,** 89–101, 2000.

[15] Lawler, E. L., J. K. Lenstra, A. H. G. Rinnooy Kan and D. B. Shmoys, "Sequencing and scheduling: algorithms and complexity", in S. C. Graves, A. H. G. Rinnooy Kan and P. H. Zipkin (Eds.), *Handbooks in Operations Research and Management Science, Volume 4: Logistics of Production and Inventory,* North-Holland, Amsterdam, 1993.

[16] Li, C.-L., X. Cai and C.-Y. Lee, "Scheduling with multiple-job-on-one-processor pattern", *IIE Transactions,* **30,** 433–445, 1998.

[17] Lim, A., "The berth planning problem", *Operations Research Letters,* **22,** 105–110, 1998.

[18] Peterkofsky, R. I. and C. F. Daganzo, "A branch and bound solution method for the crane scheduling problem", *Transportation Research B,* **24B,** 159–172, 1990.

[19] Powell, W. B. and T. A. Carvalho, "Real-time optimization of containers and flatcars for intermodal operations", *Transportation Science,* **32,** 110–126, 1998.

[20] Shabayek, A. A. and W. W. Yeung, "A simulation model for the Kwai Chung container terminals in Hong Kong", *European Journal of Operational Research,* **140,** 1–11, 2002.

Appendix

Proof of Theorem 6: Observe that $Z(\sigma^*) \geq Z(\sigma^{FAT})$. This is because σ^* is the optimal schedule for the jobs $U \cup L$ while σ^{FAT} is the optimal schedule for the subset of jobs U, and therefore, the makespan of the former must be no less than the makespan of the latter. Similarly, $Z(\sigma^*) \geq Z(\sigma^{LBT})$. These imply that $Z(\sigma^{H1}) \leq Z(\sigma^{FAT}) + Z(\sigma^{LBT}) \leq 2Z(\sigma^*)$, or equivalently, $Z(\sigma^{H1})/Z(\sigma^*) \leq 2$.

To see that the bound of 2 is tight, consider an instance with $m+1$ unloading jobs and 1 loading job. The one-way travel times and quay crane processing requirements of the unloading jobs are $t_0(J_1^U) = N$, $t_0(J_i^U) = N+1$ $(i = 2, 3, \ldots, m)$, $t_0(J_{m+1}^U) = 1$, and $s(J_i^U) = 0$ $(i = 1, 2, \ldots, m+1)$. The one-way travel time and crane processing requirement of the loading job are $t_0(J_1^L) = N$ and $s(J_1^L) = 0$. Yard locations of J_1^U and J_1^L are on one side of the quay crane, while yard locations of $J_2^U, J_3^U, \ldots, J_{m+1}^U$ are on the other side of the crane (see Figure 2(a)). Hence, the travel times between yard locations of loading and unloading jobs are: $t(J_1^U, J_1^L) = 0$, $t(J_i^U, J_1^L) = 2N+1$ $(i = 2, \ldots, m)$, and $t(J_{m+1}^U, J_1^L) = N+1$. Heuristic $H1$ will generate a schedule with makespan of $4N+2$ as shown in Figure 2(b). An optimal schedule is to assign J_{m+1}^U to a truck different from that of J_1^U (see Figure 2(c)). This allows a truck to serve J_1^L immediately after processing J_1^U, where these two jobs are of zero distance apart. The makespan of the optimal schedule is $2N+4$. Thus, in this example, $Z(\sigma^{H1})/Z(\sigma^*) = (4N+2)/(2N+4) \to 2$, as $N \to \infty$. This completes the proof of the theorem. ∎

Proof of Theorem 7: To prove the theorem, we first show that $Z(\sigma^{H1}) \leq Z(\sigma^*) + 4t_{\max}$, where $t_{\max} = \max_{J \in U \cup L}\{t_0(J)\}$. Recall that T_i is the completion time of $J_{\lambda_i}^U$ in schedule σ^{FAT}, excluding the time to travel back to the quay crane afterward $(i = 1, \ldots, n_1 + m - 1)$, and T'_j is the elapsed time to complete all loading jobs in σ^{LBT} starting from $J_{\mu_j}^L$, excluding the time to travel from the quay crane to the yard location of $J_{\mu_j}^L$ $(j = 1, \ldots, n_2 + m - 1)$. Clearly, there exists a pair of jobs $J_{\lambda_i}^U, J_{\mu_j}^L$

assigned to the same truck in the heuristic solution such that

$$Z(\sigma^{H1}) = T_i + t(J^U_{\lambda i}, J^L_{\mu_j}) + T'_j. \tag{3}$$

Since all unloading jobs are scheduled using the FAT rule, the finish time of loading $J^U_{\lambda_i}$ onto the truck at the quay crane is $T_i - t_0(J^U_{\lambda_i})$ and this is the earliest possible time for completing the unloading of $J^U_{\lambda_i}$ at the quay crane. Similarly, the shortest possible elapsed time to complete all loading jobs in σ^{LBT} starting from unloading of $J^L_{\mu_j}$ from the truck at the quay crane is $T'_j - t_0(J^L_{\mu_j})$. Note that in every feasible schedule, $J^U_{\lambda_i}$ is processed at the quay crane prior to $J^L_{\mu_j}$. Thus,

$$[T_i - t_0(J^U_{\lambda_i})] + [T'_j - t_0(J^L_{\mu_j})] \le Z(\sigma^*),$$

and therefore, (3) becomes

$$\begin{aligned} Z(\sigma^{H1}) &\le Z(\sigma^*) + t_0(J^U_{\lambda_i}) + t_0(J^L_{\mu_j}) + t(J^U_{\lambda_i}, J^L_{\mu_j}) \\ &\le Z(\sigma^*) + 2t_0(J^U_{\lambda_i}) + 2t_0(J^L_{\mu_j}) \quad \text{(triangle inequality)} \\ &\le Z(\sigma^*) + 4t_{\max}. \end{aligned} \tag{4}$$

We can now proceed with (i). Inequality (4) yields

$$\frac{Z(\sigma^{H1})}{Z(\sigma^*)} \le 1 + \frac{4t_{\max}}{Z(\sigma^*)}. \tag{5}$$

Observe that $Z(\sigma^*) \ge Z(\sigma^{FAT}) \ge \frac{1}{m}\sum_{J\in U} 2t_0(J)$ and $Z(\sigma^*) \ge Z(\sigma^{LBT}) \ge \frac{1}{m}\sum_{J\in L} 2t_0(J)$. Hence,

$$Z(\sigma^*) \ge \frac{1}{2}\cdot\Big[\frac{1}{m}\sum_{J\in U} 2t_0(J) + \frac{1}{m}\sum_{J\in L} 2t_0(J)\Big] = \frac{1}{m}\cdot t_{\text{sum}}, \tag{6}$$

where $t_{\text{sum}} = \sum_{J\in U\cup L} t_0(J)$. From inequalities (5) and (6), we have

$$\frac{Z(\sigma^{H1})}{Z(\sigma^*)} \le 1 + 4m\cdot\frac{t_{\max}}{t_{\text{sum}}}, \tag{7}$$

which implies that

$$E\Big[\frac{Z(\sigma^{H1})}{Z(\sigma^*)}\Big] \le 1 + 4m\cdot E\Big[\frac{t_{\max}}{t_{\text{sum}}}\Big]. \tag{8}$$

Let $X_{(n)}$ be the n-th order statistic of n random variables X_j uniformly distribution in $[0, b]$ ($j = 1, 2, \ldots, n$), and let $Y_1, \ldots, Y_{n-1}$ be the other $n-1$ random variables in the set $\{X_1, \ldots, X_n\}$. Then,

it is easy to check that $\frac{Y_1}{X_{(n)}}, \frac{Y_2}{X_{(n)}}, \ldots, \frac{Y_{n-1}}{X_{(n)}}$ are independent uniform random variables distributed in $[0,1]$. We make use of the following inequality in Coffman and Gilbert ([7], eq. (10)):

$$E\Big[\frac{1}{1+z_{n-1}}\Big] \leq \frac{2}{n-2}$$

for $n > 2$, where z_{n-1} denotes the sum of $n-1$ independent uniform random variables distributed in $[0,1]$. Hence,

$$E\Big[\frac{X_{(n)}}{\sum_{j=1}^{n} X_j}\Big] = E\Big[\frac{1}{1+\sum_{j=1}^{n-1} Y_j/X_{(n)}}\Big] \leq \frac{2}{n-2},$$

for $n > 2$. This implies that $E[t_{\max}/t_{\text{sum}}] \leq \frac{2}{n-2}$ when $n > 2$. Combining this with inequality (8) yields (i).

To prove (ii), we note that inequality (7) implies

$$Pr\Big(\frac{Z(\sigma^{H1})}{Z(\sigma^*)} > 1 + \frac{8m}{(n-1)\eta}\Big) \leq Pr\Big(1 + 4m\cdot\frac{t_{\max}}{t_{\text{sum}}} > 1 + \frac{8m}{(n-1)\eta}\Big),$$

which in turn implies that

$$Pr\Big(\frac{Z(\sigma^{H1}) - Z(\sigma^*)}{Z(\sigma^*)} > \frac{8m}{(n-1)\eta}\Big) \leq Pr\Big(\frac{t_{\max}}{t_{\text{sum}}} > \frac{2}{(n-1)\eta}\Big) \leq Pr\Big(\frac{t_{\text{sum}} - t_{\max}}{t_{\max}} < \frac{(n-1)\eta}{2}\Big). \quad (9)$$

Coffman and Gilbert ([7], eq. (20)) have shown that

$$Pr\Big(z_{n-1} \leq \frac{\eta(n-1)}{2}\Big) \leq (\eta e^{1-\eta})^{n-1},$$

which implies

$$Pr\Big(\frac{\sum_{j=1}^{n} X_j - X_{(n)}}{X_{(n)}} \leq \frac{\eta(n-1)}{2}\Big) = Pr\Big(\sum_{j=1}^{n-1} \frac{Y_j}{X_{(n)}} \leq \frac{\eta(n-1)}{2}\Big) \leq (\eta e^{1-\eta})^{n-1}.$$

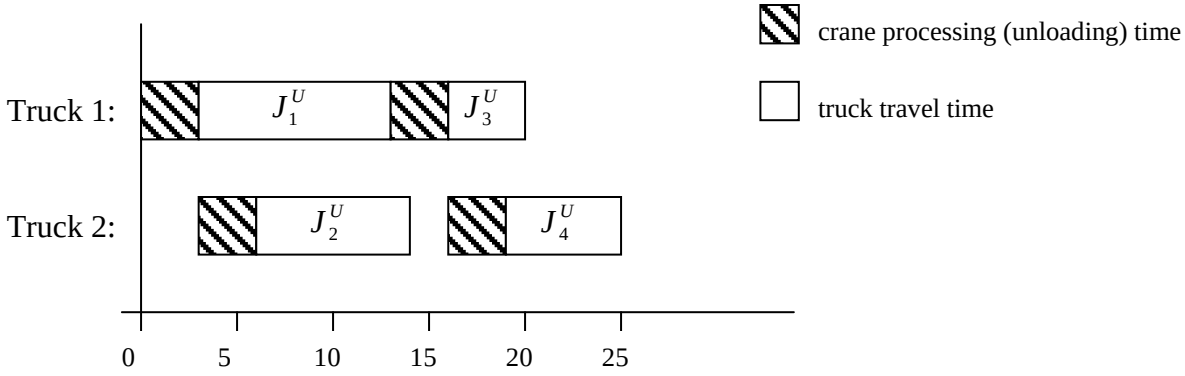
Hence,

$$Pr\Big(\frac{t_{\text{sum}} - t_{\max}}{t_{\max}} \leq \frac{\eta(n-1)}{2}\Big) \leq (\eta e^{1-\eta})^{n-1}.$$

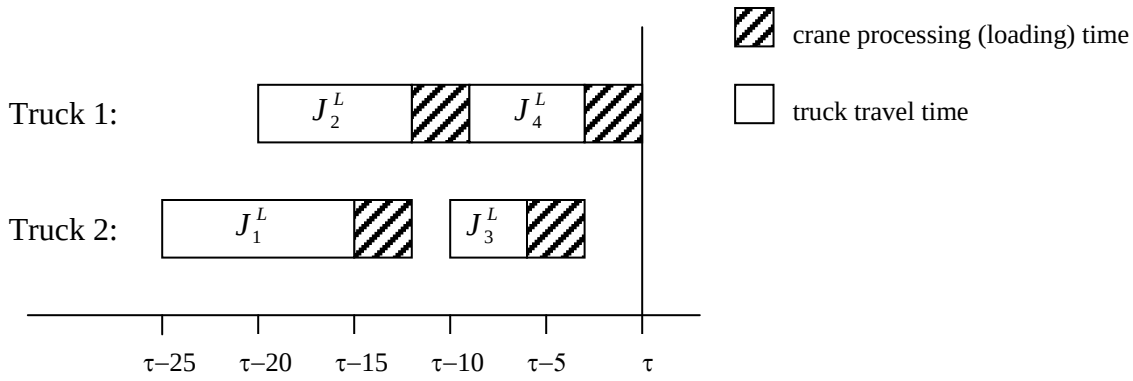
Combining this with inequality (9) yields (ii). ∎

Table 1: Relative errors of heuristics

m	n_1+n_2	travel times	$avg(r)\times 100\%$			average CPU time (sec.)		
			$H1$	$H2$	$H3$	$H1$	$H2$	$H3$
2	20	$[1,50]$	7.0%	5.4%	3.1%	0.0	0.0	0.0
		$[1,100]$	6.1%	4.8%	2.8%	0.0	0.0	0.0
	40	$[1,50]$	5.7%	4.4%	2.7%	0.1	0.1	0.8
		$[1,100]$	5.0%	3.9%	2.5%	0.1	0.1	0.9
	80	$[1,50]$	6.5%	5.3%	3.0%	0.1	0.2	2.9
		$[1,100]$	6.1%	4.8%	3.0%	0.2	0.2	3.1
	160	$[1,50]$	5.6%	4.6%	2.5%	0.3	0.3	5.1
		$[1,100]$	5.2%	4.0%	2.1%	0.2	0.3	4.9
4	20	$[1,50]$	6.1%	4.8%	3.5%	0.0	0.0	0.0
		$[1,100]$	5.7%	4.5%	3.1%	0.0	0.0	0.0
	40	$[1,50]$	5.1%	3.8%	2.9%	0.1	0.1	1.1
		$[1,10]$	4.6%	3.5%	2.5%	0.2	0.2	0.9
	80	$[1,50]$	5.2%	4.1%	2.4%	0.2	0.3	2.9
		$[1,100]$	5.1%	3.8%	2.2%	0.2	0.2	3.0
	160	$[1,50]$	4.6%	3.3%	1.8%	0.2	0.3	5.0
		$[1,10]$	4.2%	2.9%	1.5%	0.3	0.4	5.2
8	20	$[1,50]$	4.5%	3.3%	1.3%	0.0	0.0	0.0
		$[1,100]$	4.1%	2.9%	1.2%	0.0	0.0	0.0
	40	$[1,50]$	4.0%	2.7%	1.0%	0.1	0.1	1.2
		$[1,100]$	4.0%	2.5%	1.0%	0.2	0.2	1.3
	80	$[1,50]$	4.0%	3.0%	0.9%	0.2	0.2	3.1
		$[1,100]$	4.2%	3.1%	1.1%	0.3	0.3	3.5
	160	$[1,50]$	4.2%	3.1%	0.9%	0.4	0.4	5.3
		$[1,100]$	3.9%	2.6%	1.0%	0.4	0.4	6.1

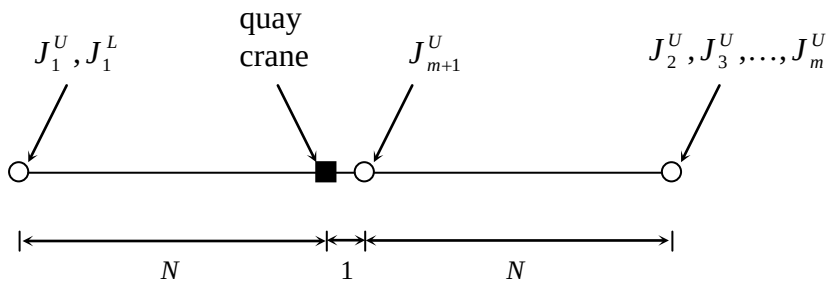


(a) Schedule of unloading jobs obtained by the FAT rule

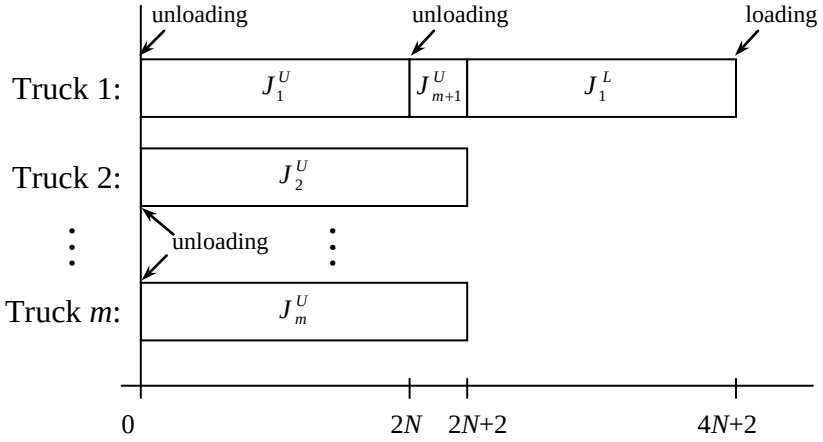


(b) Schedule of loading jobs obtained by the LBT rule

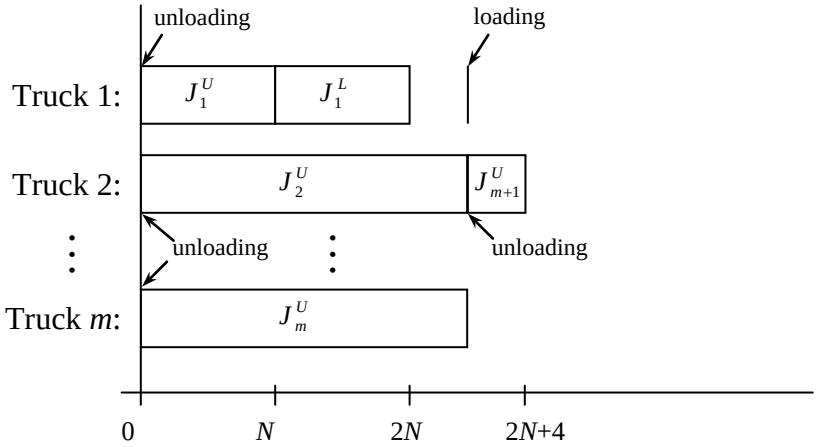
Figure 1. Examples of FAT and LBT rules



(a) Yard locations of jobs



(b) Heuristic solution



(c) Optimal solution

Figure 2. Example in the proof of Theorem 6